Extraction multidimensionnelle et multicritère de données émergentes pour l'équilibrage de jeux vidéo

Mickaël Martin Nevot

▶ **To cite this version:**

Mickaël Martin Nevot. Extraction multidimensionnelle et multicritère de données émergentes pour l'équilibrage de jeux vidéo. Informatique [cs]. Aix-Marseile Université, 2023. Français. ⟨NNT : 2020AIXM0001⟩. ⟨tel-04763342⟩

HAL Id: tel-04763342

https://hal.science/tel-04763342v1

Submitted on 1 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY-NC-ND 4.0 - Attribution - Non-commercial use - No Derivative Works - International License

Aix*Marseille université
Socialement engagée

NNT/NL : 2020AIXM0001/001ED000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 09 novembre 2023 par

Mickaël MARTIN NEVOT

Extraction multidimensionnelle et multicritère de données émergentes pour l'équilibrage de jeux vidéo

Discipline
Informatique

École doctorale
Ecole Doctorale en Mathématiques et Informatique de Marseille

Laboratoire/Partenaires de recherche
Laboratoire d'Informatique et Systèmes — UMR 7020
Institut Universitaire de Technologie d'Aix-Marseille
Société d'accélération du transfert de technologies Sud-Est

Composition du jury

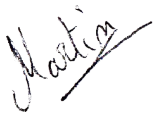
Pascal PONCELET Université de Montpellier	Rapporteur
Fadila BENTAYEB Université Lumière Lyon 2	Rapporteuse
Julian ALVAREZ Université de Lille	Examinateur
Mohamed QUAFAFOU Université d'Aix-Marseille	Président du jury
Lotfi LAKHAL Université d'Aix-Marseille	Directeur de thèse
Sébastien NEDJAR Université d'Aix-Marseille	Co-directeur de thèse

LIS Laboratoire d'Informatique & Systèmes
iUT Institut Universitaire de Technologie Aix*Marseille Université
SATT SUD-EST ACCÉLÉRONS LE TRANSFERT DE TECHNOLOGIES

Affidavit

Je soussigné, Mickaël Martin Nevot, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de Rosine Cicchetti, de Lotfi Lakhal et de Sébastien Nedjar, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat. Ce travail n'a pas été soumis auparavant en France ou à l'étranger, dans une version identique ou similaire, à un organisme examinateur.

Fait à Aix-en-Provence le 12 juillet 2023.



Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Liste de publications

Liste des publications réalisées dans le cadre du projet de thèse :

1. Mickaël MARTIN NEVOT, Sébastien NEDJAR et Lotfi LAKHAL. « Hierarchical data-cubes ». In : *Journal of Computer and Communications* Vol.11 No.6, June 2023 (juin 2023), p. 43-72. DOI : 10.4236/jcc.2023.116004 (MARTIN NEVOT, NEDJAR et LAKHAL 2023)
2. Mickaël MARTIN NEVOT, Sébastien NEDJAR, Lotfi LAKHAL et Rosine CICCHETTI. « C-Idea - A Fast Algorithm for Computing Emerging Closed Datacubes ». In : *Utilizing Big Data Paradigms for Business Intelligence*. 2019, p. 129-169. DOI : 10.4018/978-1-5225-4963-5.ch005. URL : https://hal.archives-ouvertes.fr/hal-02446921 (MARTIN NEVOT, NEDJAR, LAKHAL et CICCHETTI 2019).
3. Gaylor RABU, Morgane REVERCHON-BILLOT, Mickaël MARTIN NEVOT et al. « Glossaire ». In : *Les enjeux juridiques de l'e-sport*. Sous la dir. de Presses Universitaires D'AIX-MARSEILLE. Aix-en-Provence ; Aix-Marseille, sept. 2017, p. 591-615. ISBN : 9782731410709. URL : https://hal.archives-ouvertes.fr/hal-03081749 (RABU, REVERCHON-BILLOT, MARTIN NEVOT et al. 2017)

Participation aux conférences et écoles d'été au cours de la période de thèse :

Résumé

Cette thèse propose d'appliquer la fouille de données multidimensionnelles à des données de jeux vidéo afin d'offrir des méthodes et des outils décisionnels pour aider leurs concepteurs à équilibrer leurs mécaniques, dynamiques et esthétiques.

Dans un contexte OLAP, notamment en application au domaine des jeux vidéo, les dimensions peuvent souvent être des hiérarchies d'attributs liés entre eux. Cet aspect est peu exploré et pris en compte par les approches de représentation et de calcul des cubes de données. Nous proposons une caractérisation formelle de structures pour les entrepôts de données avec dimensions hiérarchiques.

Combiner l'analyse décisionnelle multicritère et la découverte de renversements de tendances permet d'extraire les données globalement optimales pour un ensemble de préférences de choix d'une recherche puis d'observer la significativité statistique de leur évolution selon un critère décisionnel. L'utilisation des cubes de données permet d'explorer et d'appréhender les liens entre plusieurs niveaux d'agrégation. À cette fin, nous caractérisons le Skycube émergent en présentant une représentation réduite grâce aux treillis de Concepts reposant sur la théorie des ensembles en Accord.

Nous soumettons également un outil de classification efficace de Skyline afin d'organiser de manière logique et cohérente les informations recherchées.

L'objectif de ce travail de recherche est d'offrir une solution pour assister l'équilibrage de la conception de jeux vidéo. Pour cela, nous étendons la plateforme algorithmique IDEA en mettant à disposition la méthode TAGED permettant de calculer ces nouvelles structures en fonction de la classification des dispositifs ludiques.

Mots clefs : OLAP, entrepôts de données, *big data*, fouille de données multidimensionnelles, fouille de données de jeux vidéo, Conception de jeu, treillis cube, cubes de données, fermés, hiérarchiques, émergents, analyse de renversement de tendances, dimensions hiérarchiques, analyse décisionnelle multicritère, Skyline, Skycube, treillis de concepts.

Abstract

This thesis proposes to apply multidimensional data mining to video games data in order to provide decision-making methods and tools to help designers balance their mechanics, dynamics and aesthetics.

In an OLAP context, particularly applied to the video games field, dimensions can often be hierarchies of interrelated attributes. This aspect is underexplored and underconsidered by datacubes representation and calculation approaches. We propose a formal characterization of structure for data warehouses with hierarchical dimensions.

Combining multi-criteria decision-making analysis and trend reversal discovery allows us the possibility to extract globally optimal data for a set of preferential query parameters, thus observing the statistical significance in their evolution according to the decision criterion. Usage of datacubes makes it possible to explore and understand links between multiple aggregation levels. To this end, we characterize the emergent Skycube by presenting a reduced representation using Agree Concept lattice for multidimensional database analysis.

We also present an efficient Skyline classification tool in order to organise queried data in a logical and coherent fashion.

The main goal of this research work is to offer a tool-assisted solution in video game equilibrium design. Therefore, we extend the IDEA algorithmic platform by providing the TAGED method, to compute these new structures depending on video game classification.

Keywords: databases, OLAP, data warehouses, big data, business intelligence, multidimensional data minig, video game data mining, game design, cube lattice, datacube, closed datacube, trend analysis, emerging datacube, hierarchical dimensions, hierarchical datacube, multi-criteria decision-making analysis, Skyline, Skycube, Agree sets, concept lattices, classification, IDEA.

Remerciements

Je souhaite ici rendre hommage et exprimer ma profonde gratitude à tous ceux qui, de près ou de loin, ont contribué à la réalisation de cette thèse et à son aboutissement.

En premier lieu, je tiens à montrer mon immense reconnaissance envers Rosine Cicchetti, feu professeur à l'Université d'Aix-Marseille, ma regrettée première directrice de thèse, Lotfi Lakhal, professeur à l'Université d'Aix-Marseille, mon second directeur de thèse, pour m'avoir épaulé et apporté tant de précisions dans son domaine d'expertise, et Sébastien Nedjar, maître de conférences à l'Université d'Aix-Marseille, mon co-directeur de thèse, pour son implication extrêmement généreuse, sans laquelle, tout particulièrement, cette aventure n'aurait même pas eu lieu, pas plus qu'elle aurait été menée à bien non plus.

Je souhaite ensuite remercier Pascal Poncelet, professeur à l'Université de Montpellier et Fadila Bentayeb, professeur à l'Université Lumière Lyon 2, les rapporteurs de ma thèse.

Je tiens à saisir cette occasion pour adresser mes remerciements et ma reconnaissance aux responsables et au personnel de l'Université d'Aix-Marseille.

En particulier, je désire remercier Liva Ralaivola, Mustapha Ouladsine et Frédéric Béchet, professeurs à l'Université d'Aix-Marseille et directeurs successifs du Laboratoire d'Informatique et Systèmes durant ma thèse. Je souhaite également remercier Mohamed Quafafou, professeur à l'Université d'Aix-Marseille, président de mon jury de thèse, et Alain Casali, maître de conférences à l'Université d'Aix-Marseille, respectivement directeur et directeur adjoint de mon équipe de recherche Data Mining at scale (DANA). De même, je remercie Jean-Luc Massat, maître de conférences à l'Université d'Aix-Marseille, pour son soutien et son aide.

J'adresse mes remerciements à Julian Alvarez, maître de conférences à l'Université de Lille, mon parrain de thèse, membre de mon comité de suivi de thèse et examinateur de mon jury de thèse, ainsi que Petru Valicov, maître de conférences à l'Université de Montpellier, également membre de mon comité de suivi de thèse.

Je tiens aussi à remercier Marguerite Leenhardt et Franck Orsatti de la Société d'accélération du transfert de technologies Sud-Est pour leur confiance et leur professionnalisme.

J'ai une pensée enthousiaste et nostalgique pour Christophe Delagarde, ainsi que nos nombreuses séances de travail, et le remercie pour son excellente implication ayant permit de donner du corps à mes idées.

J'adresse ma gratitude au Yéti pour son aide précieuse sans laquelle Léviathor n'aurait pas su aussi bien nous assister.

Je souhaite remercier grandement l'ensemble des relecteurs, même ceux occasionnels, de ce mémoire de thèse.

Le modèle de thèse AMU n'existerait pas sans la contribution des doctorants. Je souhaite remercier tout particulièrement Mickaël Bojados, Flora Cordoleani et Florian Caullery pour leur aide précieuse et la qualité de leurs fichiers sources LaTeX. La mise à jour effectuée en 2018 doit beaucoup à l'excellent travail de Dorian Depriester.

Je remercie de tout mon cœur mes amis, et particulièrement Cyril et Laurent pour leur soutien et leurs encouragements.

J'ai une pensée constante pour mes proches, présents, absents et perdus. J'en ai des toutes particulières pour Robert Martin, mon papa, et Angel Nevot, mon papi, tous deux partis trop tôt.

J'ai une tendre pensée pour mes enfants, Adam et Arthur.

Je dédie cette thèse à ma femme, Juliana, pour sa présence et son soutien constants, pour moi et les enfants, et sa bonhomie m'ayant porter à chaque instant dans cette réalisation.

Table des matières

Affidavit 2

Liste de publications et participation aux conférences 3

Résumé 4

Abstract 5

Remerciements 6

Table des matières 8

Table des figures 12

Liste des tableaux 14

Liste des algorithmes 16

1. Introduction 17
1.1. Survol des travaux existants . . . 17
1.2. Cadre de la thèse . . . 20
1.3. Synthèse des contributions . . . 20
1.4. Organisation du mémoire . . . 23

I. État de l'art 25

2. Analyse des bases de données 26
2.1. Introduction . . . 27
2.2. Cube de données . . . 27
2.2.1. Introduction . . . 27
2.2.2. Rappels théoriques . . . 29
2.2.3. Cas d'utilisation . . . 31
2.2.4. Définition formelle du cube de données . . . 34
2.2.5. Calcul du cube de données . . . 44
2.2.6. Synthèse . . . 53
2.3. Cube de données fermé . . . 55
2.3.1. Introduction . . . 55
2.3.2. Cadre formel du cube de données fermé . . . 56
2.3.3. Définition formelle du cube de données fermé . . . 69

2.3.4. Synthèse . . . 72
2.4. Skyline et Skycube . . . 74
2.4.1. Introduction . . . 74
2.4.2. L'opérateur Skyline . . . 75
2.4.3. Skycube . . . 81
2.4.4. Synthèse . . . 84
2.5. Conclusion . . . 84

3. Cube de données émergent **86**
3.1. Introduction . . . 86
3.2. Concepts . . . 87
3.3. Bordures pour le cube de données émergent . . . 91
3.3.1. Bordures [L; U] . . . 93
3.4. Calcul du cube émergent et de ses bordures . . . 95
3.4.1. Cube de données émergent en SQL . . . 95
3.4.2. L'algorithme du cube de données émergent : E-IDEA . . . 98
3.4.3. F-IDEA . . . 104
3.5. Cube de données fermé émergent . . . 108
3.5.1. L-cube de données fermé émergent . . . 110
3.5.2. L'algorithme de cube de données fermé émergent : $\mathbb{C}$-IDEA . . . 116
3.6. Conclusion . . . 117

4. Conception de jeux vidéo **120**
4.1. Introduction . . . 121
4.2. Jouet, jeu et jeu vidéo . . . 122
4.3. Approche formelle de la conception de jeux vidéo . . . 123
4.3.1. Mécaniques . . . 124
4.3.2. Dynamiques . . . 124
4.3.3. Esthétiques . . . 124
4.4. Classification . . . 124
4.4.1. Classification de modèles de joueurs . . . 124
4.4.2. Classification de jeux vidéo . . . 126
4.5. Gameplay . . . 130
4.5.1. Méthodologie de conception de jeu vidéo . . . 130
4.5.2. Fun . . . 130
4.5.3. Flow . . . 132
4.5.4. Défi, incertitude, difficulté et virtuosité . . . 133
4.5.5. Boucle de jeu . . . 135
4.5.6. Motivations . . . 136
4.6. Équilibrage . . . 137
4.6.1. Domaines d'équilibrage . . . 138
4.6.2. La théorie des jeux . . . 139
4.7. Conclusion . . . 140

II. Contribution **142**

5. Cubes de données hiérarchiques **143**
5.1. Introduction et motivations . . . 143
5.2. Cas d'utilisation . . . 144
5.2.1. Open Match 3 . . . 144
5.2.2. Classification atomique d'Open Match 3 . . . 146
5.2.3. Relation exemple . . . 147
5.3. Dimension hiérarchique . . . 153
5.3.1. Types de hiérarchies . . . 153
5.3.2. Cadre formel d'une dimension hiérarchique . . . 154
5.3.3. Définition formelle d'une dimension hiérarchique . . . 156
5.4. Cube de données hiérarchique . . . 166
5.5. Cube de données hiérarchique fermé . . . 169
5.6. Conclusion . . . 172

6. Skycube émergent **174**
6.1. Introduction et motivations . . . 174
6.2. Cas d'utilisation . . . 175
6.2.1. Pokémon Showdown! . . . 175
6.2.2. Kaggle . . . 181
6.2.3. Classification atomique de Pokémon Showdown! . . . 181
6.2.4. Relation exemple . . . 182
6.3. Treillis des Concepts Accords et Skylines . . . 188
6.3.1. Treillis des partitions . . . 188
6.3.2. Ensembles en accord . . . 190
6.3.3. Concepts Accords d'une relation . . . 191
6.3.4. Treillis des Concepts Skylines et Skycube . . . 195
6.3.5. Évaluations expérimentales . . . 201
6.3.6. Comparaison avec les travaux antérieurs . . . 204
6.4. Calcul du Skycube émergent . . . 205
6.5. Skycube fermé émergent . . . 209
6.6. Conclusion . . . 213

7. CoSky **215**
7.1. Introduction et motivations . . . 215
7.2. Cas d'utilisation . . . 216
7.3. Classement de Skyline . . . 219
7.3.1. Méthode dp-idp . . . 219
7.3.2. Amélioration de la méthode dp-idp . . . 220
7.3.3. Méthode CoSky . . . 222
7.3.4. Méthode top-k . . . 227
7.4. Conclusion . . . 228

8. TAGED **229**
8.1. Introduction et motivations . . . 229
8.2. Cas d'utilisation . . . 230
8.3. Valorisation . . . 230
8.3.1. Dépôt par l'Agence de protection de programmes (APP) . . . 231
8.4. Méthode TAGED . . . 231
8.4.1. Phase 1 : configuration . . . 234
8.4.2. Phase 2 : collecte . . . 237
8.4.3. Phase 3 : analyse . . . 238
8.4.4. Phase 4 : classement . . . 242
8.4.5. Phase 5 : rendu . . . 243
8.4.6. Plateforme algorithmique TAGED . . . 244
8.5. Applications aux jeux vidéo . . . 244
8.5.1. Applications au game design . . . 244
8.5.2. Applications à l'e-sport . . . 246
8.6. Conclusion . . . 260

9. Conclusion **261**
9.1. Bilan des contributions . . . 261
9.2. Perspectives . . . 262

Bibliographie **264**

Index **286**

A. Annexes **291**
A. Réalisations autour des jeux . . . 292
A.1. Genius . . . 292
A.2. Unplugged . . . 296

Table des figures

2.1. Diagramme de Hasse : ensemble des parties par l'inclusion ensembliste 30
2.2. Jeu vidéo Diablo III : Reaper of Souls 32
2.3. Jeu vidéo Diablo III : Reaper of Souls : un exemple de héros 33
2.4. Jeu vidéo Diablo III : Reaper of Souls : les classes de personnages 34
2.5. Cuboïdes de Diablo3 selon Type, Propriété, Équipement . . . 35
2.6. Cuboïdes détaillés de Diablo3 selon Type, Propriété, Équipement 36
2.7. Graphe d'exécution de Partitioned-Cube pour la relation Diablo3 . . 47
2.8. Arbre de parcours des cuboïdes par BUC pour la relation Diablo3 . . 48
2.9. Arbre de parcours des partitions par BUC pour la relation Diablo3 . . 54
2.10. Diagramme de Hasse du treillis cube de la relation Diablo3 64
2.11. Représentation de l'ensemble des clefs de la relation Diablo3 71
2.12. Diagramme de Hasse du treillis cube fermé de la relation Diablo3 . . 73
2.13. Dominance et frontière de Pareto 76
2.14. Représentation du treillis du Skycube de la relation Diablo3 82

3.1. Représentation du cube de données émergent par les bordures [L; U] . 94
3.2. Représentation du cube de données émergent et de ses bordures $[L;U]$ 96
3.3. Illustration des différentes phases d'E-IDEA 101
3.4. Automate de transitions de phases de l'algorithme E-IDEA 102
3.5. Illustration du cube de données émergent avec ses bordures $[L;U]$ . . . 113
3.6. Illustration du L-cube de données fermé émergent 115

4.1. Classification des joueurs de Bartle 125
4.2. Briques *gameplay* 129
4.3. Glow Tic Tac Toe, exemple de jeu vidéo de tic-tac-toe 131
4.4. La dynamique du *flow* 132
4.5. Représentation du « paysage expérientiel » du *flow* 134
4.6. Boucle OCR 136
4.7. Street Fighter II' : Champion Edition, exemple de jeu vidéo de combat 138

5.1. Jeu vidéo Open Match 3 145
5.2. Briques *gameplay* d'Open Match 3 146
5.3. Schéma en étoile de l'entrepôt OM3 147
5.4. Jeu vidéo Open Match 3, exemple d'explosion 149
5.5. Dimension de D_{Joueur} 151
5.6. Diagramme de Hasse du treillis cube hiérarchique de OM3 165
5.7. Cuboïdes de OM3 suivant D_{Joueur} (IdJ), D_{Tour} (IdT) et $D_{Série}$ (IdS) 167
5.8. Diagramme de Hasse du treillis cube hiérarchique fermé de OM3 172

6.1. Jeu vidéo Pokémon Showdown ! : un exemple de combat 176
6.2. Jeu vidéo Pokémon Showdown ! : un exemple d'équip 177
6.3. Jeu vidéo Pokémon Showdown ! : utilisation du Pokédex 179
6.4. Jeu vidéo Pokémon Showdown ! : l'application de bureau 181
6.5. Briques *gameplay* de Pokémon Showdown ! 182
6.6. Dracaufeu, Grodoudou, Alakazam, Flagadoss, Noadkoko, Leveinard, Staross, Ronflex 182
6.7. Représentation du treillis du Skycube de la relation $\texttt{Pokémon}_1$ 186
6.8. Représentation du treillis du Skycube de la relation $\texttt{Pokémon}_2$ 187
6.9. Diagramme de Hasse du treillis des partitions de $E = \{1,2,3,4\}$ 190
6.10. Diagramme de Hasse du treillis des concepts Accords des novices . . . 193
6.11. Diagramme de Hasse du treillis des concepts Accords des experts . . . 194
6.12. Diagramme de Hasse du treillis des concepts Skylines des novices . . . 196
6.13. Représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_1$ 197
6.14. Diagramme de Hasse du treillis des concepts Skylines des experts . . . 198
6.15. Représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_2$ 199
6.16. Réduction de la taille 202
6.17. Temps de réponse des requêtes 203

7.1. Alakazam, Flagadoss, Noadkoko, Leveinard, Staross, Tauros, Ronflex . 217
7.2. Exemple de graphe de hiérarchie de dominance 221
7.3. Graphe de hiérarchie de dominance de Pokémon Showdown ! 222

8.1. Société d'Accélération du Transfert des Technologies Sud-Est (SATT SE) 231
8.2. Dépôt d'IDEA par l'Agence de protection de programmes (APP) 232
8.3. Dépôt de TAGED par l'Agence de protection de programmes (APP) . . 232
8.4. Principe de fonctionnement de TAGED 233
8.5. Méthode TAGED 235
8.6. Briques *gameplay* de Pokémon Showdown ! : avant équilibrage 246
8.7. Briques *gameplay* de Pokémon Showdown ! : après équilibrage 246
8.8. SpaceWar ! (1962), premier jeu vidéo compétitif 247
8.9. Salle d'arcade en Corée du Sud 248
8.10. Tournoi d'*e-sport* 249
8.11. Système de classement de League of Legends 258

A.1. Genius : maquette 292
A.2. Genius : écran de chargement 293
A.3. Genius : écran de ville zonée en quartiers 293
A.4. Genius : tutoriel 294
A.5. Genius : écran des options 295
A.6. Genius : mission 295
A.7. Genius : mini-jeu 296
A.8. Jeux Unplugged conçus et développés 298

Liste des tableaux

2.1. Relation exemple Diablo3 32
2.2. Représentation en relation du cube de données de Diablo3 39
2.3. Cube de données de $t_6 = (Croisé, Vitesse, Bastion, Enchanteresse)$ 42
2.4. Représentation en relation du cube de données iceberg de Diablo3 . 43
2.5. Représentation en relation du cube intervallaire de Diablo3 44
2.6. Espace multidimensionnel de la relation Diablo3 58
2.7. Bornes du treillis cube contraint pour $300 \leq$ SUM(Quantité) ≤ 700 . . 67
2.8. La relation Diablo3 77

3.1. Relation exemple Diablo3 88
3.2. Relation Diablo3_1, personnages de la Saison 19 89
3.3. Relation exemple Diablo3_2, personnages de la Saison 20 89
3.4. Cube de données émergent de Diablo3_1 vers Diablo3_2 92
3.5. Bordures $[L;U]$ du cube de données émergent 95
3.6. Cube émergent calculé par la requête avec fusion des relations 99
3.7. Relation « fusionnée » Diablo3_{Fus} 100
3.8. Relation fusionnée dédoublonnée Diablo3^{+}_{Fus} 103
3.9. Relation exemple Diablo3 111
3.10. Relation exemple Diablo3_1 111
3.11. Relation exemple Diablo3_2 112
3.12. Cube de données émergent de Diablo3_1 vers Diablo3_2 113
3.13. Ensemble des tuples fermés émergents 114
3.14. Bordure $[L;U]$ du cube émergent de Diablo3_1 vers Diablo3_2 115

4.1. Classification de joueurs de Yee 126
4.2. Liste des métabriques 129

5.1. Relation de la table des faits Fait_{OM3} 149
5.2. Relation de la dimension D_{Tour} 150
5.3. Relation de la dimension D_{Joueur} 150
5.4. Relation de la dimension $\text{D}_{\text{Série}}$ 152
5.5. Espace multidimensionnel de l'entrepôt de données OM3 157
5.6. Représentation en relation du cube de données hiérarchique de OM3 . 170

6.1. La relation exemple Pokémon 183
6.2. La relation exemple Pokémon_1, des joueurs novices 184
6.3. La relation exemple Pokémon_2, des joueurs experts 184
6.4. Relation fusionnée dédoublonnée Pokémon^{+}_{Fus} 185

6.5. Relation fusionnée $\texttt{Pokémon}_{Fus}$ 208
6.6. Tuples émergents mesurés de la relation $\texttt{Pokémon}_{Fus}$ selon Échec . . 210
6.7. Relation fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$ 211
6.8. Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$ 211
6.9. Bordure L du Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$ 212
6.10. Tuples fermés émergents de la relation $\texttt{Pokémon}^{+}_{Fus}$ 213
6.11. L-Skycube fermé émergent 214

7.1. La relation $\texttt{Pokémon}_1$ 216
7.2. La relation $\texttt{Pokémon}_2$ 217
7.3. Calcul de score avec la méthode dp-idp améliorée 223
7.4. Classement de Skyline avec la méthode CoSky 227

8.1. La relation exemple Pokémon 239
8.2. Relation fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$ 240
8.3. Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$ 241
8.4. Classement du Skyline des joueurs novices 242
8.5. Classement du Skyline des joueurs experts 242
8.6. Skycube émergent ordonné de la relation $\texttt{Pokémon}^{+}_{Fus}$ 243

Liste des algorithmes

1. Algorithme Partitioned-Cube 46
2. Algorithme Agréger 49
3. Algorithme ÉcrireSuivants 49
4. Algorithme BUC 50
5. Algorithme Partitionner 51

6. Algorithme Agréger (pour E-IDEA) 104
7. Algorithme E-IDEA 105
8. Algorithme E-IDEA_P_0 106
9. Algorithme F-IDEA (pour la bordure L) 109
10. Algorithme $\mathbb{C}$-IDEA 118

11. Algorithme Sort-Filter-Skyline (SFS) 202

12. Algorithme DeepSky 228

1. Introduction

Sommaire

1.1. Survol des travaux existants 17
1.2. Cadre de la thèse 20
1.3. Synthèse des contributions 20
1.4. Organisation du mémoire 23

1.1. Survol des travaux existants

Notre travail de recherche, bien que s'inscrivant principalement en sciences des données dans un contexte OLAP, est hybride puisqu'il combine fouille de données multidimensionnelles et conception de jeux vidéo.

En terme de survol des travaux existants, ou état de l'art, nous proposons d'étudier et de présenter dans un premier temps l'analyse des bases de données, et plus précisément le calcul et la représentation efficace d'un cube de données ainsi que l'analyse multicritère, suivi dans un deuxième temps de la formalisation du cube de donnée émergent, et enfin, dans une troisième partie, de l'état actuel de la théorie de la conception de jeux vidéo.

Dans le contexte OLAP (*online analytical processing*), il est fondamental de proposer des approches de calcul de cubes de données efficaces (AGARWAL, AGRAWAL, DESHPANDE et al. 1996). Dans cette optique, nombreux sont les algorithmes qui ont été conçus avec pour objectif l'optimisation des temps de calcul nécessaires. Pour ce faire, un calcul d'agrégations pour chaque combinaison de dimensions est effectué. Toutes ces combinaisons constituent l'ensemble des parties de l'ensemble des dimensions qui est, de manière classique, représenté sous forme d'un treillis : le treillis des cuboïdes. Cette structure offre une solution permettant d'organiser au mieux les différentes étapes du calcul et de même, d'exploiter les résultats intermédaires. De manière historique, cela se fait grâce à un parcours de bas en haut de la structure, afin d'obtenir des cuboïdes, non pas à partir de la relation, mais d'agrégations d'un autre cuboïde, soit des cuboïdes les plus détaillés à ceux les plus agrégés. L'algorithme BUC (BEYER et RAMAKRISHNAN 1999), qui a en outre l'avatange d'être totalement intégrable au sein d'un SGBD [1], propose une autre approche : le calcul du cuboïde le plus agrégé est effectué en premier, ce qui permet d'utiliser les résultats intermédiaires

1. Un système de gestion de base de données (SGBD) est le logiciel qui permet d'enregistrer, récupérer, ajouter, supprimer et modifier des données.

en désagrégeant des données, et le cube de données iceberg (BEYER et RAMAKRISHNAN 1999), permettant de retenir que les tendances suffisamment pertinentes, caractérisé par une valeur de la mesure au delà d'un seuil donné, est utilisé. BUC utilise notamment l'algorithme Partitioned-Cube (ROSS et SRIVASTAVA 1997) qui offre une technique efficace pour partitionner, ou fragmenter, des données trop volumineuses pour être chargées en mémoire centrale, les traiter par partition avant de fusionner les résultats. Afin de formaliser une représentation des cubes de données, un espace de recherche, appelé treillis cube (LAKSHMANAN, PEI et HAN 2002 ; CASALI, CICCHETTI et LAKHAL 2003a) a été formalisé. De la sorte, les tuples représentant les tendances sont organisés selon un ordre de généralisation, et une représentation réduite, le cube de données fermé, délestée des redondances, est offerte afin de réduire l'espace de stockage nécessaire. Le cube de données fermé est, à notre connaissance, la plus petite représentation sans perte d'information, du cube de données. Sans constituer un panorama exhaustif de la littérature dans le domaine de l'OLAP, notre survol des travaux existants retrace les recherches effectuées sur les cubes de données selon son calcul, sa représentation formelle et un nouveau type d'analyse. À quoi nous ajoutons une présentation de l'analyse multicritère, se proposant de retourner les meilleurs résultats même lorsqu'il n'en existe pas d'optimaux, et de l'opérateur Skyline (BÖRZSÖNYI, KOSSMANN et STOCKER 2001) introduisant une relation de dominance entre tuples, et permettant de ne retenir que ceux non-dominés par d'autres. La combinaison du concept de cube de données à celui de Skyline a donné naissance au Skycube (PEI, YUAN, LIN et al. 2006), permettant de répondre efficacement à n'importe quelle requête du décideur en recherchant les tuples dominants selon diverses combinaisons de critères et leurs préférences, ainsi que de pouvoir observer le comportement des tuples dominants à travers l'espace multidimensionnel et de mieux appréhender les facteurs de dominance.

Si les approches OLAP présentées permettent de faire une analyse de tendances, celle du Skycube, une analyse de préférences, le cube de données émergent (NEDJAR, CASALI, CICCHETTI et al. 2007) permet l'analyse de renversement de tendances. Comme le cube de donnée iceberg combine données et motif fréquent (AGRAWAL, MANNILA, SRIKANT et al. 1996), et le Skycube associe dominance de Pareto et cube de données, le cube de données émergent est une synthèse entre le cube de données et le concept de motif émergent (DONG et J. LI 2005) introduit en fouille de données binaires. À partir de deux relations de schéma comparable ayant chacune un cube de données, alors l'analyse de renversement de tendances permet de comparer les deux cubes de données afin de détecter les tendances non significatives dans un cube de données mais pertinente dans l'autre, appelées tendances émergentes, et réunies au sein du cube de données émergent. La contrainte d'émergence est la combinaison d'un seuil de valeur de mesure à ne pas dépasser pour le premier cube de données, et d'un autre à dépasser dans le second cube de données. Dans diverses applications, la connaissance des tendances émergentes présente un intérêt, et nous considérons que c'est particulièrement le cas dans le domaine des jeux vidéo. L'importance de la chronologie est fondamentale dans la gestion des entrepôts de données. Les données collectées jusqu'à un certain instant, d'une part, et celles correspondant à leur rafrai-

chissement, d'autre part, sont idéales pour le calcul du cube émergent, qui capture les tendances initialement jugées peu pertinentes mais qui deviennent significatives, ou inversement. De la même manière que le calcul d'un cube de données a un coût important et nécessite un espace de stockage qui l'est également, celui du cube émergent l'est aussi, et plutôt deux fois qu'une. L'intérêt d'une représentation réduite du cube de données, avec notamment le cube de données fermé, se retrouve également naturellement concernant le cube de données émergent. Pour ce faire, les bordures L (*Lower*) et U (*Upper*), correspondant aux tuples minimaux et maximaux satisfaisant la contrainte d'émergence, utilisées traditionnellement en fouille de données (RAEDT et KRAMER 2001) ont été transposées. Malheureusement, les bordures offrent une représentation, très réduite, mais avec perte d'information, les valeurs de l'attribut mesure ne pouvant pas être retrouvées. Une famille de représentations sans perte basé sur le concept de fermeture cubique (CASALI, CICCHETTI et LAKHAL 2003b), et constitués de tuples fermés émergents, a été proposé. De la sorte, un ensemble de tuples véhiculant la même sémantique mais à différents niveaux de granularité peut être réduit à un seul représentant, contenant toute l'information nécessaire pour dériver les autres. Néanmoins, l'ensemble des tuples fermés émergents ne constitue pas, à lui-seul, une représentation sans perte, bien que cela peut être obtenu avec une bordure, et c'est pour cela que le L-cube de données fermé émergent (NEDJAR, CASALI, CICCHETTI et al. 2010b) a été défini. Bien qu'utile pour des volumes de données de taille raisonnable, les requêtes SQL de cube de données émergent ont surtout un objectif pédagogique, et la plateforme IDEA est une solution algorithmique offrant des méthodes de calcul du cube de données émergent et ses représentations. Basés sur BUC avec une relation fusionnée et déboulonnées des deux cubes de données, les algorithmes E-IDEA (*emergent cube integrable database algorithm*), F-IDEA et $\mathbb{C}$-IDEA calculent respectivement le cube de données entier, le cube de données qui retourne L et U, et le cube de données fermé émergent sans perte.

Pour terminer cet état de l'art, nous faisons une incursion hors du stricte domaine des sciences de données et de leur fouille en présentant la conception théorique de jeux vidéo. La conception de jeux vidéo est un processus complexe du fait que les concepteurs de jeux vidéo ne forment pas directement l'expérience de jeu expérimentée par les joueurs, mais mettent en place des mécaniques, dynamiques et esthétiques (SALEN et ZIMMERMAN 2004) qui vont l'engendrer. Des méthodes pour s'assurer que les intentions des concepteurs de jeu soient du mieux possible transposées en produit ludique ont été proposées. Cependant, il n'existe pas, à notre connaissance, de théorie unifiée pour la conception de jeux vidéo (SCHELL 2014), et la mise en œuvre de l'expérience de jeu de manière plaisante et motivante est complexe. Ainsi, la modification d'un élément de conception peut avoir, des conséquences involontaires sur le dispositif ludique et entrainer les répercutions sur le comportement des joueurs. Après essayer de tout d'abord circonscrire les concepts mêmes de jeu et jeux vidéo, et décrire la structure MDA (*mechanics, dynamics and aesthetics*) comme approche formelle (HUNICKE, LEBLANC et ZUBEK 2004) de trois composants fondamentaux distincts bien que liés de manière causale d'un dispositif ludique (LEBLANC 2004), nous présentons des méthodes de classification non exhaustive, mais représentative,

de modèles de joueurs (BARTLE 1996 ; YEE 2006) et de jeux vidéo (CAILLOIS 1958 ; ALVAREZ 2007 ; FULLERTON 2008 ; ALVAREZ 2018). Bien que la plupart des méthodes de conception de jeux vidéo n'aient jamais été évaluées scientifiquement (NEIL 2012), nous tentons de décrire un consensus partagés par nombre d'auteurs sur le processus, itératif, de méthodologie de conception de *gameplay* (GUARDIOLA et NATKIN 2005) durant toute la durée de création d'un jeu vidéo (ALBINET 2015). Pour cela, nous présentons des constituants fondamentaux de ce processus que sont le *fun* (KOSTER 2014), le *flow* (CSIKSZENTMIHALYI 2009) et son paysage expérientiel, la boucle OCR (*objective, challenge, reward*), les différents types de motivations (RYAN et DECI 2000) et la SDT (*self-determination theory*). L'équilibrage est un domaine primordial en conception de jeux vidéo, et il est au centre de nos expérimentations dans ce travail de thèse. Qu'il soit peu ou prou empirique, avec des *playtests*, qui sont des études qualitatives relativement couteuses où le comportement des joueurs sont observés durant leurs parties, ou mathématiques avec l'usage de la théorie des jeux et l'équilibre de Nash notamment (NASH 1950 ; NEUMANN et MORGENSTERN 1972), il correspond à la recherche et au réglage de dominances stratégiques involontaires ou contrôlées afin de créer une expérience de jeu optimale et qu'un comportement de jeu ne puisse pas être perçu significativement injustement par certains joueurs.

1.2. Cadre de la thèse

Ce document présente les travaux de thèse effectués en collaboration avec la SATT SE (Société d'Accélération du Transfert des Technologies Sud-Est) au sein d'équipe de recherche DANA (*data mining at scale*) du LIS (Laboratoire d'informatique & systèmes), UMR 7020.

1.3. Synthèse des contributions

Ce travail de thèse a pour objectif principal de proposer des solutions qui aide à l'amélioration de l'équilibrage, et de sa maitrise, dans la conception de jeux vidéo. Côté sciences des données, nos travaux de recherche se situent dans ce contexte général de l'OLAP, et nos propositions s'inscrivent dans le cadre de l'espace de recherche treillis cube. L'association entre cube de données et jeux vidéo n'a, à notre connaissance, fait à ce jour l'objet d'aucun travail de recherche international ou national.

De nombreuses approches ont proposé de pré-calculer des cubes de données afin de répondre efficacement aux requêtes OLAP dans les entrepôts de données. Cependant, peu ont proposé des solutions intégrant l'ensemble des résultats possibles, et c'est cette idée qui nous a encouragé à intégrer dans ces réponses les dimensions hiérarchiques (HURTADO et A. O. MENDELZON 2002). Pour répondre à ce besoin, nous proposons, dans cette thèse, une redéfinition complète du cadre et de la définition formelle de l'analyse des bases de données traditionnelles (KUIJPERS et VAISMAN 2016) par le prisme des dimensions hiérarchiques. Après avoir caractérisé le treillis cube de données hiérarchique, nous introduisons le cube de données hiérarchique

et sa représentation réduite la plus concise, le cube de données hiérarchique fermé. Comme pour le cube de données traditionnel, et même le cube de données émergent, il offre une reproduction compacte de manière à optimiser l'espace de stockage en supprimant les redondances des données, notamment celles fortement corrélées. De telles données sont typiques des entrepôts de données, et en particulier dans le jeu vidéo, notre domaine d'études et d'expérimentation, où les attributs dimensions hiérarchiques sont largement représentés. Par ailleurs, les auteurs de la littérature reconnaissent traditionnellement deux classes d'opérateurs OLAP différentes. La première est dédiée à la structure des données et en permet sa manipulation afin d'en extraire les connaissances recherchées. La seconde est orientée granularité des données et dispose d'opérateurs qui synthétisent et agrègent les informations dans un sens (*roll up*), et qui les précisent ou les décomposent dans l'autre (*drill down*) (FAVRE, BENTAYEB, BOUSSAID et al. 2013). Les hiérarchies dimensionnelles définissent les chemins d'accès dans les données et permettent la mise en œuvre de cette seconde classe d'opérateurs OLAP. Cela constitue une autre raison de considérer les dimensions hiérarchiques alors même que le volume de données calculé, encore plus conséquent que dans le cas du cube de données traditionnel, doit être transposé par une représentation particulièrement réduite pour être utilisable de manière efficace.

Nous introduisons le concept de Skycube émergent en combinant ceux de Skycube et de cube de données émergent. Comme pour le cube de données émergent, il n'existe à notre connaissance aucune solution intégrée aux SGBD permettant de calculer le Skycube émergent, et de pouvoir ainsi tirer partie des outils d'analyse ROLAP, mais l'ensemble des algorithmes de la plateforme IDEA, E-IDEA, F-IDEA et ℂ-IDEA (MARTIN NEVOT, NEDJAR, LAKHAL et CICCHETTI 2019) peut être utilisé à cette fin. Plus le décideur recherche des fonctionnalités, plus le volume d'informations à préserver est important. Notre proposition permet d'obtenir des données optimales, ou non-dominées, selon plusieurs critères, et d'observer leur évolution selon une propriété définie. Jusqu'à présent, un cube de données émergent ne comportait qu'une seule mesure. Nous proposons d'en utiliser dorénavant plusieurs pour répondre au mieux aux besoins du décideur, et pour correspondre à l'exigence de l'analyse décisionnelle multicritère qui est inutile lorsqu'employé avec un seul attribut. Un cube de données est couteux à calculer. Un cube de données émergent l'est peu ou prou deux fois plus. En revanche, un Skycube émergent l'est moins car le calcul de la recherche de renversement de tendances est effectuée après le calcul de deux Skycubes, qui baisse sensiblement le nombre de tuples par rapport à la relation initiale. Pour autant, il est malgré tout possible de gagner en temps de calcul ainsi qu'en espace de stockage, et à cette fin, nous proposons deux représentations réduites s'appliquant successivement. La première porte sur la matérialisation partielle sans perte d'information du Skycube en utilisant le treillis des Concepts Skylines, lui-même basé sur le treillis des Concepts Accords d'une relation et le treillis des partitions. La seconde, qui suit la première, est une famille de réduction concernant le Skycube émergent à proprement parlé, et il peut s'agir du Skycube fermé émergent pour une réduction avec perte d'information, ou du L-Skycube fermé émergent pour une réduction moins importante mais sans perte d'information.

La recherche multicritère dans les bases de données permettant d'identifier efficacement les tuples optimaux par rapport à un ensemble d'attributs et leurs préférences a été activement étudiée, essentiellement avec l'utilisation de l'opérateur Skyline. Pourtant, peu d'approches ont été proposées pour un classement pertinent des points Pareto-optimal, ou points du Skyline, permettant d'ordonner avantageusement les résultats à forte cardinalité, et d'orienter le décideur. Dans le cas d'une cardinalité élevée, la significativité de l'information extraite diminue fortement en principe. En effet, plus les points d'un Skyline sont nombreux, au plus ils ont des intérêts proches. L'approche dp-idp (*dominance power and inverse dominance power*), inspiré du schéma de pondération tf-idf (*term frequency-inverse document frequency*) utilisé en recherche d'information, et qui utilise la relation de dominance de Pareto, est certainement une des approches les plus récentes et permet d'attribuer un score à chaque point du Skyline (VALKANAS, PAPADOPOULOS et GUNOPULOS 2014). Nous proposons d'améliorer la méthode dp-idp, et son calcul des scores, en introduisant pour cela le concept de hiérarchie de dominance. La méthode dp-idp n'offre pas toujours la possibilité de distinguer deux points du Skyline, aussi pour palier à cela, nous introduisons la méthode Cosky (cosine Skyline), une solution de type TOPSIS (*technique for order preference by similarity to ideal solution*), issue à la fois des domaines de la recherche d'information et de l'analyse multicritère dans les base de données, qui ne fait pas appel à la dominance de Pareto (BEHZADIAN, KHANMOHAMMADI OTAGHSARA, YAZDANI et al. 2012). Cosky, intégrable directement dans un SGBD, effectue une pondération automatique d'attributs normalisés grâce à l'indice de Gini, suivi d'un calcul de score avec le cosinus de Salton par rapport à un point idéal déterminé. En couplant le principe de Skyline multiniveaux (PREISINGER et ENDRES 2015) à la méthode Cosky, nous caractérisons l'algorithme DeepSky.

Il n'existe pas dans la littérature, à notre connaissance, de solution automatisée formalisée satisfaisante pour assister le concepteur de jeu vidéo dans l'équilibrage de jeu vidéo. C'est ce qui a motivé notre volonté de proposer la méthode TAGED (*tool assisted game equilibrium design*) et qu'elle soit adaptable à des jeux vidéo solo ou multijoueur à fort volume de données, quelque soit sa classification (ALVAREZ 2018). Les jeux vidéo actuels, et à plus forte raison ceux avec une importante connectivité, génèrent de gros volumes de données intrinsèques au dispositif ludique, ou données inhérentes aux mécaniques du jeu vidéo, mais également extrinsèques, ou données extraites des *verbatims* des comportements de joueurs, déviants ou atypiques, et des informations de parties. Leur analyse permet de mieux comprendre le comportement et les motivations des joueurs, mais aussi de connaître les modifications à apporter au dispositif ludique afin d'améliorer leur satisfaction et leur temps de jeu. Au plus des données de jeux vidéo sont longitudinales, au plus elles sont riches. En se basant sur cette idée, nous proposons de mettre en application le Skycube émergent ordonné pour la trajectoire temporelle qu'il décrit par l'intermédiaire de son analyse de renversement de tendances. Afin d'extraire les connaissances contenues dans le Skycube émergent ordonné, nous combinons le Skycube émergent, associant lui-même l'analyse multicritère et la recherche de renversement de tendances avec les concepts de Skycube et de cube de données émergent, avec la classification par l'algorithme

DeepSky et la méthode Cosky couplé au principe de Skyline multiniveaux. Comme pensé par ses auteurs, nous utilisons le concept de cube de données émergent ainsi que la plateforme algorithmique IDEA comme un outil d'aide à l'administration d'un entrepôt de données, et nous le prolongeons avec la solution TAGED, qui a obtenue une déclaration d'invention ainsi qu'une pré-maturation auprès de la SATT SE. La méthode est constituée de cinq phases, configuration, collecte, analyse, classement et rendu. La phase initiale, celle de configuration, est essentiellement caractérisée par une personnalisation des paramètres d'exécution de la méthode. Le nettoyage des données (*data cleaning*) est une des problématiques principales des entrepôts de données (RAHM et DO 2000). La phase suivante s'inscrit dans cette problématique avec la récupération, l'extraction, le nettoyage et la transformation des données brutes d'entrée. Elle est suivie de la phase principale, celle d'analyse, qui a en charge le calcul du Skycube émergent. La phase de classement par l'algorithme DeepSky des tuples optimaux vers les moins pertinents. La dernière phase présente les connaissances obtenues par la méthode. Nous terminons nos contributions en décrivant les applications possibles au *game design* et, après avoir fait une tentative de caractérisation du statut de l'*e-sport*, nos définissons ses enjeux, juridiques (RABU, REVERCHON-BILLOT, MARTIN NEVOT et al. 2017), arbitraux (ROUSTANT et BOUTET 2003) et classificatoires (ELO 2008). Nous ouvrons la discussion sur la potentialité de considérer l'*e-sport* comme un outil éducatif, mais aussi comme un domaine de recherche (HEERE 2017 ; REITMAN, ANDERSON-COTO, WU et al. 2019).

1.4. Organisation du mémoire

Ce mémoire de thèse est organisé en deux parties et s'achève par une conclusion générale.

La première partie est composée de trois chapitres consacrés à l'état de l'art.

Le premier chapitre présente un ensemble de travaux relatifs à l'analyse des bases de données, et plus particulièrement à la fouille de données multidimensionnelles. Y sont décrites, après un rappel des concepts de base, les méthodes de calcul de cubes de données, avec une attention particulière pour l'algorithme BUC, leurs approches de représentation réduites, et notamment le cube de données fermé, ainsi que l'analyse multidimensionnelle des préférences offerte par l'opérateur Skyline et le Skycube.

Le deuxième chapitre introduit l'ensemble des concepts nécessaire à la définition du cube de données émergent et à sa représentation par bordures. Il contient également les approches permettant d'obtenir le cube de données émergent et ses bordures, autrement dit les requêtes SQL correspondantes ainsi que les algorithmes E-IDEA, F-IDEA et $\mathbb{C}$-IDEA.

Le troisième chapitre propose la théorisation actuellement établie de la conception de jeux vidéo par son approche formelle, ses classifications de modèles de joueurs et de jeux vidéo, son *gameplay* ainsi que ses techniques d'équilibrage.

Les domaines exposés ne sont, bien sûr, pas présentés de manière exhaustive et seules les approches ayant un lien intéressant avec nos travaux sont décrites.

La seconde partie est composée de quatre chapitres consacrés à nos contributions.

Le quatrième chapitre donne une caractérisation des dimensions hiérarchiques et des cubes données hiérarchiques, traditionnel et fermé.

Le cinquième chapitre introduit le Skycube émergent. Sont présentés, en lien avec l'analyse multicritère et la dominance de Pareto, le treillis des partitions, celui des Concepts Accords et celui des Concepts Skylines, employés pour la matérialisation partielle du Skycube. Concernant la recherche de renversement de tendances, c'est la transposition des approches permettant d'obtenir le Skycube émergent depuis le cube de données émergent, et le Skycube fermé émergent depuis le cube de données fermé, qui est proposée.

Le sixième chapitre explore le classement de Skyline, avec l'amélioration de la méthode dp-idp et la présentation de la méthode CoSky et de l'algorithme DeepSky basé conjointement sur la solution CoSky et sur le principe de Skyline multiniveaux.

Le septième chapitre décrit la méthode TAGED reprenant les approches des deux chapitres précédents, et des trois chapitres de l'état de l'art, afin d'offrir la possibilité d'extraire des connaissance de Skycubes émergents ordonnés, et de les mettre en application avec les jeux vidéo.

Le dernier chapitre de ce mémoire établit en conclusion un bilan des contributions apportées avant d'ouvrir différentes perspectives de recherche à venir.

En annexe sont mis à disposition la présentation les dispositifs ludiques que nous avons conçu et développés par ailleurs et qui auraient typiquement pu tirer, ou qui tireront certainement partie, de la solution TAGED.

Première partie

État de l’art

2. Analyse des bases de données

Sommaire

2.1. Introduction . . . 27
2.2. Cube de données . . . 27
2.2.1. Introduction . . . 27
2.2.2. Rappels théoriques . . . 29
2.2.3. Cas d'utilisation . . . 31
2.2.3.1. Jeu vidéo Diablo III : Reaper of Souls . . . 31
2.2.3.2. Relation exemple . . . 31
2.2.4. Définition formelle du cube de données . . . 34
2.2.4.1. Catégorie de fonctions agrégatives . . . 40
2.2.4.2. Variantes du cube de données . . . 42
2.2.5. Calcul du cube de données . . . 44
2.2.5.1. L'algorithme Partitioned-Cube . . . 44
2.2.5.2. L'algorithme BUC . . . 46
2.2.6. Synthèse . . . 53
2.3. Cube de données fermé . . . 55
2.3.1. Introduction . . . 55
2.3.2. Cadre formel du cube de données fermé . . . 56
2.3.2.1. Treillis cube . . . 56
2.3.2.2. Treillis cube convexe . . . 65
2.3.2.3. Treillis cube et treillis des parties . . . 68
2.3.3. Définition formelle du cube de données fermé . . . 69
2.3.4. Synthèse . . . 72
2.4. Skyline et Skycube . . . 74
2.4.1. Introduction . . . 74
2.4.2. L'opérateur Skyline . . . 75
2.4.2.1. Problèmes d'optimisation multicritère . . . 75
2.4.2.2. Cadre d'application . . . 76
2.4.2.3. Opérateur Skyline en SQL . . . 78
2.4.2.4. Préférence Skyline . . . 78
2.4.2.5. Définition formelle de Skyline . . . 78
2.4.2.6. Calcul de Skyline . . . 79
2.4.3. Skycube . . . 81
2.4.3.1. Cadre formel du Skycube . . . 81
2.4.3.2. Problèmes associés aux Skylines . . . 82

2.4.3.3. Calcul de Skycube 83
2.4.4. Synthèse 84
2.5. Conclusion 84

2.1. Introduction

Les données brutes sont rarement d'une utilité directe. Leur véritable valeur repose sur la capacité à extraire des informations utiles pour l'aide à la décision ou l'exploration et la compréhension des phénomènes régissant les sources de données. Traditionnellement, l'analyse était un processus strictement manuel. Un ou plusieurs analystes se familiarisaient avec les données et, à l'aide de techniques statistiques, produisaient des résumés et des rapports. En fait, les analystes agissaient comme des exécutants de requêtes sophistiquées. Cependant, une telle approche devient obsolète rapidement lorsque la quantité de données augmente et que le nombre de dimensions s'accroît. La quantité de données augmente si rapidement que l'analyse manuelle (même si elle est possible) ne peut tout simplement pas suivre le rythme (FAYYAD et UTHURUSAMY 1996).

Le domaine de découvertes de connaissances à partir de données (*knowledge discovery in databases* ou KDD), du point de vue des bases de données, met l'accent sur les concepts et techniques de l'exploration, ou fouille, de données (*data mining*) afin de découvrir des modèles de données intéressants « cachés » dans de grands ensembles de données. L'exploration de données a suscité une grande attention dans l'industrie de l'information et dans la société en général ces dernières années, en raison de la grande disponibilité d'énormes quantités de données et du besoin imminent de transformer ces données en informations et connaissances utiles. La fouille de données fait partie de l'évolution « naturelle » de la technologie des bases de données (HAN, KAMBER et PEI 2011).

Ce type d'application sont couramment utilisées en informatique décisionnelle (ou BI pour *business intelligence*, ou encore DSS pour *decision support system*), comme outil de pilotage des décideurs et des directions, offrant la possibilité d'avoir une vue transversale de l'activité d'une entreprise.

Dans ce chapitre, nous présentons succinctement le cadre de l'analyse des bases de données traditionnelle. Nous pouvons trouver une présentation détaillée de nombre de concepts développés ici dans NEDJAR 2009.

2.2. Cube de données

2.2.1. Introduction

Les entrepôts de données, ou *data warehouses* (INMON 1996), permettent l'enregistrement d'énormes volumes de données accumulées au fil du temps dans les bases opérationnelles. En effet, récemment l'évolution des technologies a conduit les entreprises à conserver leurs données et ainsi préserver la « mémoire » de leurs activités.

Les entrepôts de données ont été conçus dans cet objectif. Contrairement aux bases opérationnelles, ils ont des caractéristiques notables. Tout d'abord, ils ne sont pas destinés à la gestion quotidienne du système d'information mais à offrir une véritable aide à la prise de décision. Leurs utilisateurs, des décideurs, sont donc peu nombreux et s'intéressent non pas au détail des données mais à des tendances générales, selon tel ou tel critère, non explicitement enregistrées. Les données des entrepôts sont également particulières. Régulièrement insérées dans l'entrepôt lors de rafraîchissements, ces données sont pérennes dans le sens où elles ne font l'objet d'aucune mise à jour. De plus, à leur insertion, ces données sont pourvues d'une estampille temporelle. On dit qu'elles sont historisées. À partir des gisements de données ainsi constitués, il était naturel de chercher à les exploiter au mieux. Là encore, il ne s'agit pas de formuler des requêtes classiques, simples et fréquentes, sélectionnant généralement quelques dizaines de tuples [1], mais de procéder à des analyses nécessitant d'agréger les données afin de dégager des grandes tendances. De telles requêtes sont particulièrement coûteuses car elles demandent des balayages d'importants volumes de données et qu'elles sont par nature même complexes. Cependant, s'inscrivant dans un processus d'aide à la décision (ou OLAP [2]), la formulation de ces requêtes dépend à la fois des connaissances antérieures et des besoins des décideurs (Chaudhuri et Dayal 1997 ; Chaudhuri, Dayal et Narasayya 2011). OLAP s'oppose au traitement de transaction en ligne (OLTP pour *online transaction processing*) Elle a donc lieu de manière *ad hoc* et idéalement devrait être interactive. Or le calcul sous-jacent est complexe et long, d'où l'idée de pré-calculer les résultats. Cependant, ne pouvant prévoir les souhaits de l'utilisateur, le pré-calcul doit concerner l'ensemble de tous les résultats possibles. Le concept de cube de données, ou hypercube OLAP, a été introduit pour répondre à cette problématique (Gray, Chaudhuri, Bosworth et al. 1997). C'est le concept central pour l'analyse OLAP. Le cube de données est constitué de l'union des résultats des requêtes agrégatives de partitionnement (utilisant l'opérateur `GROUP BY`) sur

1. Collection ordonnée des valeurs d'un nombre indéfini d'attributs relatifs à un même objet.
2. Ce terme a été défini par Codd 1990 au travers de douze règles à appliquer à une base de données pour qu'elle soit OLAP :
   1. Vue conceptuelle multidimensionnelle ;
   2. Transparence ;
   3. Accessibilité ;
   4. Constance des temps de réponses ;
   5. Architecture client-serveur ;
   6. Indépendance des dimensions ;
   7. Gestion des matrices creuses ;
   8. Accès multi-utilisateurs ;
   9. Pas de restrictions sur les opérations inter et intra dimensions
   10. Manipulation aisée des données
   11. Simplicité des rapports
   12. Nombre illimité de dimensions et nombre illimité d'éléments sur les dimensions.

toutes les combinaisons possibles des critères d'analyse (attributs catégories). Grâce au pré-calcul du cube, l'utilisateur peut avoir une réponse quasi instantanée à toutes les requêtes qui lui seront utiles.

La matérialisation complète du cube de données permet d'avoir des temps de réponse rapides pour l'utilisateur mais en contrepartie nécessite un coût énorme autant en temps de calcul qu'en espace de stockage. Plusieurs approches ont tenté de trouver le meilleur compromis entre temps de réponse aux requêtes utilisateur et besoin en ressources pour le calcul et l'enregistrement du cube de données (GRAY, CHAUDHURI, BOSWORTH et al. 1997 ; AGARWAL, AGRAWAL, DESHPANDE et al. 1996 ; ROSS et SRIVASTAVA 1997 ; BEYER et RAMAKRISHNAN 1999 ; HARINARAYAN, RAJARAMAN et ULLMAN 1996 ; SHUKLA, DESHPANDE et NAUGHTON 1998 ; KOTSIS et MCGREGOR 2000). Elles se séparent en deux familles distinctes. La première regroupe les méthodes qui ne matérialisent qu'une partie du cube sacrifiant ainsi le temps de réponse utilisateur au profit d'un coût d'enregistrement moindre. Dans la seconde, les approches refusent de sacrifier le temps de réponse utilisateur tout en optimisant autant que possible le calcul et l'enregistrement du cube de données complet.

Dans cette section, nous introduisons les concepts associés au cube de données. Puis nous nous intéressons à certaines méthodes de calcul et matérialisation de ces cubes, celles nous intéressant tout particulièrement pour étayer notre propos, en décrivant les approches les plus intéressantes matérialisant partiellement ou totalement le cube.

2.2.2. Rappels théoriques

Toutes les notions présentées dans cette section sont fondées sur des concepts mathématiques de la théorie des ordres, basés sur la théorie ensembliste, l'arithmetique et la théorie relationnelle (binaire), que nous rappelons ci-après.

Les contributeurs à la géométrie ordonnée ont été répertoriés dans COXETER 1989 : C'est PASCH 1882, qui a été le premier à souligner qu'une telle géométrie pouvait être développée sans référence à une mesure. Son système d'axiomes a été progressivement amélioré par PEANO 1889, HILBERT et VOLKERT 1899 et VEBLEN 1904. Les concepts, résultats et usages des ensembles ordonnés finis ont notamment été approfondis par CASPARD, LECLERC et MONJARDET 2007 et GANTER et WILLE 1999.

Définition 2.1 (Relation d'ordre / d'ordre partiel) - Une relation d'ordre ou ordre partiel est une relation binaire $\leq$ s'appliquant sur les éléments d'un ensemble P et qui vérifie les trois propriétés suivantes. Pour tout $x, y, z \in P$,

1. Réflexivité : $x \leq x$;
2. Anti-symétrie : $x \leq y$ et $y \leq x \Rightarrow x = y$;
3. Transitivité : $x \leq y$ et $y \leq z \Rightarrow x \leq z$.

L'ensemble P muni d'une telle relation binaire d'ordre $\leq$ se note $(P, \leq)$ et est appelé « ensemble ordonné ».

Définition 2.2 (Relation de couverture) - Soit $(P, \leq)$ un ensemble ordonné et $a, b \in P$ deux éléments appartenant à l'ensemble. La relation de couverture associée à P pour deux de ses éléments a et b est dénotée par $a \prec b$ et signifie que b est immédiatement supérieur à a selon l'ordre $\leq$. Plus formellement, $a \prec b$ si et seulement si $a \leq b$ et $\nexists c \in P,\ a < c \leq b$.

Propriété 2.1 (Diagramme de Hasse) - Un diagramme de Hasse (du nom de Helmut Hasse, qui n'était pourtant pas le premier à l'utiliser, un exemple antérieur à celui de Hasse peut être trouvé dans VOGT 1895) est une représentation graphique d'un ordre, en particulier un ordre fini. On associe aux sommets un élément (ordonné) de l'ensemble et aux arêtes une relation entre les éléments, puis on ôte les arêtes qu'on peut obtenir par réflexivité et par transitivité de telle manière que :

— si $a \prec b$, le sommet a est en dessous de b, ainsi les arêtes n'ont pas besoin d'être fléchées pour qu'on puisse en déduire leur orientation ;
— un diagramme de Hasse n'est pas obligatoirement unique.

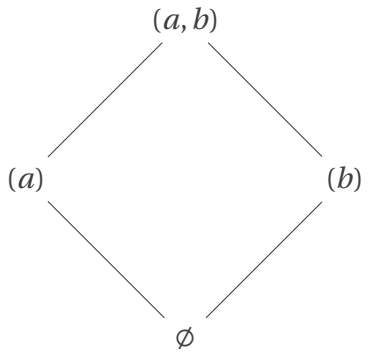


FIGURE 2.1. – Diagramme de Hasse : ensemble des parties par l'inclusion ensembliste

Définition 2.3 (Bordures U/L) - Soit $S \subseteq P$ et $u \in P$. u est un majorant (*upper bound*) de S s'il est au moins aussi grand que chaque élément de S, c'est-à-dire si $\forall s \in S,\ s \leq u$. L'ensemble $U(S)$ dénote l'ensemble des majorants de S. De manière duale, on définit l'ensemble des minorants (*lower bounds*) $L(S)$ de S :

$$U(S) = u \in P | \forall s \in S,\ s \leq u$$
$$L(S) = l \in P | \forall s \in S,\ l \leq s$$

Définition 2.4 (Join / Meet) - Pour un ensemble $S \in P$, l'élément $Join(S)$ (*extremum* de S), s'il existe, est l'unique plus petit majorant de l'ensemble des majorants des éléments de S, $UB(S)$. De même, l'élément $Meet(S)$ (*infimum* de S), s'il existe, est l'unique plus grand minorant de l'ensemble des minorants des éléments de S, $LB(S)$.

$$Join(S) = Min(UB(S))$$
$$Meet(S) = Max(LB(S))$$

Définition 2.5 (Treillis) - Un ensemble ordonné non vide $(T, \leq)$ est un treillis si $\forall x, y \in T$, l'ensemble x, y possède un plus petit majorant $Join(x, y)$ ainsi qu'un plus grand minorant $Meet(x, y)$. Ce même ensemble est un treillis complet si $\forall S \subseteq T$, $Meet(S)$ et $Join(S)$ existent.

Définition 2.6 (Opérateur de fermeture) - Soit P un ensemble fini et $\leq$ une relation d'ordre formant l'ensemble ordonné $(P, \leq)$. Une application h de $(P, \leq)$ est une fermeture si elle vérifie les propriétés suivantes $\forall x, y \in P$:

1. Monotone croissante : $x \leq y \Rightarrow h(x) \leq h(y)$;
2. Extensive : $x \leq h(x)$;
3. Idempotente : $h(h(x)) = h(x)$.

Pour un ensemble ordonné $(P, \leq)$, $e \in P$ est appelé fermé si son image par fermeture est l'élément e lui-même, c'est à dire $h(e) = e$. L'ensemble des fermés d'un treillis T doté d'une fermeture dans T peut être doté d'une structure de treillis nommée treillis des fermés.

2.2.3. Cas d'utilisation

2.2.3.1. Jeu vidéo Diablo III : Reaper of Souls

Pour ce chapitre, considérons la relation `Diablo3` (cf. tableau 2.1) du jeu vidéo d'action et de rôle de type *Hack 'n' slash*[3] Diablo III, et de son extension Diablo III : Reaper of Souls, développé par Blizzard Entertainment et publié sur pour les systèmes d'exploitation Microsoft Windows et Mac OS X. Il s'agit d'un jeu vidéo célèbre (avec plus de 30 millions d'exemplaires vendus, il est dans les dix jeux les plus vendus de tous les temps, et il possède également à son actif plus de 65 millions de joueurs) ayant reçu des critiques très positives (pour des notes globales de 88 % compilées par le site Web Metacritic et 87,64 % par le site Web GameRankings).

Initialement, nous avons porter notre préférence sur ce type de jeu vidéo pour sa grande popularité, aidant à rendre l'exemple compréhensible du plus grand nombre et son application pertinente.

2.2.3.2. Relation exemple

La relation exemple (cf. tableau 2.1) caractérise un personnage, ou héros, par son `Type (T)`, ou classe de personnage (cf. figure 2.4), sa `Propriété (P)` principale à prioriser, ou bonus d'équipement (cf. figure 2.3), son `Équipement (É)` à privilégier (cf. figure 2.3) et son `Compagnon (C)` (cf. figure 2.2). Lors d'une analyse faite par un décideur, il peut consulter la base de données pour connaitre la `Quantité (Q)` de héros concernés.

3. Un jeu vidéo Hack 'n' slash (abréviation de *hack and slash*) qualifie un jeu qui se concentre sur le combat afin de gagner des points d'expérience, servant à améliorer les caractéristiques des personnages incarnés, et de l'équipement de plus en plus performant. L'aspect stratégique de ce type de jeu vidéo réside dans le fait d'optimiser le développement (ou *build*) des personnages.

TABLEAU 2.1. – Relation exemple `Diablo3`

RowId	Type	Propriété	Équipement	Compagnon	Quantité [a]
1	*Féticheur*	*Vitesse* [b]	*Bastion* [c]	*Enchanteresse*	400
2	*Féticheur*	*Vitesse*	*Marche* [d]	*Templier*	100
3	*Sorcière*	*Chance* [e]	*Marche*	*Enchanteresse*	100
4	*Sorcière*	*Vitesse*	*Bastion*	*Templier*	300
5	*Croisé*	*Chance*	*Bastion*	*Enchanteresse*	200

a. Nombre arrondi de personnages dans le classement pour l'Europe.
b. Vitesse d'attaque.
c. Le Bastion de la volonté.
d. La Marche sans fin.
e. Chance de coup critique.

FIGURE 2.2. – Jeu vidéo Diablo III : Reaper of Souls [a]

a. Exemple de *gameplay* (cf. section 4.5) du jeu vidéo Diablo III : Reaper of Souls avec comme `Type` *Féticheur* (en haut à gauche), `Propriété` *Chance*, `Équipement` *Marche* et `Compagnon` *Enchanteresse* (en haut à gauche, à droite du héros).

FIGURE 2.3. – Jeu vidéo Diablo III : Reaper of Souls : un exemple de héros [a]

a. Exemple de présentation de héros du jeu vidéo Diablo III : Reaper of Souls avec comme `Type` *Féticheur* (illustration en haut à gauche), `Propriété` *Chance* (en bas à gauche et détails à droite) et `Équipement` *Marche* (à droite).

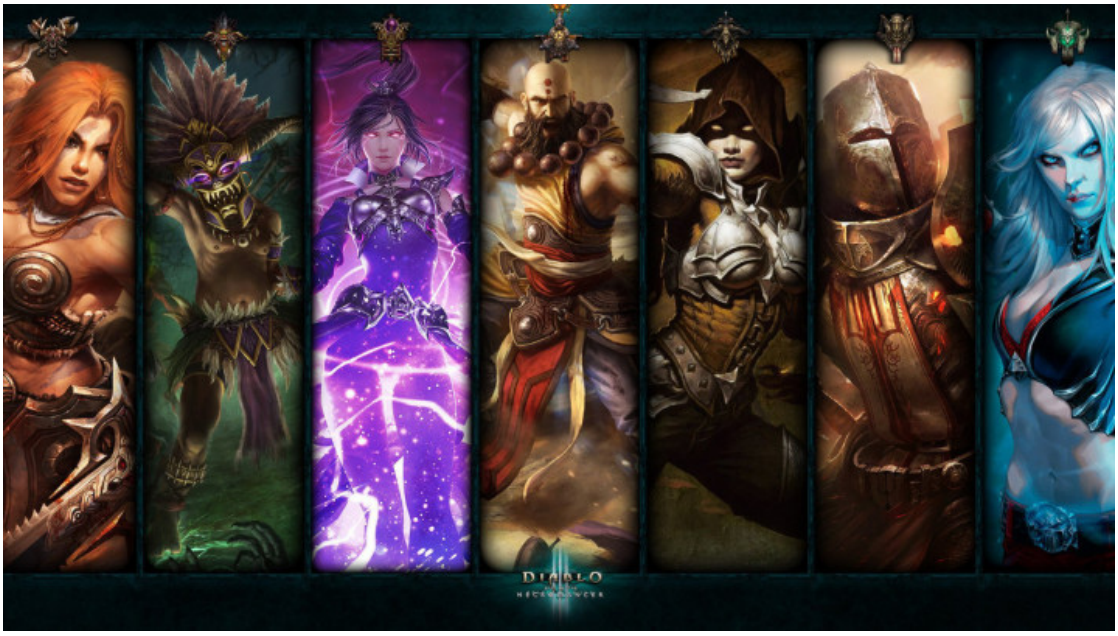


FIGURE 2.4. – Jeu vidéo Diablo III : Reaper of Souls : les classes de personnages [a]

a. `Type` de héros, de gauche à droite : *Barbare*, *Féticheur*, *Sorcière*, *Chasseuse de démons*, *Croisé* et *Nécromancienne*.

2.2.4. Définition formelle du cube de données

Dans les entrepôts, les données sont dites multidimensionnelles car l'utilisateur voit et manipule ces données suivant différents critères, aussi nommés « attributs dimensions ». Durant le processus d'analyse, le décideur a besoin d'appréhender ses données suivant plusieurs combinaisons de ces dimensions. Pour ce faire, il va devoir effectuer des requêtes agrégeant les données par rapport à l'ensemble des critères choisis. Typiquement, ces requêtes mettent aussi en jeu des fonctions agrégatives (ou statistiques) appliquées sur des « attributs mesures » selon les diverses dimensions retenues. Ces différentes interrogations sont généralement calculées par le SGBD en utilisant l'opérateur `GROUP BY` de SQL. Les interrogations faites sur un entrepôt vont se décider au fur et à mesure de manière *ad hoc*. En outre, ces requêtes nécessitent le balayage de milliers, millions voire de milliards de tuples. Elles sont donc extrêmement coûteuses en temps de calcul alors que l'idéal pour les utilisateurs est que ce processus soit interactif.

GRAY, CHAUDHURI, BOSWORTH et al. 1997 ont introduit le concept de cube de données et l'opérateur associé `GROUP BY CUBE` pour concilier ces deux besoins contradictoires. On peut les voir comme une généralisation aux données multidimensionnelles des agrégats et de l'opérateur `GROUP BY`. L'idée sous-jacente est de pré-calculer et d'enregister tous les agrégats possibles, à différents niveaux de granularité. Ainsi, répondre à toute requête OLAP se réduit à une simple sélection de résultats préalablement enregistrés.

Dans la suite de ce mémoire nous supposons que les relations respectent les hypothèses et les notations suivantes : soit r une relation de schéma $\mathscr{R}$. Les attributs de $\mathscr{R}$ sont divisés en deux ensembles :

1. $\mathscr{D}$ l'ensemble des attributs dimensions, appelés aussi catégories. Il correspond aux critères d'analyse pour l'OLAP. De plus, les attributs de $\mathscr{D}$ sont totalement ordonnés (un tel ordre est noté $\geq_{\mathscr{D}}$) et $\forall A \in \mathscr{D}$, r($A$) est la projection de r sur A.
2. $\mathscr{M}$ l'ensemble des attributs mesures, sur lesquels sont appliquées les fonctions agrégatives. Par la suite, pour simplifier les formulations, on ne considérera qu'une seule mesure.

En plus des attributs de $\mathscr{R}$, nous considérons un attribut supplémentaire `RowId` qui est implicite et ne fait donc pas partie du schéma de la relation. Les valeurs de cet attribut servent d'identifiant unique à chaque tuple et sont attribuées au moment de leur insertion. On note t_i le tuple tel que t(`RowId`) = i.

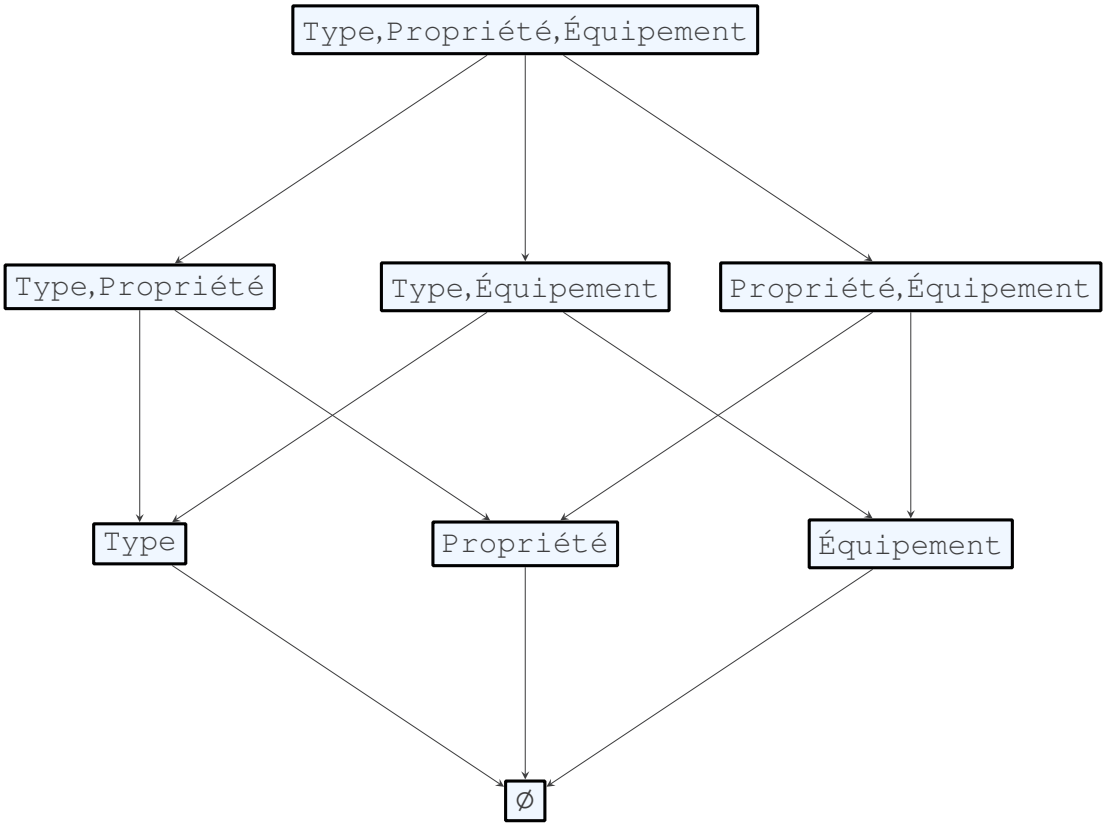


FIGURE 2.5. – Cuboïdes de `Diablo3` selon `Type`, `Propriété`, `Équipement`

Le résultat d'une requête de partitionnement, selon un ensemble $X \subseteq \mathscr{D}$ de dimensions, est appelé cuboïde, noté $cuboide_X(r)$. L'ensemble de tous les cuboïdes constitue le cube de données, noté $datacube(r)$. Les cuboïdes peuvent être ordonnés par rapport à leur niveau de détail par la relation d'ordre partiel $\preceq$. Par exemple le cuboïde selon les dimensions `Type`, `Propriété` est plus détaillé que le cuboïde

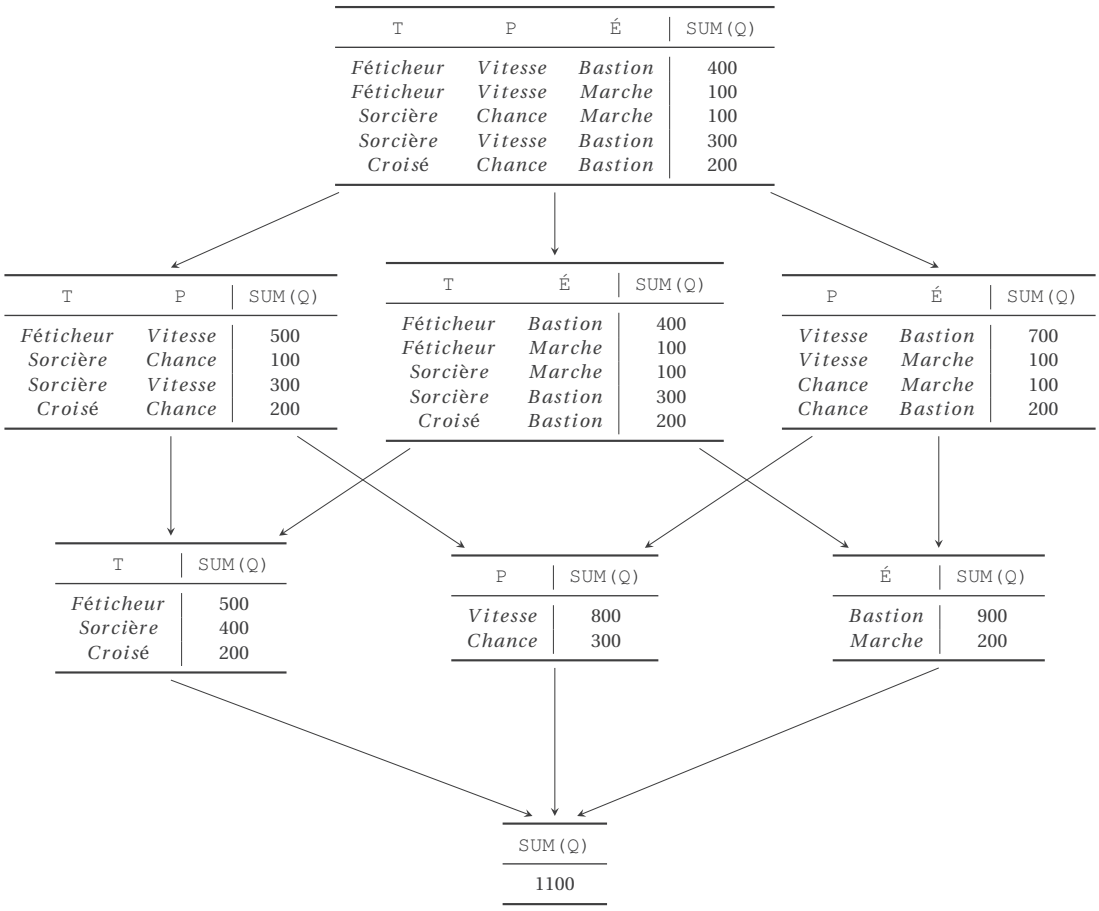


FIGURE 2.6. – Cuboïdes détaillés de `Diablo3` selon `Type`, `Propriété`, `Équipement`

selon le `Type` ou celui selon la `Propriété`. En d'autres termes, $cuboide_{\texttt{Type}}(r) \preceq cuboide_{\texttt{Type,Propriété}}(r)$ et $cuboide_{\texttt{Propriété}}(r) \preceq cuboide_{\texttt{Type,Propriété}}(r)$. L'ensemble des cuboïdes muni de cet ordre forme un treillis. La requête de calcul d'un cuboïde ressemble typiquement à :

```
01 |  SELECT X, f(M)
02 |  FROM r
03 |  GROUP BY X
```

Les cuboïdes de ce treillis sont regroupés par niveau en fonction de leur nombre de dimensions. Ces niveaux sont numérotés en partant du bas du treillis (cuboïde ne portant sur aucune dimension) et en remontant vers le sommet (cuboïde suivant tous les critères possibles appelé « cuboïde de base »). Soit deux cuboïdes $cuboide_{\mathcal{U}}(r)$ selon le sous-ensemble de dimensions $\mathcal{U}$ et $cuboide_{\mathcal{V}}(r)$ selon $\mathcal{V}$, si $\mathcal{U} \subset \mathcal{V}$ alors $cuboide_{\mathcal{U}}(r) \preceq cuboide_{\mathcal{V}}(r)$, on dit que les cuboïdes ont un lien de parenté, $cuboide_{\mathcal{V}}(r)$ est nommé « l'ancêtre » du $cuboide_{\mathcal{U}}(r)$ et $cuboide_{\mathcal{U}}(r)$ « le descendant » de $cuboide_{\mathcal{V}}(r)$. De plus, dans le cas où $|\mathcal{U}| = |\mathcal{V} - 1|$, $cuboide_{\mathcal{V}}(r)$ est appelé le cuboïde père de $cuboide_{\mathcal{U}}(r)$ et $cuboide_{\mathcal{U}}(r)$ le cuboïde fils de $cuboide_{\mathcal{V}}(r)$.

Les figures 2.5 et 2.6 donnent la représentation du treillis des cuboïdes pour la relation `Diablo3` suivant les dimensions `Type`, `Propriété`, `Équipement`.

Chaque cuboïde est a priori doté du schéma $X \cup \mathcal{M}$. Si le cube de données est calculé avec f comme fonction agrégative, nous notons $f_{val}(t,r)$ [4] la valeur de la fonction agrégative f associée au tuple t dans $datacube(r)$.

Dans notre exemple (cf. tableau 2.1), pour exprimer en SQL le cube de données nécessaire pour analyser, en `Quantité`, des personnages par `Type`, `Propriété`, `Équipement` (en ignorant l'attribut `Compagnon`), la requête suivante peut être formulée :

```
01 |  SELECT Type, Propriété, Équipement, SUM(Quantité)
02 |  FROM Diablo3
03 |  GROUP BY CUBE Type, Propriété, Équipement
```

Cette requête va calculer les $2^3 = 8$ partitionnements (où $\emptyset$ représente le partitionnement selon aucune dimension, *i.e.* agréger toute la relation en un seul tuple) :

— $'$`Type,Propriété,Équipement`$'$;
— $'$`Type,Propriété`$'$;
— $'$`Type,Équipement`$'$;
— $'$`Propriété,Équipement`$'$;
— $'$`Type`$'$;
— $'$`Propriété`$'$;
— $'$`Équipement`$'$;
— $'\emptyset'$.

La manière naïve de calculer ce type de requête est de la réécrire comme une collection de huit requêtes agrégatives et de les exécuter séparément. Cependant,

4. cf. section 2.3 p. 55

chacune de ses sous-requêtes a un schéma qui lui est propre. Pour rendre tous ces cuboïdes uni-compatibles, les attributs ne faisant pas partie des attributs de partitionnement (*i.e.* $\forall\ A \in \mathscr{D} \setminus X$) prennent la valeur ALL. Cette valeur, introduite dans GRAY, CHAUDHURI, BOSWORTH et al. 1997 pour la définition de l'opérateur `GROUP BY CUBE`, est une généralisation de toutes les valeurs du domaine d'un attribut dimension. Ainsi $\forall A \in \mathscr{D}, \forall a \in Dim(A), \{a\} \subset ALL$. Autrement dit, les cuboïdes partagent tous le même schéma ($\mathscr{D} \cup \mathscr{M}$) et peuvent être regroupés dans une seule et même relation : le cube de données. Les éléments de ce cube de données sont appelés tuples (ou cellules) multidimensionnels. Chaque tuple multidimensionnel est constitué d'un ensemble de valeurs pour ses dimensions et d'une valeur numérique pour la mesure. La valeur de la mesure est calculée en agrégeant l'ensemble des tuples de la relation originelle partageant les mêmes valeurs des dimensions sélectionnées.

Ainsi $\forall A \in \mathscr{D}, \forall a \in Dim(A),\ \{a\} \subset ALL$ (cf. sous-section 2.2.4 p. 34).

Par exemple, le tuple multidimensionnel $t = (Féticheur, Vitesse, ALL)$ est obtenu en agrégeant par `Type` et `Propriété` les tuples suivants de la relation exemple (cf. tableau 2.1) :

$$t_1 = (Féticheur, Vitesse, Bastion, Enchanteresse)$$
$$t_2 = (Féticheur, Vitesse, Marche, Templier)$$

La mesure de t est obtenue en appliquant la fonction mesure sur l'ensemble des tuples nécessaires à l'agrégation. Dans notre cas, nous avons : $f_{val}(t,r) = t_1(\mathscr{M}) + t_2(\mathscr{M}) = 400 + 100 = 500$.

Avec la relation exemple (cf. tableau 2.1), la requête pour calculer naïvement le cube de données suivant les dimensions `Type`, `Propriété`, `Équipement` est la suivante :

```
01 |  SELECT Type, Propriété, Équipement, SUM(Quantité)
02 |  FROM Diablo3
03 |  GROUP BY Type, Propriété, Équipement
04 |  UNION
05 |  SELECT Type, Propriété, ALL, SUM(Quantité)
06 |  FROM Diablo3
07 |  GROUP BY Type, Propriété
08 |  UNION
09 |  SELECT Type, ALL, Équipement, SUM(Quantité)
10 |  FROM Diablo3
11 |  GROUP BY Type, Équipement
12 |  UNION
13 |  SELECT ALL, Propriété, Équipement, SUM(Quantité)
14 |  FROM Diablo3
15 |  GROUP BY Propriété, Équipement
16 |  UNION
17 |  SELECT Type, ALL, ALL, SUM(Quantité)
18 |  FROM Diablo3
19 |  GROUP BY Type
20 |  UNION
21 |  SELECT ALL, Propriété, ALL, SUM(Quantité)
```

```
22 |  FROM Diablo3
23 |  GROUP BY Propriété
24 |  UNION
25 |  SELECT ALL, ALL, Équipement, SUM(Quantité)
26 |  FROM Diablo3
27 |  GROUP BY Équipement
28 |  UNION
29 |  SELECT ALL, ALL, ALL, SUM(Quantité)
30 |  FROM Diablo3
```

Le résultat de cette requête est donné par le tableau 2.2. Par souci de clarté, les différents cuboïdes sont séparés par une ligne horizontale.

TABLEAU 2.2. – Représentation en relation du cube de données de `Diablo3`

Type	Propriété	Équipement	Quantité
Féticheur	*Vitesse*	*Bastion*	400
Féticheur	*Vitesse*	*Marche*	100
Sorcière	*Chance*	*Marche*	100
Sorcière	*Vitesse*	*Bastion*	300
Croisé	*Chance*	*Bastion*	200
Féticheur	*Vitesse*	*ALL*	500
Sorcière	*Chance*	*ALL*	100
Sorcière	*Vitesse*	*ALL*	300
Croisé	*Chance*	*ALL*	200
Féticheur	*ALL*	*Bastion*	400
Féticheur	*ALL*	*Marche*	100
Sorcière	*ALL*	*Marche*	100
Sorcière	*ALL*	*Bastion*	300
Croisé	*ALL*	*Bastion*	200
ALL	*Vitesse*	*Bastion*	700
ALL	*Vitesse*	*Marche*	100
ALL	*Chance*	*Marche*	100
ALL	*Chance*	*Bastion*	200
Féticheur	*ALL*	*ALL*	500
Sorcière	*ALL*	*ALL*	400
Croisé	*ALL*	*ALL*	200
ALL	*Vitesse*	*ALL*	800
ALL	*Chance*	*ALL*	300
ALL	*ALL*	*Bastion*	900
ALL	*ALL*	*Marche*	200
ALL	*ALL*	*ALL*	1100

Cette approche naïve montre en partie la difficulté du calcul d'un cube de données. En effet, si la relation d'entrée possède $|\mathscr{D}|$ dimensions, le nombre de cuboïdes est

$2^{|\mathscr{D}|}$. Cela implique que le cube de données a une taille exponentielle par rapport au nombre de dimensions. Dans la plupart des entrepôts, la relation d'entrée est de l'ordre du Giga-octet avec un nombre de dimensions compris entre 3 et 20. Le développement de méthodes de calcul et d'enregistrement efficaces pour le cube de données est donc un problème critique. Ce problème a fait l'objet de nombreux travaux (MORFONIOS, KONAKAS, IOANNIDIS et al. 2007). Nous détaillons les plus importantes dans les paragraphes suivants.

Avant de nous intéresser à ces méthodes à proprement parler, nous allons d'abord étudier les propriétés sur lesquelles se basent les algorithmes pour optimiser le calcul.

2.2.4.1. Catégorie de fonctions agrégatives

Dans le calcul du cubes de données, la majeure partie du temps est utilisée à agréger les données et à calculer les valeurs de la fonction mesure. Ces fonctions ont été abondamment étudiées (HAN, KAMBER et PEI 2011). Les plus connues d'entre elles sont celles disponibles directement en SQL tel que : `COUNT`, `SUM`, `MIN`, `MAX`, `AVG`, `VAR`, `STDDEV`... Ces fonctions mesures peuvent être organisées en trois catégories.

Fonction distributive Une fonction agrégative f est distributive s'il existe une fonction g telle que $\forall I \subseteq r(RowId)$ et $\forall J$ partition de I on a :

$$f(\{t_i \mid i \in I\}) = g(\{f(\{t_j \mid j \in \mathcal{J}\}) \mid \mathcal{J} \in J\})$$

En d'autres termes, une fonction f est distributive si elle peut être calculée comme suit : supposons que la relation soit partitionnée en n sous-ensembles. Nous appliquons d'abord la fonction f sur chacun des sous-ensemble. La valeur de f sur l'intégralité de la relation est dérivée (grâce à une fonction g donnée) à partir de la valeur de f précédemment calculée sur les éléments de la partition. La fonction distributive f est dite additive si la fonction g est la somme.

Par exemple, la fonction `COUNT` peut être calculée pour un cube de données en partitionnant le cube en un ensemble de « sous-cubes ». Le cube de données est obtenu en sommant les valeurs des comptages pour chacun des sous-cubes. Pour les même raisons les fonctions `SUM`, `MIN`, `MAX` sont aussi distributives. Cette famille de fonctions est intéressante car elle permet de distribuer (ou découper) le calcul du cube de données.

Fonction algébrique Une fonction agrégative f est algébrique (ou calculable) s'il existe une fonction h à M arguments (avec M un entier naturel borné) et un ensemble $G = \{g_1, \ldots, g_M\}$ de M fonctions distributives, tel que : $\forall I \subseteq r(RowId)$ et $\forall J$ partition de I on a :

$$f(\{t_i \mid i \in I\}) = h(\{g_j(\{t_i \mid i \in I\}) \mid j = 1 \ldots M\})$$

Exprimé plus simplement, une fonction algébrique est une fonction qui peut être calculée à partir du résultat de M fonctions distributives. Généralement, on enregistre uniquement les résultats des M fonctions distributives dans le cube et la mesure

est dérivée à nouveau au besoin. Cette approche permet de bénéficier de tous les avantages des fonctions distributives pour des fonctions qui ne le sont pas.

Par exemple, `AVG` (la moyenne) peut être calculée en divisant le résultat de la fonction `SUM` par le résultat de la fonction `COUNT`. Dans M. C. CHEN et MCNAMEE 1989 une approche de paramétrisation est proposée et les expressions de calcul des principales fonctions statistiques sont détaillées : la covariance s'exprime à partir du comptage et de la somme de produits, l'analyse de régression est calculable à partir du comptage, somme, somme de produits, somme de carrés...

Fonction holistique Les fonctions holistiques sont celles qui ne sont ni distributives ni algébriques. Pour ces fonctions il n'existe pas de borne constante pour décrire l'agrégat avec des sous-agrégats. Dans ce cas, la mesure doit être recalculée à partir des données initiales sans pouvoir réutiliser des résultats intermédiaires. Les fonctions agrégatives holistiques typiques sont celles basées sur le rang comme la médiane, les k-meilleurs, les quartiles, ...

Fonctions agrégatives appliquées aux entrepôts de données Parmi ces catégories de fonctions, celles qui sont les plus intéressantes pour les entrepôts sont les fonctions distributives. En effet, le fait de pouvoir calculer un cube de données de manière distribuée a plusieurs conséquences importantes :

— lors d'un rafraîchissement de l'entrepôt (insertion de nouveaux tuples), si l'on considère ce dernier et l'ensemble des tuples à insérer comme une partition d'une nouvelle relation, il suffit de calculer le cube de données correspondant aux nouveaux tuples et, à partir des résultats obtenus, d'effectuer les mises à jour sur le cube de données précédemment calculé. Nous évitons ainsi un nouveau calcul complet du cube de données. Le rafraichissement est donc beaucoup moins coûteux;

— pendant la phase de calcul du cube de données, des résultats intermédiaires peuvent être exploités pour optimiser le temps de calcul. En effet, lorsque l'on agrège une relation r suivant un ensemble de dimensions X, les tuples de r sont partitionnés en fonction de leurs valeurs sur X. Tous les tuples ayant même valeur pour X sont regroupés dans une même classe d'équivalence. Si maintenant nous souhaitons agréger r suivant $Y \subset X$, il est possible d'utiliser le résultat du partitionnement selon X car chaque classe d'équivalence selon Y provient d'un ensemble de classes d'équivalence selon X. Grâce à la nature distributive de ses fonctions, nous pouvons agréger un agrégat et nous avons la garantie d'obtenir le bon résultat.

Exemple 2.1 - Pour la relation `Diablo3` (cf. tableau 2.1) où la fonction mesure utilisée est la somme (notée `SUM`) :

— si nous rajoutons le tuple $t_6 = (Croisé, Vitesse, Bastion, Enchanteresse)$ avec $Sum_{val}(t_6, r) = 200$ durant un rafraîchissement, alors pour mettre à jour le cube de données, il nous suffit de calculer le cube de données de ce tuple et de mettre à jour le tableau 2.2. Le tableau 2.3 donne ce cube de données. Les tuples grisés

sont ceux à ajouter dans le tableau 2.2, pour les autres, il faut modifier la valeur de la mesure en cumulant les valeurs présentes dans chacune des deux tables ;
— le tuple multidimensionnel $t = (Sorcière, ALL, ALL)$ peut être obtenu en agrégeant par `Type`, les tuples multidimensionnels $u = (Sorcière, Vitesse, ALL)$ et $v = (Sorcière, Chance, ALL)$ (cf. tableau 2.2). La fonction agrégative étant distributive, $Sum_{val}(t, r) = Sum_{val}(u, r) + Sum_{val}(v, r) = 300 + 100 = 400$.

TABLEAU 2.3. – Cube de données de $t_6 = (Croisé, Vitesse, Bastion, Enchanteresse)$

Type	Propriété	Équipement	Quantité
Croisé	*Vitesse*	*Bastion*	200
Croisé	*Vitesse*	*ALL*	200
Croisé	*ALL*	*Bastion*	200
ALL	*Vitesse*	*Bastion*	200
Croisé	*ALL*	*ALL*	200
ALL	*Vitesse*	*ALL*	200
ALL	*ALL*	*Bastion*	200
ALL	*ALL*	*ALL*	200

La plupart des applications requérant le calcul d'un cube de données utilisent des fonctions distributives ou algébriques. Cependant, si la relation comporte de nombreux tuples, le calcul rapide du cube de données pour une fonction agrégative holistique peut poser des problèmes. C'est pourquoi, il existe des techniques d'approximation pour ces fonctions (HAN, KAMBER et PEI 2011).

2.2.4.2. Variantes du cube de données

Le cube de données a été proposé comme un outil, permettant aux décideurs d'analyser leurs données selon plusieurs dimensions pour en extraire les grandes tendances apparaissant et en comprendre leurs origines. Selon le type d'analyse à mener, il peut être intéressant de se focaliser seulement sur une certaine partie du cube. Les données trop détaillées ou trop générales par exemple, peuvent n'apporter que peu d'informations utiles. C'est pour répondre à ce type de besoin que plusieurs variantes du cube de données sont apparues ces dernières années. Chacune d'entre elles permet à l'utilisateur de définir des contraintes que doivent respecter les tuples pour qu'ils soient considérés pertinents.

Cubes icebergs En s'inspirant des motifs fréquents, BEYER et RAMAKRISHNAN 1999 introduisent les cubes icebergs. Ces derniers, sont présentés comme des sous-ensembles de tuples du cube de données qui satisfont, pour les valeurs de la mesure, une contrainte de seuil. L'objectif sous-jacent est triple. En premier lieu, il s'agit d'exhiber les tendances suffisamment fortes pour être pertinentes pour le décideur. Il en découle deux intérêts techniques importants : ne pas calculer ni matérialiser la totalité

du cube, d'où un gain notable à la fois en temps d'exécution et en espace disque. La requête SQL permettant de calculer un cube de données iceberg a la forme suivante :

```
01 | SELECT D, f({M|*})
02 | FROM r
03 | GROUP BY CUBE D
04 | HAVING f({M|*}) >= MinSeuil;
```

TABLEAU 2.4. – Représentation en relation du cube de données iceberg de `Diablo3`

Type	Propriété	Équipement	Quantité
Féticheur	*Vitesse*	*Bastion*	400
Sorcière	*Vitesse*	*Bastion*	300
Féticheur	*Vitesse*	*ALL*	500
Sorcière	*Vitesse*	*ALL*	300
Féticheur	*ALL*	*Bastion*	400
Sorcière	*ALL*	*Bastion*	300
ALL	*Vitesse*	*Bastion*	700
Féticheur	*ALL*	*ALL*	500
Sorcière	*ALL*	*ALL*	400
ALL	*Vitesse*	*ALL*	800
ALL	*Chance*	*ALL*	300
ALL	*ALL*	*Bastion*	900
ALL	*ALL*	*ALL*	1100

Exemple 2.2 - Avec la relation exemple `Diablo3` (cf. tableau 2.1), le cube de données iceberg obtenu pour la contrainte « `SUM(Quantité)` ≥ 300 » est donné dans le tableau 2.4.

Cubes intervallaires Le cube intervallaire, introduit par NEDJAR, CASALI, CICCHETTI et al. 2010a, ne contient que les tuples du cube de données pour lesquels les valeurs de la mesure sont comprises dans un intervalle donné. La requête SQL permettant de calculer un tel cube est la suivante :

```
01 | SELECT D, f({M|*})
02 | FROM r
03 | GROUP BY CUBE D
04 | HAVING f({M|*}) BETWEEN MinSeuil AND MaxSeuil;
```

Exemple 2.3 - Avec la relation exemple `Diablo3` (cf. tableau 2.1), le cube intervallaire obtenu pour la contrainte « 300 ≤ `SUM(Quantité)` ≤ 700 » est donné dans le tableau 2.5.

TABLEAU 2.5. – Représentation en relation du cube intervallaire de Diablo3

Type	Propriété	Équipement	Quantité
Féticheur	*Vitesse*	*Bastion*	400
Sorcière	*Vitesse*	*Bastion*	300
Féticheur	*Vitesse*	*ALL*	500
Sorcière	*Vitesse*	*ALL*	300
Féticheur	*ALL*	*Bastion*	400
Sorcière	*ALL*	*Bastion*	300
ALL	*Vitesse*	*Bastion*	700
Féticheur	*ALL*	*ALL*	500
Sorcière	*ALL*	*ALL*	400
ALL	*Chance*	*ALL*	300

Il est important de remarquer que tout algorithme pouvant calculer le cube de données iceberg ou le cube intervallaire peut aussi être utilisé pour calculer le cube de données complet. Il suffit pour le cube de données iceberg de choisir comme contrainte $\texttt{COUNT}(*) \geq 1$ ou pour le cube intervallaire la contrainte $1 \leq \texttt{COUNT}(*) \leq \infty$.

2.2.5. Calcul du cube de données

2.2.5.1. L'algorithme Partitioned-Cube

L'algorithme Partitioned-Cube (ROSS et SRIVASTAVA 1997), vise à corriger les défauts des approches antérieures (les algorithmes 2^D et GBLP proposé par GRAY, CHAUDHURI, BOSWORTH et al. 1997 et les algorithmes PipeSort et Overlap de AGARWAL, AGRAWAL, DESHPANDE et al. 1996) avec les jeux de données peu denses. Ce travail est particulièrement important car les relations réelles ont souvent des attributs avec de larges domaines et sont donc généralement parsemées. La méthode présentée repose sur deux algorithmes distincts. Le premier, Partitioned-Cube, a pour objectif de découper (partitionner) récursivement la relation d'entrée jusqu'à obtenir une sous-relation dont le cube de données peut être directement calculé en mémoire. En réunissant les différents sous-cubes, l'algorithme dérive l'intégralité du cube de données souhaité. Le second algorithme, nommé Memory-Cube, est dédié au calcul de cube de données. Il est tout à fait classique mais optimisé pour fonctionner exclusivement en mémoire centrale. C'est grâce à cet algorithme que Partitioned-Cube calcule tous les sous-cubes. Les deux algorithmes étant indépendants, tout algorithme peut bénéficier des qualités de Partitioned-Cube. Cette approche est tellement efficace qu'à l'heure actuelle, les méthodes de calcul de cubes font quasi-systématiquement l'hypothèse implicite de disposer de cette méthode de partitionnement et supposent donc que les relations rentrent toujours en mémoire centrale.

Partitioned-Cube est décrit dans l'algorithme 1. C'est un algorithme récursif utili-

sant une stratégie « diviser pour régner » déjà employée avec succès pour répondre efficacement à des opérations complexes telles que la jointure ou le tri : découper la relation d'entrée jusqu'à obtenir des partitions tenant en mémoire, calculer l'opération souhaitée sur chacun d'eux, puis réunir les résultats pour obtenir la solution désirée. L'algorithme transforme le calcul d'un cube à k dimensions avec une relation d'entrée de taille T en $n+1$ calculs de sous-cubes. Les n premiers sous-cubes ont k dimensions et sont basés sur approximativement T/n tuples. Le dernier sous-cube se base sur moins de T tuples et comporte $k-1$ dimensions.

À chaque étape l'algorithme procède comme suit : il choisit une dimension $d \in \mathscr{D}$, il partitionne la relation d'entrée r en n parties en fonction des valeurs des attributs pour cette dimension. Sur chacune de ces partitions, plus réduit que la relation r, il calcule les sous-cubes de données. Puis il réunit les résultats au sein d'un premier sous-cube noté $SousCube_1$. Il est constitué de tous les cuboïdes contenant la dimension d. Après cette opération, l'algorithme calcule un deuxième sous-cube, noté $SousCube_2$, qui porte sur les dimensions $\mathscr{D} \setminus d$. Ce cube est donc constitué de tous les cuboïdes ne contenant pas d. Pour le déterminer, on prend comme entrée, non pas la relation r, mais le cuboïde le moins agrégé de $SousCube_1$ déjà calculé. Ce cuboïde étant une agrégation de r, il a une cardinalité inférieure ou égale à celle de r.

Les deux sous-cubes contiennent chacun la moitié des nœuds du cube. Les sous-cubes sont tous de taille inférieure à celle de r et ont moins de dimensions. L'algorithme a donc divisé le problème en deux sous-problèmes plus facilement résolvables. On applique récursivement cette procédure jusqu'à obtenir des jeux de données traitables. Cette méthode, réduisant à chaque étape la difficulté, est certaine de se terminer en un temps fini.

Exemple 2.4 - La figure 2.7 montre le partitionnement choisi par Partitioned-Cube pour calculer le cube de données de la relation `Diablo3` (cf. tableau 2.1). Les nœuds en pointillés correspondent aux différents appels de la fonction `Partitioned-Cube`.

Les principaux avantages de Partitioned-Cube sont sa souplesse et son faible coût en Entrées/Sorties (E/S) :

— en effet, l'algorithme est particulièrement souple car le choix des dimensions et la méthode de partitionnement sont libres. L'algorithme peut donc toujours s'adapter quelle que soit la nature de la relation. Par exemple, si la dimension de partitionnement contient des données fortement biaisées, il est possible de découper une partition en plusieurs et inversement, regrouper en une seule autant de partitions que la mémoire le permet ;

— de plus, si un attribut est ajouté, on ne fait dans le pire des cas qu'un appel supplémentaire à l'algorithme en réutilisant les résultats calculés. Les E/S sont donc largement diminuées ; lorsque l'algorithme est utilisé avec Memory-Cube, le coût des E/S croît proportionnellement au nombre de dimensions ; son comportement est ainsi bien meilleur que Overlap et PipeSort qui ont respectivement un coût quadratique et exponentiel.

Dans la suite du travail Partitioned-Cube sera utilisé quasi-systématiquement pour

Algorithme 1 Algorithme Partitioned-Cube

Entrée :
La relation r.
L'ensemble de dimensions $\mathscr{D}$.
Sortie :
Le cube de données de r divisé en deux : F le cuboïde le moins agrégé, et D l'ensemble des autres cuboïdes.
01: **si** r tient en mémoire centrale **alors**
02: **retourner** Memory-Cube($r,\mathscr{D}$)
03: **fin si**
04: choisir une dimension $d \in \mathscr{D}$
05: partitionner r, selon d, en n sous-relations : $r_1,\ldots,r_n$
06: **pour** $i = 1,\ldots,n$ **faire**
07: (F_i, D_i) = Partitioned-Cube($r_i,\mathscr{D}$) //Calcul de $SousCube_1$
08: **fin pour**
09: $F := \bigcup_{i=1}^{n} F_i$
10: (F', D') = Partitioned-Cube($F, \mathscr{D} \setminus d$) //Calcul de $SousCube_2$
11: $D := (\bigcup_{i=1}^{n} D_i) \cup F' \cup D'$
12: **retourner** (F, D)

garantir aux algorithmes que les relations d'entrée rentrent toujours en mémoire centrale.

2.2.5.2. L'algorithme BUC

L'algorithme BUC (pour *bottom up cube*), proposé par BEYER et RAMAKRISHNAN 1999, est le premier algorithme de calcul efficace pour le cube de données iceberg et le cube de données. La plus grande innovation introduite par cet algorithme provient de son sens de parcours du treillis des cuboïdes de bas en haut. En effet, tous ses prédécesseurs ont voulu optimiser le temps de réponse en évitant de faire des opérations de tris inutiles et en partageant autant que possible les calculs d'agrégats entre les différents cuboïdes. Or BUC, originellement conçu pour le cube de données iceberg, choisit de parcourir le treillis des cuboïdes en partant du bas (du plus agrégé vers le moins agrégé) pour pouvoir appliquer une stratégie d'élagage, inspirée de la fouille de données binaire, à la apriori (AGRAWAL, IMIELINSKI et SWAMI 1993). Ce choix lui permet d'économiser énormément de temps en ne calculant que les agrégats utiles. En plus d'être particulièrement efficace pour le cube de données iceberg et le cube de données, l'approche adoptée par BUC a le mérite de ne pas s'appuyer sur des structures de données complexes supplémentaires. Il est ainsi directement intégrable au sein d'un SGBD. MORFONIOS et IOANNIDIS 2008 ont même montré qu'en plus de calculer le cube de données, on pouvait aussi l'utiliser avec un grand bénéfice tout au long du cycle de vie d'un cube de données (de la construction à maintenance incrémentale en passant par le calcul des requêtes et l'indexation). C'est grâce à toutes

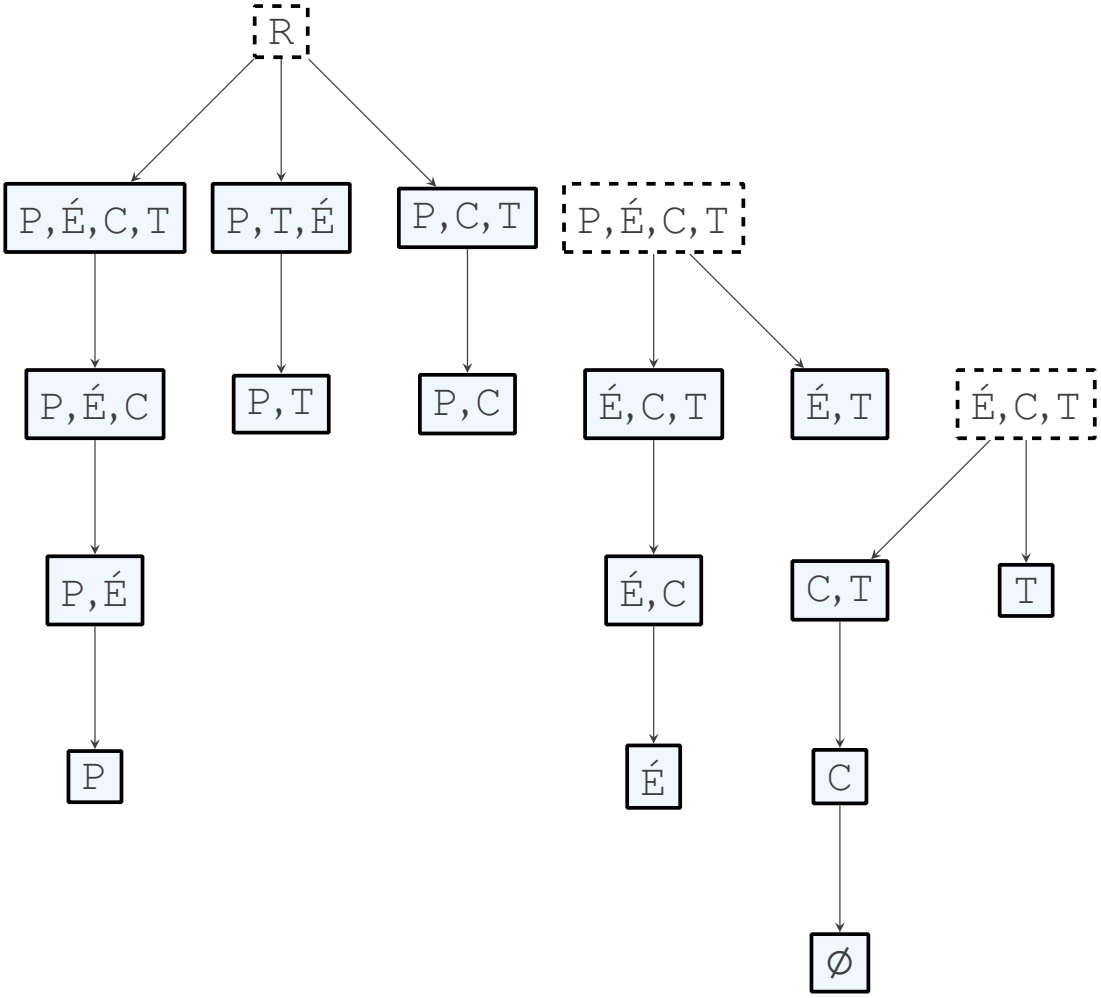


FIGURE 2.7. – Graphe d'exécution de Partitioned-Cube pour la relation `Diablo3`

ces qualités, que les algorithmes basés sur BUC restent aujourd'hui encore parmi les meilleurs concurrents pour le calcul du cubes de données et de certaines de ses variantes.

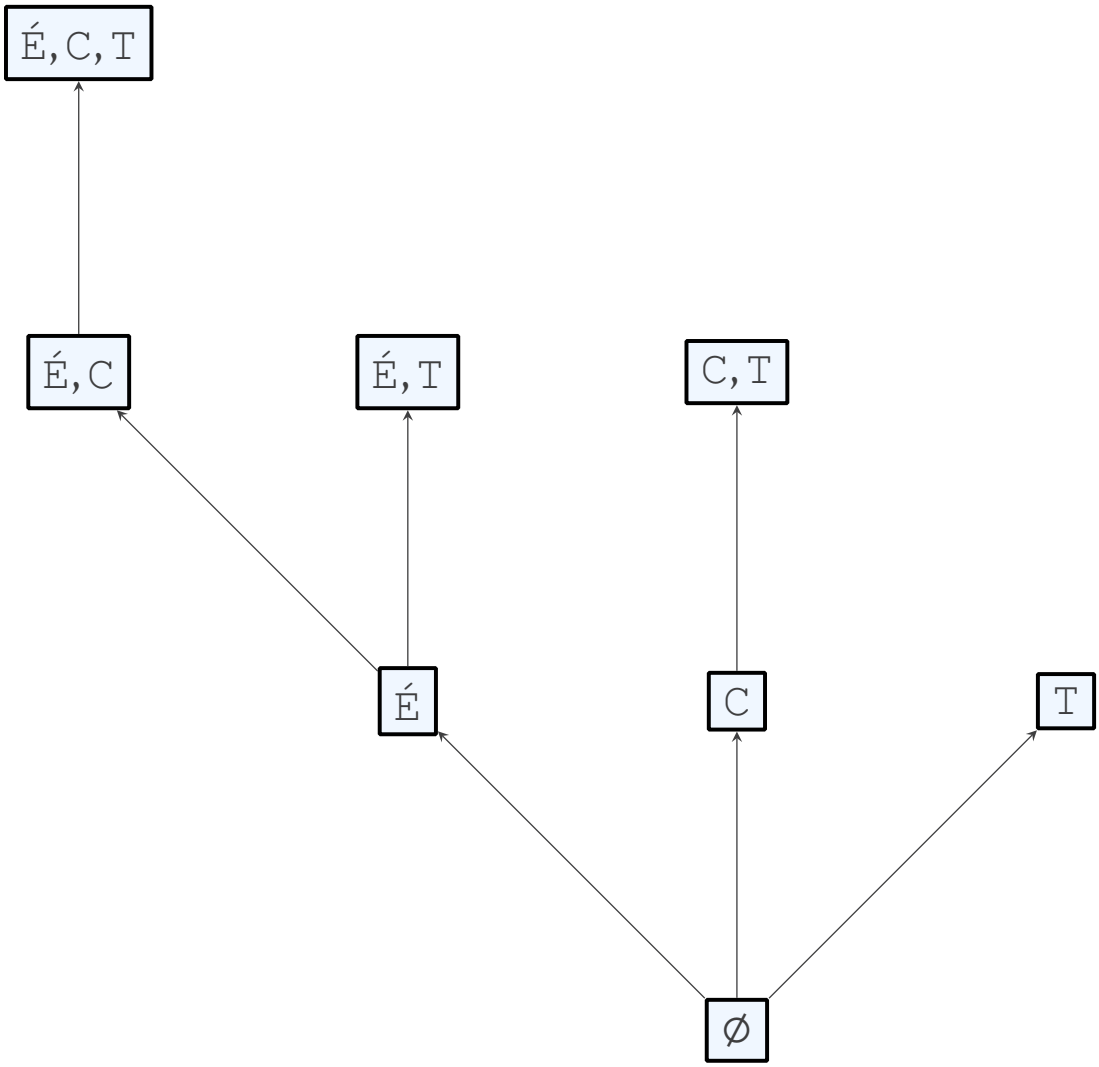


FIGURE 2.8. – Arbre de parcours des cuboïdes par BUC pour la relation `Diablo3`

BUC, comme son nom l'indique, parcourt son espace de recherche de bas en haut. Il part du cuboïde selon $\emptyset$ et remonte à travers le treillis des cuboïdes en utilisant un ordre lexicographique. Pour simplifier la présentation, nous supposerons que l'ordre entre attributs ($\geq_{\mathscr{D}}$) est l'ordre alphabétique. Ainsi, l'ordre lexicographique de parcours des nœuds fait par l'algorithme est identique à l'ordre du dictionnaire. La relation d'ordre $\geq_{\mathscr{D}}$ n'a aucune influence sur la justesse de l'algorithme mais peut avoir des conséquences importantes sur son efficacité. En pratique, plusieurs heuristiques pour trouver le meilleur ordre ont été proposées par BEYER et RAMAKRISHNAN 1999 et expérimentées par NASR et BADR 2003.

BUC est un algorithme récursif basé sur le partitionnement horizontal. Assez gros-

sièrement, on peut imaginer BUC comme une version de Partitioned-Cube qui ne fait jamais appel à Memory-Cube. Ainsi, il bénéficie à la fois de la bonne efficacité des E/S de Partitioned-Cube et de l'élagage à la apriori. Il est décrit dans l'algorithme 4. Au premier appel, la cellule courante passée en paramètre est $(ALL, \ldots, ALL)$. La première agrégation fait donc un passage sur r pour calculer la valeur de la mesure (cf. algorithme 2). Une fois cette étape réalisée, on vérifie si r ne contient qu'un seul tuple. Dans ce cas, le cube de données devient évident à calculer, l'algorithme s'arrête et ÉcrireSuivants est appelé (cf. algorithme 3). Ce dernier remplace, attribut après attribut, chaque valeur ALL de la cellule courante par celle de l'unique tuple de la relation. Cette étape est particulièrement importante dans le cas des données peu denses où il y a potentiellement de nombreuse partitions très réduites. L'algorithme, dans ce cas, économise de nombreux appels récursifs et donc beaucoup de temps. Si la relation d'entrée n'est pas réduite à un seul tuple, la cellule courante est écrite. Expliquons maintenant le cœur de l'algorithme. Pour chacune des dimensions d_j passées en paramètre, la relation courante est partitionnée en fonction de ses valeurs sur l'attribut d_j. Deux tuples t et t' seront dans la même partition si et seulement si $t[d_j] = t'[d_j]$. Cette étape nécessite de faire un tri, c'est donc la plus coûteuse. Pour obtenir une efficacité optimale, les auteurs ont choisi d'utiliser un tri par comptage dont la complexité en temps est linéaire. Partitionner est décrit dans l'algorithme 5. Une fois cette opération effectuée, BUC est appelé récursivement pour chacune des partitions vérifiant la « contrainte iceberg ». Lorsque toutes les partitions ont été traitées pour la dimension d_j, tous les cuboïdes contenant d_j l'ont été également. La valeur ALL est alors mise dans la dimension d_j de la cellule courante.

Algorithme 2 Algorithme Agréger

Entrée :
Une relation r.
La cellule courante $CelluleCour$.
Une fonction agrégative additive f.
01: **pour tout** $t \in r$ **faire**
02: $CelluleCour.M = CelluleCour.M + f(t)$;
03: **fin pour**

Algorithme 3 Algorithme ÉcrireSuivants

Entrée :
Un tuple t.
L'ensemble des dimensions à traiter $\mathscr{D} = \{d_1, \ldots, d_n\}$.
La cellule courante $CelluleCour$.
01: **pour tout** $d_j \in \mathscr{D}$ **faire**
02: $CelluleCour.d_j := t.d_j$;
03: ÉCRIRE($CelluleCour$);
04: **fin pour**

Algorithme 4 Algorithme BUC

Entrée :
Une relation r.
L'ensemble des dimensions $\mathscr{D} = \{d_1, \ldots, d_n\}$ à traiter.
La cellule courante $CelluleCour$.
Sortie :
Le cube de données de r.

```
01: DimCour := d_1;
02: Agréger(r, CelluleCour, f); //Calcule la valeur de la
    fonction mesure et l'écrit dans CelluleCour
03: si |r| = 1 alors
04:   ÉcrireSuivants(r[0], 𝒟, CelluleCour); //La relation n'a
      qu'un tuple, on écrit les résultats sans agréger
05:   retourner
06: fin si
07: Écrire(CelluleCour)
08: pour tout d_j ∈ 𝒟 faire
09:   C := |r(d_j)|;
10:   Partitionner(r, d_j); //r est partitionnée suivant d_j en
      C partitions : r_1, ..., r_C
11:   pour i = 1, ..., C faire
12:     si |r_i| ≥ MinSeuil alors
13:       //CelluleCour reçoit la valeur de d_j pour r_i
14:       CelluleCour.d_j := r_i[0].d_j;
15:       BUC(r_i, {d_{j+1}, ..., d_n}, CelluleCour);
16:     fin si
17:   fin pour
18:   CelluleCour.d_j := ALL; //d_j est entièrement traitée
19: fin pour
```

Algorithme 5 Algorithme Partitionner

```
Entrée :
   Une relation r.
   La dimension dimCour du partitionnement de r.
Sortie :
   La relation r partitionnée suivant les valeurs de
   l'attribut dimCour en C partitions : r_1,...,r_C.
01: soit C : le nombre de valeurs de dimCour dans r
02: soit hist : un tableau de |Dom(dimCour)| entiers
   initialisés à 0
03: soit nbTuples : le nombre de tuples dans r
04: pour i = 1,...,nbTuples faire
05:   hist[t_i.dimCour] := hist[t_i.dimCour] + 1; //On compte le nombre
     d'occurences de chaque valeur de dimCour
06: fin pour
07: //On exploite l'histogramme
08: soit indTab : un tableau d'entiers initialisés à 0
09: soit premier : un entier initialisé à 0
10: soit dernier : un entier initialisé à 0
11: pour i = 1,...,|Dom(dimCour)| faire
12:   si hist[i] ≠ 0 alors
13:     C := C + 1;
14:   fin si
15:   dernier := hist[i];
16:   indTab[i] := premier;
17:   premier := premier + dernier;
18: fin pour
19: soit indTabTmp : une copie locale de indTab;
20: pour i = 1,...,nbTuples faire
21:   soit v : la valeur du tuple t_i pour dimCour;
22:   rTmp[indTabTmp[v]] := t_i;
23:   indTabTmp[v] := indTabTmp[v] + 1;
24: fin pour
25: soit j : un entier initialisé à 1;
26: pour i = 1,...,|Dom(dimCour)| faire
27:   si hist[i] ≠ 0 alors
28:     r_j := rTmp[indTab[i]..indTab[i] + hist[i]];
29:     j := j + 1;
30:   fin si
31: fin pour
32: retourner  r_1,...,r_C;
```

Exemple 2.5 - La figure 2.8 montre le parcours du treillis des cuboïdes employé par l'algorithme BUC pour calculer le cube de données de la relation `Diablo3` (cf. tableau 2.1). Lors du premier appel, BUC calcule la valeur de la fonction agrégative en parcourant l'intégralité de la relation `Diablo3` obtenant ainsi le tuple multidimensionnel (*ALL*, *ALL*, *ALL*) : 1100, directement écrit sur le disque.

RowId	É	C	T	Q
1	*Bastion*	*Enchanteresse*	*Féticheur*	400
2	*Marche*	*Templier*	*Féticheur*	100
3	*Marche*	*Enchanteresse*	*Sorcière*	100
4	*Bastion*	*Templier*	*Sorcière*	300
5	*Bastion*	*Enchanteresse*	*Croisé*	200

Agréger → Écriture sur disque : (*ALL*, *ALL*, *ALL*) : 1100

Une fois cette étape réalisée, BUC partitionne la relation `Diablo3` suivant les valeurs de l'attribut `Équipement` (le premier dans l'ordre alphabétique). Cette dimension prenant pour valeurs simplement $Bastion$ et $Marche$, la relation est alors découpée en deux partitions $P_1 = \{t_1, t_4, t_5\}$ et $P_2 = \{t_2, t_3\}$.

L'algorithme traite ensuite les partitions une par une. Pour P_1, la « contrainte iceberg » est contrôlée et si elle est satisfaite, la cellule courante est mise à jour avec la valeur des tuples de P_1 sur la dimension `Équipement` ($CelluleCour$ = ($Bastion$, ALL, ALL)). Puis BUC est appelé récursivement pour P_1.

RowId	É	C	T	Q
1	*Bastion*	*Enchanteresse*	*Féticheur*	400
4	*Bastion*	*Templier*	*Sorcière*	300
5	*Bastion*	*Enchanteresse*	*Croisé*	200

Aggréger → Écriture sur disque : (*Bastion*, *ALL*, *ALL*) : 900

P_1 est partitionné sur la dimension suivante, `Compagnon`, en deux parties $P_3 = \{t_4\}$ et $P_4 = \{t_1, t_5\}$.

$P3$ est à présent traité, alors $CelluleCour = (Bastion, Templier, ALL)$. La partition ne contenant plus qu'un seul tuple il est inutile de calculer la valeur de la fonction aggrégative. L'algorithme écrit directement sur disque tous les descendants de ($Bastion$, $Templier$, ALL).

RowId	É	C	T	Q
4	*Bastion*	*Templier*	*Sorcière*	300

EcrireSuivant → Écriture sur disque :
(*Bastion*, *Templier*, *ALL*) : 300
(*Bastion*, *Templier*, *Sorcière*) : 300

Le même processus est exécuté pour la partition P_4.

RowId	É	C	T	Q
1	*Bastion*	*Enchanteresse*	*Féticheur*	400
5	*Bastion*	*Enchanteresse*	*Croisé*	200

Aggréger → Écriture sur disque : (*Bastion*, *Enchanteresse*, *ALL*) : 600

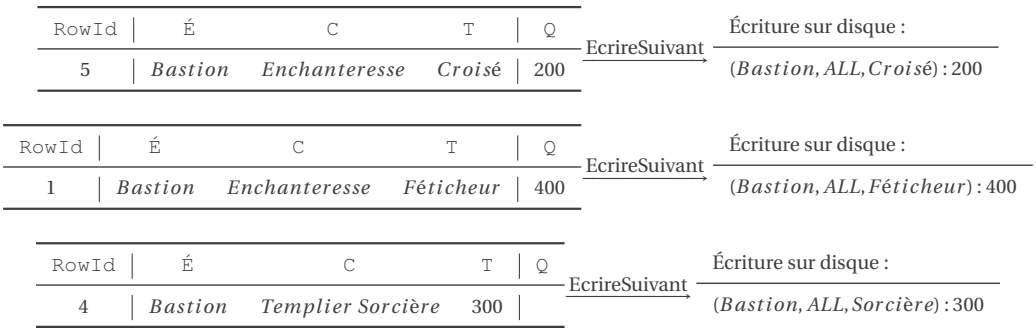
RowId	É	C	T	Q
5	*Bastion*	*Enchanteresse*	*Croisé*	200

EcrireSuivant → Écriture sur disque : (*Bastion*, *Enchanteresse*, *Croisé*) : 200

RowId	É	C	T	Q
1	*Bastion*	*Enchanteresse*	*Féticheur*	400

EcrireSuivant → Écriture sur disque : (*Bastion*, *Enchanteresse*, *Féticheur*) : 400

Chacune des partitions P_5 et P_6 n'est constituée que d'un seul tuple, les résultats sont écrits sans faire de calcul. L'algorithme remonte jusqu'à P_1 qu'il partitionne à nouveau suivant l'attribut `Type`. Il obtient ainsi trois nouvelles partitions P_7, P_8 et P_9. Chacune d'entre elles est constituée d'un unique tuple, l'algorithme écrit les tuples multidimensionnels associés sans calcul.



Une fois toutes ces partitions traitées, l'algorithme remonte et traite P_2 récursivement. Les cuboïdes contenant la dimension `Équipement` ont été générés. Cette dimension est donc ignorée par BUC à partir de maintenant. L'algorithme part ensuite de la relation d'origine en démarrant le partitionnement avec la dimension `Compagnon` et effectue le processus précédemment décrit. BUC se termine lorsque tous les attributs de la relation `Diablo3` ont été traités de la sorte.

La figure 2.9 donne les détails de toutes les partitions calculées par BUC durant le parcours de son espace de recherche. À l'intérieur de chaque nœud figurent les tuples de cette partition ainsi que les attributs de partitionnement. Les arêtes en pointillés correspondent aux nœuds obtenus sans calcul d'agrégation grâce à l'algorithme EcrireSuivant. Les nœuds en pointillés, quant à eux, correspondent au moment où BUC fait un retour sur trace et réutilise la même entrée pour continuer son calcul.

2.2.6. Synthèse

Le cube de données est le concept majeur pour la gestion des entrepôts de données. Dans cette partie, nous avons proposé un survol de travaux centrés sur ce concept. Plus précisément, nous avons rappelé ce qu'est un cube de données et deux des variantes qui ont été introduites. Puis nous nous sommes focalisés sur les approches permettant de calculer des cubes complets. Il existe aussi des moyens de calculer des cubes partiels, que nous n'avons pas traité ici. De très nombreux travaux ont abordé ces problématiques, si bien qu'il était délicat d'en brosser un tableau exhaustif. Un des principaux critères de choix qui nous a fait retenir les approches présentées est leur propriété d'intégration au sein des SGBD. Cette qualité reste rare car de nombreuses approches ont recours à des structures de données trop spécifiques et non adaptables dans un contexte relationnel. Citons notamment les trees, FP-trees et leurs extensions (Han, Pei et Yin 2000 ; Pei, Han et Mao 2000 ; Han, Pei, Dong et al. 2001 ; Han, Y. Chen, Dong et al. 2005 ; Xin, Shao, Han et al. 2006 ; Cheng, Yu et Han 2006 ; J. Wang, Han et Chun Li 2007 ; Xin, Han, X. Li et al. 2007 ; N. Li,

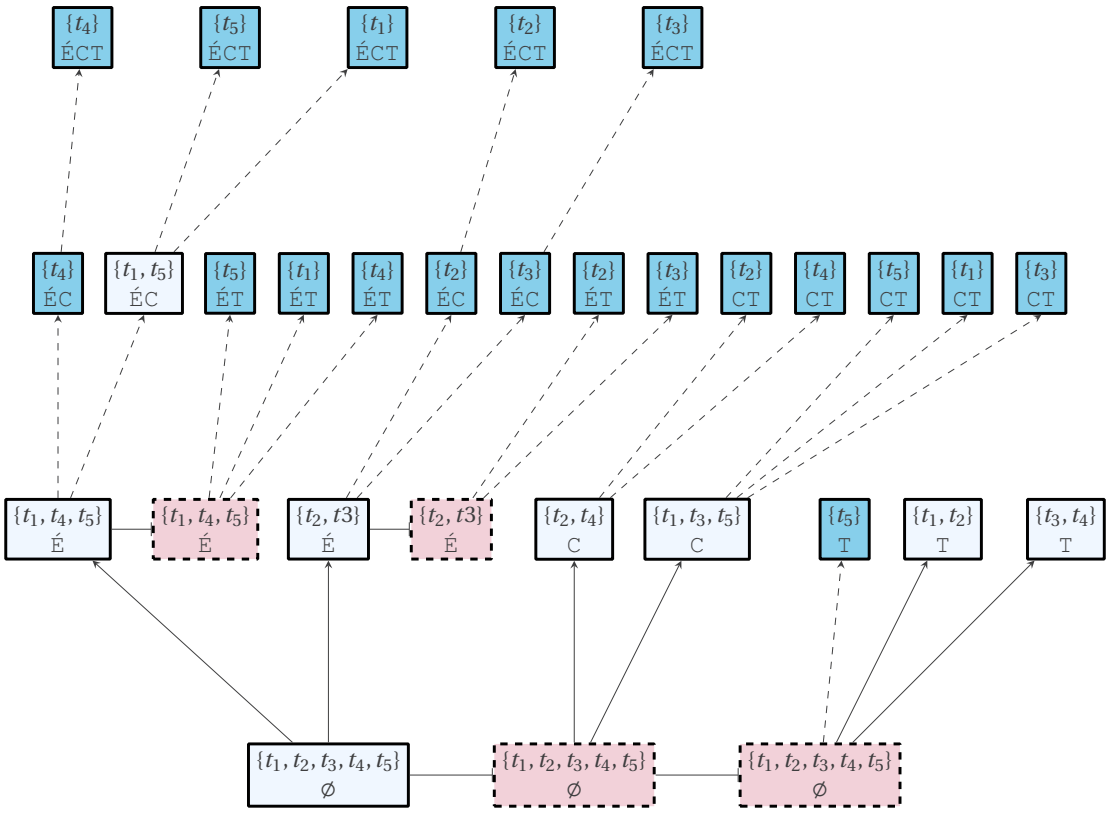


FIGURE 2.9. – Arbre de parcours des partitions par BUC pour la relation `Diablo3`

GUAN, REN et al. 2013 ; DOAN, QI, RAJASEGARAR et al. 2018). Les algorithmes que nous avons décrits représentent une sorte d'historique en matière de calcul de cubes. Le point commun de la première génération d'approches est de se concentrer sur la construction du meilleur arbre d'exécution possible à partir du treillis des cuboïdes. L'idée sous-tendant ces approches est de ré-exploiter des résultats intermédiaires afin d'optimiser les temps de calcul. Évidemment, ceci implique un parcours des cuboïdes les plus détaillés vers ceux qui sont de plus en plus agrégés. Avec la seconde génération, l'idée de construire un arbre d'exécution a été abandonnée. Ainsi en se libérant de cette contrainte, le sens de parcours du treillis a été inversé, donnant naissance à des algorithmes bien plus efficaces que leurs prédécesseurs. Nous avons présenté Partitioned-Cube car il est incontournable pour toute méthode souhaitant s'affranchir des problèmes d'accès à la mémoire secondaire. Les raisons qui nous ont fait retenir BUC sont ses très bonnes performances, sa simplicité, sa polyvalence et bien sur, son intégrabilité.

2.3. Cube de données fermé

2.3.1. Introduction

Il existe des points communs et des convergences entre les approches de fouille de données classiques et les travaux sur les cubes de données. En témoigne l'algorithme BUC (BEYER et RAMAKRISHNAN 1999) qui se base sur les propriétés exploitées par Apriori (AGRAWAL, MANNILA, SRIKANT et al. 1996) pour optimiser le calcul du cube de données iceberg. Bien que développés dans un contexte de données binaires, les algorithmes de fouille peuvent être adaptés pour le calcul de cubes de données. Mais ce type d'extension a des effets indésirables. Les approches binaires classiques considèrent un ensemble *d'items* qui représentent les valeurs d'un seul attribut (*e.g.* l'identifiant des produits pour les bases de transactions commerciales). Dès lors, transposer des solutions performantes lorsqu'un seul attribut est considéré au cas où quelques dizaines d'attributs doivent être manipulés a des conséquences importantes.

— Si l'on considère le treillis des parties comme l'espace de recherche dans un contexte multidimensionnel, l'ensemble des *items* à considérer est l'ensemble de toutes les valeurs de tous les attributs dimensions. Il y a alors explosion de l'espace de recherche. Dans le cadre d'applications OLAP, le problème s'aggrave car les attributs dimensions peuvent avoir des domaines de définition extrêmement larges (ROSS et SRIVASTAVA 1997 ; BEYER et RAMAKRISHNAN 1999) ;

— À partir de l'ensemble des valeurs de tous les attributs, le treillis des parties contient toutes les combinaisons possibles de ces valeurs. Ces dernières, par construction même, vont pouvoir associer deux valeurs d'un même attribut. De telles combinaisons sont, sémantiquement, invalides d'un point de vue multidimensionnel. En effet, le point de départ du calcul de cube étant une relation classique et normalisée (1-NF), toute combinaison incluant deux valeurs d'un même attribut est un « motif impossible ». Pourtant, il est considéré par les

approches transposant des solutions binaire à l'environnement multidimensionnel (SRIKANT et AGRAWAL 1996 ; B. LIU, HSU et MA 1998 ; W. LI, HAN et PEI 2001). Il en résulte trois conséquences :

— le calcul des mesures, qui nécessite un parcours des données, est effectué pour ces motifs impossibles alors qu'il est inutile ;
— si la contrainte antimonotone de fréquence est mise en œuvre, de tels motifs sont éliminés du résultat mais dégradent la complexité ;
— pour d'autres types de contraintes monotones ou antimonotones, ces motifs impossibles deviennent bel et bien des solutions du problème.

Pour éviter les problèmes présentés, certaines méthodes dédiées au contexte OLAP ont été proposées. Cette section leur est consacrée. Nous ne nous intéresserons pas au concept de transversaux cubiques (CASALI, CICCHETTI et LAKHAL 2003b), qui sont des cas particuliers des transversaux d'un hypergraphe (BERGE 1989 ; EITER et GOTTLOB 1995 ; GUNOPULOS, KHARDON, MANNILA et al. 1997), pour rester concentrés sur les concepts de cube de données.

Deux approches ont concurremment tenu compte de la spécificité des données multidimensionnelles à savoir qu'elles sont structurées. En effet, dans le contexte OLAP, il ne s'agit pas de manipuler des ensembles *d'items* mais des tuples dont chaque valeur a une sémantique particulière en fonction de l'attribut concerné.

Le cube de données fermé (CASALI, CICCHETTI et LAKHAL 2003b) caractérise l'espace de recherche du treillis cube. Tous ses tuples sont des solutions potentielles sémantiquement valides. Cet espace est muni d'une relation d'ordre entre tuples, la généralisation (ou de façon duale la spécialisation), et de plusieurs opérateurs de parcours.

Cet espace de recherche propose un cadre de travail formellement et solidement défini dans les contributions présentées ultérieurement. Pour cette raison, nous le présentons séparément des approches qui l'ont originellement défini.

La première des approches présentées, le cube fermé (CASALI, CICCHETTI et LAKHAL 2003b), en s'inspirant des motifs fermés fréquents (PASQUIER, BASTIDE, TAOUIL et al. 1999 ; ZAKI et C.-J. HSIAO 2002), vise à éliminer les redondances intrinsèquement présentes au sein d'un cube de données. Le résultat est une structure solidement fondée et sans perte d'information. À partir de cette structure, les réponses aux requêtes sur les cubes de données peuvent en être déduites. De plus elle a le mérite d'être, à notre connaissance, l'une des représentations sans perte parmi les plus réduites d'un cube de données. Cette réduction se fait au prix d'une impossibilité de naviguer entre les différents niveaux de granularité des données du cube de données.

2.3.2. Cadre formel du cube de données fermé

2.3.2.1. Treillis cube

Contrairement aux motifs recherchés en fouille de bases de données binaires, les motifs multidimensionnels sont dotés d'une structure et il importe de les caractériser en l'exhibant. D'autre part, les liens qui existent entre de tels motifs véhiculent une

sémantique importante dans un contexte de travail multidimensionnel.

La solution proposée par CASALI, CICCHETTI et LAKHAL 2003a a consisté à introduire la notion d'espaces multidimensionnels dont les éléments représentent les tuples multidimensionnels possibles. Pour formaliser leurs liens, ils s'appuient sur une relation d'ordre et définissent deux opérateurs fondamentaux de construction. Dès lors, les espaces de recherche pour les problèmes de fouille de bases multidimensionnelles peuvent être caractérisés, ce qui est fait en définissant le concept de treillis cube. Dans ce contexte multidimensionnel, le treillis cube est un nouveau langage de description de concepts qui peut être utilisé non seulement en fouille de bases de données multidimensionnelles mais aussi, par exemple, en apprentissage automatique (MITCHELL 1997).

Espaces multidimensionnels Comme dans la section précédente, nous considérons une relation r dont le schéma $\mathscr{R}$ comporte deux types d'attributs : $\mathscr{D}$ l'ensemble des attributs dimensions et $\mathscr{M}$ l'ensemble des attributs mesures (pour l'OLAP). L'espace multidimensionnel de la relation r regroupe toutes les combinaisons sémantiquement valides des ensembles de valeurs existant dans r pour les attributs dimensions, enrichis de la valeur ALL (cf. sous-section 2.2.4 p. 34).

Définition 2.7 (Espace multidimensionnel) **-** L'espace multidimensionnel de r est noté et défini comme suit : $Space(r) = \{\times_{A \in \mathscr{D}}(Dim(A) \cup \{ALL\})\} \cup \{(\emptyset, \ldots, \emptyset)\}$ où $\times$ symbolise le produit cartésien, et $(\emptyset, \ldots, \emptyset)$ la combinaison de valeurs vides. Toute combinaison de $Space(r)$ est un tuple et représente un motif multidimensionnel.

Exemple 2.6 - Considérons à nouveau la relation `Diablo3` utilisée dans la section précédente (cf. tableau 2.1 p. 32). Son espace multidimensionnel est illustré par le tableau 2.6.

Les tuples d'identifiants de valeurs 1, ..., 12 et 37 sont des tuples possibles pour r (car toutes leurs valeurs sont réelles), même si le tuple d'identifiant de valeur 37 n'est pas présent explicitement dans la relation r. Dans ce tuple, le symbole $\emptyset$ signifie « valeur vide ». Les autres tuples (identifiants 13, ..., 36) ne peuvent pas être des tuples de r car ils comportent au moins une valeur ALL. Ils véhiculent donc une information à un niveau de détail plus agrégé que les précédents.

Ordre de généralisation Les tuples véhiculent une information à différents niveaux de granularité et, dans l'espace multidimensionnel, ils sont, par nature même, redondants. En effet, tout tuple de la relation initiale participe généralement à la construction de plusieurs tuples (premier niveau de synthèse de données). Ces derniers peuvent à nouveau être synthétisés et ainsi de suite. Au niveau le plus général, la synthèse consiste en un unique tuple dont toutes les valeurs sont ALL et qui résume la relation initiale r de la manière la plus compacte qui soit, mais aussi la plus grossière.

L'espace multidimensionnel de r est structuré par la relation de généralisation/spécialisation entre tuples. Cet ordre est originellement introduit par MITCHELL 1982 ; MITCHELL 1997 dans le cadre de l'apprentissage de concepts.

TABLEAU 2.6. – Espace multidimensionnel de la relation `Diablo3`

RowId	Type	Propriété	Équipement
1	*Féticheur*	*Vitesse*	*Bastion*
2	*Féticheur*	*Vitesse*	*Marche*
3	*Féticheur*	*Chance*	*Bastion*
4	*Féticheur*	*Chance*	*Marche*
5	*Sorcière*	*Vitesse*	*Marche*
6	*Sorcière*	*Vitesse*	*Bastion*
7	*Sorcière*	*Chance*	*Marche*
8	*Sorcière*	*Chance*	*Bastion*
9	*Croisé*	*Vitesse*	*Marche*
10	*Croisé*	*Vitesse*	*Bastion*
11	*Croisé*	*Chance*	*Marche*
12	*Croisé*	*Chance*	*Bastion*
13	*Féticheur*	*Vitesse*	*ALL*
14	*Féticheur*	*Chance*	*ALL*
15	*Sorcière*	*Chance*	*ALL*
16	*Sorcière*	*Vitesse*	*ALL*
17	*Croisé*	*Chance*	*ALL*
18	*Croisé*	*Vitesse*	*ALL*
19	*Féticheur*	*ALL*	*Bastion*
20	*Féticheur*	*ALL*	*Marche*
21	*Sorcière*	*ALL*	*Bastion*
22	*Sorcière*	*ALL*	*Marche*
23	*Croisé*	*ALL*	*Bastion*
24	*Croisé*	*ALL*	*Marche*
25	*ALL*	*Vitesse*	*Bastion*
26	*ALL*	*Vitesse*	*Marche*
27	*ALL*	*Chance*	*Bastion*
28	*ALL*	*Chance*	*Marche*
29	*Féticheur*	*ALL*	*ALL*
30	*Sorcière*	*ALL*	*ALL*
31	*Croisé*	*ALL*	*ALL*
32	*ALL*	*Vitesse*	*ALL*
33	*ALL*	*Chance*	*ALL*
34	*ALL*	*ALL*	*Bastion*
35	*ALL*	*ALL*	*Marche*
36	*ALL*	*ALL*	*ALL*
37	∅	∅	∅

Définition 2.8 (Généralisation/Spécialisation) - Soit t, t' deux tuples de l'espace multidimensionnel de r, nous définissons la relation d'ordre $\preceq_g$ comme suit :

$$t \preceq_g t' \Leftrightarrow \begin{cases} \forall A \in \mathscr{D} \text{ tel que } t[A] \neq ALL, t[A] = t'[A] \\ \text{où } t' = (\emptyset, \ldots, \emptyset) \end{cases}$$

Si $t \preceq_g t'$, nous disons que t est plus général que t' dans $Space(r)$. La relation de généralisation $\preceq_g$ est un ordre dual à la relation de spécialisation ($\succeq_s$). La relation de couverture de $\preceq_g$ est notée $\prec_g$ et définie comme ceci : $\forall t, t' \in Space(r), t \prec_g t' \Leftrightarrow t \preceq_g t'$ et $\forall t'' \in Space(r), t \preceq_g t'' \preceq_g t' \Rightarrow t = t''$ ou $t' = t''$.

Exemple 2.7 - Dans l'espace multidimensionnel de notre relation exemple (cf. tableau 2.6), en considérant que t_i est le tuple d'identifiant i, nous avons : $t_{13} \preceq_g t_1$ et $t_{29} \preceq_g t_1$. Donc t_{13} et t_{29} sont plus généraux que t_1 et t_1 plus spécifique que t_{13} et t_{29}.

Appliqués sur un ensemble de tuples, les opérateurs min et max rendent respectivement les tuples les plus généraux et les plus spécifiques de l'ensemble considéré.

Définition 2.9 (Opérateurs min/max selon $\preceq_g$) - Soit un ensemble de tuples $T \subseteq Space(r)$:

$$min_{\preceq_g}(T) = \{t \in T \mid \nexists u \in T : u \preceq_g t\}$$
$$max_{\preceq_g}(T) = \{t \in T \mid \nexists u \in T : t \preceq_g u\}$$

En utilisant l'ordre dual de la généralisation (*i.e.* la spécialisation), nous avons : $min_{\succeq_s}(T) = max_{\preceq_g}(T)$ et $max_{\succeq_s}(T) = min_{\preceq_g}(T)$.

Exemple 2.8 - Dans notre espace multidimensionnel exemple (cf. tableau 2.6), soit un ensemble de tuples $T = \{t_1, t_{13}, t_{23}\}$. Nous avons $min(T) = \{t_{13}, t_{23}\}$ et $max(T) = \{t_1\}$.

Opérateurs de base Nous présentons les deux opérateurs de construction de tuples : Somme (notée +) et Produit (noté $\bullet$). Le Semi-Produit (noté $\odot$) est une variante contrainte de l'opérateur produit.

La somme de deux tuples rend le tuple le plus spécifique généralisant les deux opérandes. Elle est définie comme suit.

Définition 2.10 (Opérateur Somme) - Soit u et v deux tuples de $Space(r)$, $t = u + v \Leftrightarrow \forall A \in \mathscr{D}$,

$$t[A] = \begin{cases} u[A] \text{ si } u[A] = v[A] \\ ALL \text{ sinon.} \end{cases}$$

t est nommé la somme des tuples u et v.

Exemple 2.9 - Dans l'espace multidimensionnel exemple (cf. tableau 2.6), nous avons $t_1 + t_2 = t_{13}$, ce qui signifie que t_{13} est construit à partir des tuples t_1 et t_2. D'autre part $(t_{13} + t_{14}) + (t_{19} + t_{20}) = t_{13} + t_{14} = t_{19} + t_{20} = t_{29}$.

Le produit de deux tuples retourne le tuple le plus général spécialisant les deux opérandes. Si pour ces deux tuples, il existe un attribut A prenant des valeurs distinctes et réelles (*i.e.* existant dans la relation initiale), alors seul le tuple $(\emptyset,\ldots,\emptyset)$ les spécialise (hormis ce tuple, les ensembles permettant de les construire sont disjoints).

Définition 2.11 (Opérateur Produit) - Soit u et v deux tuples de $Space(r)$, $t = u \bullet v \Leftrightarrow$

$$\begin{cases} t = (\emptyset,\ldots,\emptyset) \text{ si } \exists A \in \mathscr{D} \text{ tel que } u[A] \neq v[A] \neq ALL, \\ \text{sinon } \forall A \in \mathscr{D} \begin{cases} t[A] = u[A] \text{ si } v[A] = ALL \\ t[A] = v[A] \text{ si } u[A] = ALL. \end{cases} \end{cases}$$

t est dit le produit des tuples u et v.

Exemple 2.10 - Dans l'espace multidimensionnel pris en exemple (cf. tableau 2.6), nous avons $t_{13} \bullet t_{25} = t_1$ et $t_1 \bullet t_2 = (\emptyset,\emptyset,\emptyset)$, ce qui signifie que t_{13} et t_{25} généralisent t_1 et t_1 participe à la construction de t_{13} et t_{25} (directement ou indirectement). Les tuples t_1 et t_2 n'ont d'autre point commun que le tuple vide.

L'opérateur semi-produit a été proposé pour disposer d'une base rigoureuse lors de l'étape de génération des candidats dans un algorithme par niveau. Avant de présenter cet opérateur, la définition de la fonction $Attribute$ est rappelée. Pour un tuple t, elle rend l'ensemble des attributs dont la valeur est différente de ALL.

Définition 2.12 (Fonction $Attribute$) - Soit t un tuple de $Space(r)$:

$$Attribute(t) = \{A \in \mathscr{D} \mid t[A] \neq ALL\}$$

Exemple 2.11 - Dans $Space(\texttt{Diablo3})$ (cf. tableau 2.6), nous avons $Attribute(t_1) = \{Féticheur, Vitesse, Bastion\}$ et $Attribute(t_{13}) = \{Féticheur, Vitesse\}$.

L'opérateur semi-produit est un produit contraint utilisé pour la génération de candidats dans une approche utilisant des algorithmes par niveau (Agrawal, Mannila, Srikant et al. 1996 ; Mannila et Toivonen 1997). À tout niveau i, il permet de ne générer que des candidats de niveau $i+1$, s'ils existent (sinon le semi-produit retourne le tuple $(\emptyset,\ldots,\emptyset)$). De plus, chaque tuple de niveau $i+1$ n'est généré qu'une seule fois. Plus précisément, pour tout couple de tuples t, u de l'espace, les ensembles d'attributs (ordonnés selon $<_{\mathscr{D}}$) pour lesquels t et u admettent des valeurs réelles (différentes de ALL) sont considérés. Si ces ensembles ne diffèrent que par leur dernier élément alors le semi-produit $t \odot u$ génère un nouveau tuple plus agrégé que t et u. La condition de précédence selon $<_{\mathscr{D}}$ entre les derniers éléments des deux ensembles d'attributs est inspirée d'Apriori-Gen.

Définition 2.13 (Opérateur Semi-produit) - Soit (u, v) un couple de tuples de $Space(r)$

et deux ensembles $X = Attribute(u)$ et $Y = Attribute(v)$.

$$t = u \odot v \Leftrightarrow t = \begin{cases} u \bullet v \text{ si } X \setminus max_{<_{\mathscr{D}}}(X) = Y \setminus max_{<_{\mathscr{D}}}(Y) \\ \text{ et } max_{<_{\mathscr{D}}}(X) <_{\mathscr{D}} max_{<_{\mathscr{D}}}(Y) \\ (\emptyset, \ldots, \emptyset) \text{ sinon.} \end{cases}$$

où $<_{\mathscr{D}}$ est un ordre total sur $\mathscr{D}$.

Exemple 2.12 - Dans $Space(\texttt{Diablo3})$, nous avons $t_{13} \odot t_{19} = t_1$.

Définition 2.14 (Opérateur Différence) **-** La différence de deux tuples u et v de $Space(r)$ est définie comme suit :

$$t = u \setminus v \Leftrightarrow \forall A \in \mathscr{D}, t[A] = \begin{cases} u[A] \text{ si } u[A] \neq v[A] \\ ALL \text{ sinon.} \end{cases}$$

Exemple 2.13 - Dans $Space(\texttt{Diablo3})$, nous avons $t_1 \setminus t_{19} = t_{32}$.

Caractérisation du treillis cube En dotant l'espace multidimensionnel $Space(r)$ de la relation de généralisation entre tuples et en utilisant les opérateurs produit et somme, CASALI, CICCHETTI et LAKHAL 2003a ont proposé une structure algébrique appelée treillis cube qui fixe un cadre théorique et général pour la fouille de bases de données multidimensionnelles. Les différents lemmes et la proposition suivants donnent les propriétés fondamentales du treillis cube, qui sont reprises dans le théorème 2.5.

Lemme 2.1 - L'ensemble ordonné $CL(r) = \langle Space(r), \preceq_g \rangle$ est un treillis complet appelé treillis cube pour lequel :
— $\forall T \subseteq CL(r), \bigwedge T = +_{t \in T} t$ où $\bigwedge$ symbolise *l'infimum.*
— $\forall T \subseteq CL(r), \bigvee T = \bullet_{t \in T} t$ où $\bigvee$ symbolise le *supremum.*

Rappels Les co-atomes (respectivement les atomes) sont les tuples maximaux, c'est à dire les tuples les plus spécifiques (respectivement les tuples minimaux) du treillis privé de son majorant (respectivement minorant) universel. Les co-atomes (respectivement les atomes) du treillis cube d'une relation r sont notés $\mathscr{CA}t(CL(r))$ (respectivement $\mathscr{A}t(CL(r))$).

Lemme 2.2 - Le treillis $CL(r) = \langle Space(r), \preceq_g \rangle$ est un treillis co-atomique et atomique.

Dans le treillis des parties d'un ensemble $\mathscr{I}$, chaque élément peut être généré avec soit :
— les co-atomes (motifs de cardinalité $|\mathscr{I}| - 1$) et l'opérateur infimum ($\cap$) ;
— les atomes (motifs de cardinalité 1) et l'opérateur supremum ($\cup$).
De manière similaire, dans notre contexte, chaque tuple du treillis cube peut être généré avec :
— les atomes du treillis cube et l'opérateur *supremum* ($\bullet$) ;

— les co-atomes du treillis cube et l'opérateur *infimum* (+).

En conséquence, les atomes et les co-atomes du treillis cube sont deux familles génératrices de celui-ci.

La proposition ci-dessous permet à la fois de définir un plongement d'ordre du treillis cube vers le treillis des parties des attributs binaires ainsi qu'une fonction rang. Cette dernière est utilisée pour donner une autre caractérisation des atomes et co-atomes du treillis cube et pour montrer que le treillis cube est un treillis non distributif dans la majeure partie des cas et gradué (cf. lemme 2.4). Pour éviter toute ambiguïté, chaque valeur est préfixée par le nom de l'attribut concerné.

Proposition 2.3 (Treillis des parties des attributs binaires) **-** Soit $\mathscr{L}(r)$ le treillis des parties des attributs binaires de la relation binaire, *i.e.* le treillis $\langle \mathscr{P}(\bigcup_{A\in\mathscr{D}} A.a, \forall a \in Dim(A)), \subseteq\rangle$. Alors il existe un plongement d'ordre Φ :
$CL(r) \rightarrow \mathscr{L}(r)$

$$t \mapsto \begin{cases} \bigcup_{A\in\mathscr{D}} A.a, \forall a \in Dim(A) \text{ si } t = (\emptyset,\ldots,\emptyset) \\ \{d_i.t[A] \mid \forall A \in Attribute(t)\} \text{ sinon.} \end{cases}$$

Le rang d'un tuple t, noté $rank(t)$, est la longueur du plus petit chemin (nombre d'arcs minimal) dans le treillis cube le reliant au tuple $(ALL,\ldots,ALL)$. Nous avons donc : $rank(t) = |\Phi(t)|$ si $t \neq (\emptyset,\ldots,\emptyset), |\mathscr{D}| + 1$ sinon.

Cette proposition permet une autre caractérisation des atomes et des co-atomes du treillis cube. En effet, les atomes du treillis cube sont les tuples tels que $|\Phi(t)| = 1$ et les co-atomes du treillis cube sont les tuples tels que $|\Phi(t)| = |\mathscr{D} - 1|$.

Le lemme suivant montre que le treillis cube est un treillis gradué. En conséquence, nous pouvons appliquer des algorithmes par niveau sur cet espace de recherche.

Lemme 2.4 - Le treillis cube $CL(r)$ est gradué. Si $|\mathscr{D}| \leq 2$ alors $CL(r)$ n'est pas distributif.

Théorème 2.5 (Treillis cube de données) **-** Soit r une relation d'attributs catégories sur $\mathscr{D} \cup \mathscr{M}$. L'ensemble ordonné $CL(r) = \langle Space(r), \preceq_g\rangle$ est un treillis complet, atomique, co-atomique et gradué appelé treillis cube dans lequel :

— $\forall T \subseteq CL(r), \bigwedge T = +_{t\in T} t$

— $\forall T \subseteq CL(r), \bigvee T = \bullet_{t\in T} t$.

Exemple 2.14 - La figure 2.10 représente le treillis cube de notre relation exemple (cf. tableau 2.1 p. 32). Dans ce diagramme, les arêtes représentent les liens de généralisation ou spécialisation entre tuples. Les valeurs des différents attributs sont codés comme suit :

Type		Propriété		Équipement	
Féticheur	$= F$	*Vitesse*	$= V$	*Bastion*	$= B$
Sorcière	$= S$	*Chance*	$= C$	*Marche*	$= M$
Croisé	$= C$	*ALL*	$= *$	*ALL*	$= *$
ALL	$= *$				

En faisant abstraction de $(\emptyset, \emptyset, \emptyset)$, les co-atomes sont des tuples véhiculant l'information au niveau le plus détaillé, *i.e.* celui des valeurs réelles des dimensions. En d'autres termes, les co-atomes sont les tuples potentiels d'une relation. Ainsi, nous avons :

$$\mathscr{CA}t(CL(\texttt{Diablo3})) = \left\{ \begin{array}{l} (Féticheur, Vitesse, Bastion), \\ (Féticheur, Vitesse, Marche), \\ (Féticheur, Chance, Bastion), \\ (Féticheur, Chance, Marche), \\ (Sorcière, Vitesse, Marche), \\ (Sorcière, Vitesse, Bastion), \\ (Sorcière, Chance, Marche), \\ (Sorcière, Chance, Bastion), \\ (Croisé, Vitesse, Marche), \\ (Croisé, Vitesse, Bastion), \\ (Croisé, Chance, Marche), \\ (Croisé, Chance, Bastion) \end{array} \right\}$$

Les atomes du treillis offrent l'information la plus synthétique possible, à l'exception du tuple (ALL, ALL, ALL). Étant donné que nous ne considérons ici que trois attributs dimensions, tous les atomes comportent deux fois la valeur synthétique ALL. Nous avons donc :

$$\mathscr{A}t(CL(r)) = \left\{ \begin{array}{l} (Féticheur, ALL, ALL), \\ (Sorcière, ALL, ALL), \\ (Croisé, ALL, ALL), \\ (ALL, Chance, ALL), \\ (ALL, Vitesse, ALL), \\ (ALL, ALL, Marche), \\ (ALL, ALL, Bastion), \end{array} \right\}$$

La propriété suivante donne une analyse du nombre d'éléments contenus dans un niveau du treillis cube ainsi que le nombre d'éléments total. Cette propriété est particulièrement importante car elle permet de caractériser la complexité des algorithmes utilisant le treillis cube comme espace de recherche.

Proposition 2.6 (Taille d'un treillis cube) **-** La hauteur (nombre de niveaux) du treillis cube est $|\mathscr{D}| + 1$. Le nombre d'éléments pour un niveau $i (i \in 1..|\mathscr{D}|)$ est :

$$\sum_{\substack{X \subseteq \mathscr{D} \\ |X|=i}} \left(\prod_{A \in X} |Dim(A)| \right) \leq \binom{|\mathscr{D}|}{i} \max_{A \in \mathscr{D}} (|Dim(A)|)^i.$$

Le nombre total d'éléments dans le treillis cube est :

$$\sum_{i=1..|\mathscr{D}|} \left(\sum_{\substack{X \subseteq \mathscr{D} \\ |X|=i}} \left(\prod_{A \in X} |Dim(A)| \right) \right) + 2 = \left(\prod_{A \in \mathscr{D}} (|Dim(A)| + 1) \right) + 1$$

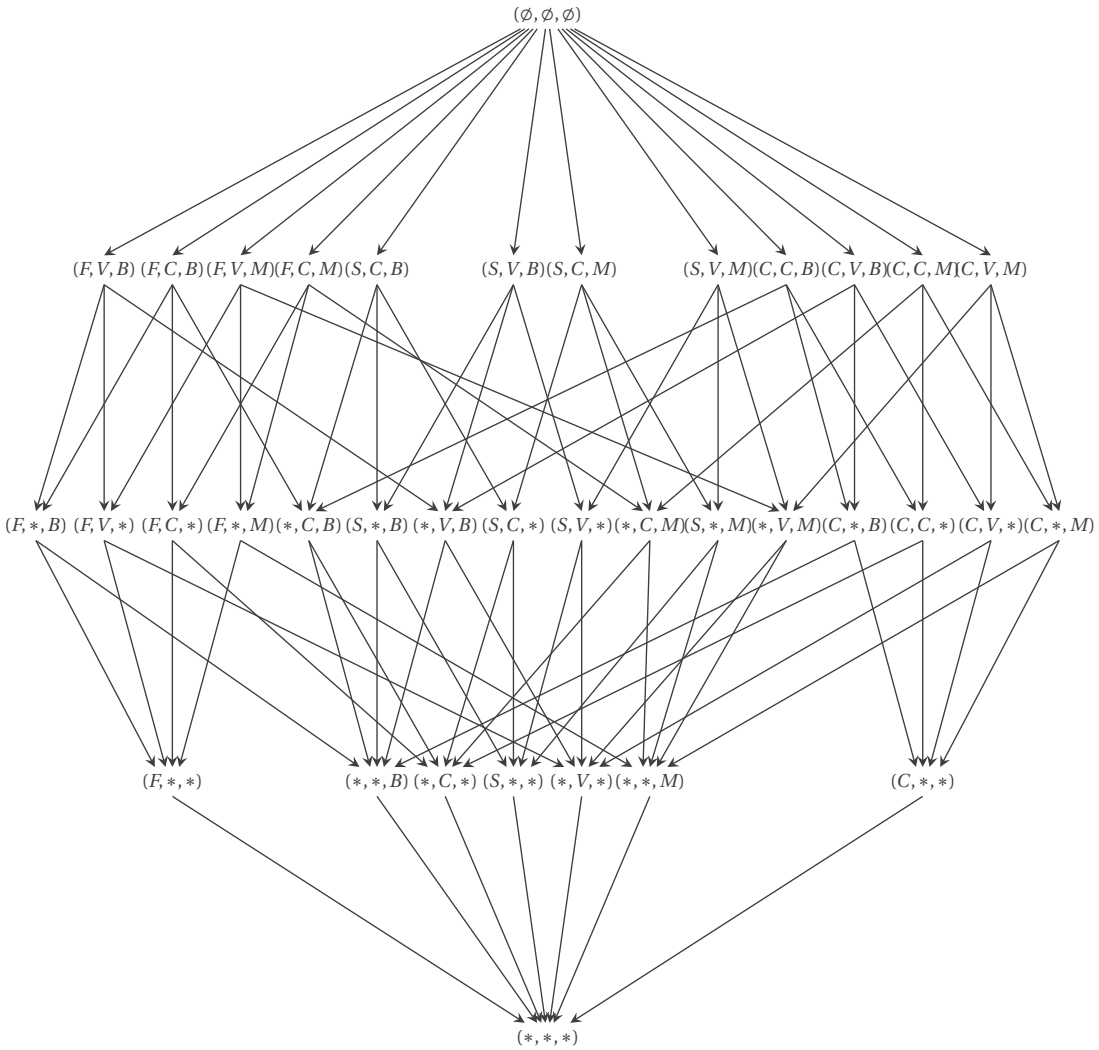


FIGURE 2.10. – Diagramme de Hasse du treillis cube de la relation `Diablo3`

2.3.2.2. Treillis cube convexe

Dans ce paragraphe, la structure du treillis cube en présence de conjonctions de contraintes monotones et/ou antimonotones est présentée. Munies d'une telle structure, les représentations classiques condensées (avec bordures) du treillis cube contraint sont décrites avec un double objectif : définir d'une manière compacte l'espace de solutions et décider si un tuple t appartient, ou pas, à cet espace. Nous prenons en compte les contraintes monotones et/ou antimonotones les plus couramment utilisées en fouille de base de données (NG, LAKSHMANAN, HAN et al. 1998 ; JR., AGRAWAL et GUNOPULOS 2000 ; GRAHNE, LAKSHMANAN et X. WANG 2000 ; LOPES, PETIT et LAKHAL 2002). Celles-ci peuvent porter sur :

— des mesures d'intérêts comme la fréquence de motifs, la confiance, la corrélation (HAN, KAMBER et PEI 2011) : dans ce cas, seuls les attributs dimensions de $\mathscr{R}$ sont nécessaires ;

— des agrégats selon des attributs mesures $\mathscr{M}$ calculés en utilisant des fonctions mathématiques agrégatives (`COUNT`, `SUM`, `MIN`, `MAX`).

Nous rappelons les définitions des contraintes monotones et antimonotones selon l'ordre de généralisation $\preceq_g$.

Définition 2.15 (Contraintes monotones/antimonotones) **-** Une contrainte $Cont$ est dite :

— monotone si et seulement si :

$$\forall\ t, t' \in CL(r) : [t \preceq_g t' \text{ et } Cont(t)] \Rightarrow Cont(t')$$

— antimonotone si et seulement si :

$$\forall\ t, t' \in CL(r) : [t \preceq_g t' \text{ et } Cont(t')] \Rightarrow Cont(t)$$

Exemple 2.15 - Dans l'espace multidimensionnel de la relation `Diablo3` (cf. tableau 2.1 p. 32), une requête porte sur tous les tuples dont la somme des valeurs pour l'attribut mesure `Quantité` est supérieure ou égale à 300. La contrainte `SUM(Quantité)` $\geq$ 300 est une contrainte antimonotone. Si la `Quantité` de personnages par `Type`, `Propriété` et `Équipement` est supérieure à 300, alors elle l'est *a fortiori* pour un niveau plus agrégé de granularité, *e.g.* par `Type` et par `Propriété` (quelque soit `Équipement`). Inversement, si la requête porte sur tous les tuples dont la somme des valeurs pour l'attribut `Quantité` est inférieure ou égale à 700, la contrainte exprimée `SUM(Quantité)` $\leq$ 700 est monotone.

Structure du treillis cube convexe Le treillis cube, en présence de contraintes monotones et/ou antimonotones, n'est plus nécessairement un treillis. Il est rappelé dans ce paragraphe qu'un tel ensemble ordonné est un espace convexe et donc représentable par des bordures.

Définition 2.16 (Espace Convexe) **-** Soit $(\mathscr{P}, \leq)$ un ensemble partiellement ordonné,

$\mathscr{C} \subseteq \mathscr{P}$ est un espace convexe (VEL 1993) si et seulement si :

$$\forall x, y, z \in \mathscr{P} \text{ tels que } x \leq y \leq z \text{ et } x, z \in \mathscr{C} \Rightarrow y \in \mathscr{C}.$$

Donc $\mathscr{C}$ est borné par deux ensembles :
— un majorant (ou *upper set*), noté U, défini par $U = \max_{\leq}(\mathscr{C})$;
— un minorant (ou *lower set*) noté L et défini par $L = \min_{\leq}(\mathscr{C})$.

Notations cmc (respectivement camc) est utilisé pour noter une conjonction de contraintes monotones (respectivement antimonotones) et chc une conjonction de contraintes hybrides (monotones et antimonotones). En reprenant les symboles U et L classiques et suivant les cas considérés, les bornes introduites sont indicées par le type de contrainte considéré. Par exemple U_{camc} symbolise l'ensemble des tuples les plus spécifiques vérifiant une conjonction de contraintes antimonotones.

Remarques Nous considérons les cas extrêmes suivants :
— supposons par la suite que le tuple $(ALL, \ldots, ALL)$ vérifie toujours la conjonction de contraintes antimonotones et que le tuple $(\emptyset, \ldots, \emptyset)$ vérifie toujours la conjonction de contraintes monotones. Avec ces hypothèses, l'espace des solutions contient toujours au moins un élément ;
— de plus, supposons que le tuple $(ALL, \ldots, ALL)$ ne vérifie jamais la conjonction de contraintes monotones et que le tuple $(\emptyset, \ldots, \emptyset)$ ne vérifie jamais la conjonction de contraintes antimonotones, car sinon l'espace de solutions est $Space(r)$.

Théorème 2.7 - Tout treillis cube avec contraintes monotones et/ou antimonotones est un espace convexe. Son majorant U_{const} et son minorant L_{const} sont :

$$\text{si } const = cmc, \begin{cases} L_{cmc} = \min_{\leq_g}(\{t \in CL(r) \mid cmc(t)\}) \\ U_{cmc} = (\emptyset, \ldots, \emptyset) \end{cases} \tag{2.1}$$

$$\text{si } const = camc, \begin{cases} L_{camc} = (ALL, \ldots, ALL) \\ U_{camc} = \max_{\leq_g}(\{t \in CL(r) \mid camc(t)\}) \end{cases} \tag{2.2}$$

$$\text{si } const = chc, \begin{cases} L_{chc} = \min_{\leq_g}(\{t \in CL(r) \mid chc(t)\}) \\ U_{chc} = \max_{\leq_g}(\{t \in CL(r) \mid chc(t)\}) \end{cases} \tag{2.3}$$

Le majorant U_{const} représente les tuples les plus spécifiques satisfaisant la conjonction de contraintes et le minorant L_{const} les tuples les plus généraux la satisfaisant aussi. Donc U_{const} et L_{const} permettent d'obtenir des représentations condensées du treillis cube en présence d'une conjonction de contraintes monotones et/ou antimonotones.

Le corollaire suivant permet la caractérisation des bordures du treillis cube contraint par une conjonction de contraintes hybrides $chc = camc \wedge cmc$ en ne connaissant que :
— soit la bordure maximale pour la contrainte antimonotone U_{camc} et la contrainte monotone cmc ;

— soit la bordure minimale pour la contrainte monotone L_{cmc} et la contrainte antimonotone camc.

Corollaire 2.8 - Considérons les deux propositions suivantes :

— étant données U_{camc} et cmc, une représentation condensée de l'ensemble des tuples satisfaisant la contrainte hybride $chc = cmc \wedge camc$ est :

$$\begin{cases} L_{chc} = min_{\preceq_g}(\{t \in CL(r) \mid \exists t' \in U_{camc} : t \preceq_g t' \text{ et } cmc(t)\}) \\ U_{chc} = \{t \in U_{camc} \mid \exists t' \in L_{chc} : t' \preceq_g t\} \end{cases}$$

— étant données L_{cmc} et camc, une représentation condensée de $CL(r)_{chc}$ est :

$$\begin{cases} U_{chc} = max_{\preceq_g}(\{t \in CL(r) \mid \exists t' \in L_{cmc} : t' \preceq_g t \text{ et } camc(t)\}) \\ L_{chc} = \{t \in L_{camc} \mid \exists t' \in U_{chc} : t \preceq_g t'\}. \end{cases}$$

TABLEAU 2.7. – Bornes du treillis cube contraint pour 300 ≤ SUM(Quantité) ≤ 700

U_{camc}	$(Féticheur, Vitesse, Bastion)$ $(Sorcière, Vitesse, Bastion)$ $(ALL, Chance, Bastion)$
U_{chc}	$(Féticheur, Vitesse, Bastion)$ $(Sorcière, Vitesse, Bastion)$ $(ALL, Chance, Bastion)$
L_{cmc}	$(ALL, Vitesse, Bastion)$ $(Féticheur, ALL, ALL)$ $(Sorcière, ALL, ALL)$ $(ALL, ALL, Marche)$ $(ALL, Chance, ALL)$
L_{chc}	$(ALL, Vitesse, Bastion)$ $(Féticheur, ALL, ALL)$ $(Sorcière, ALL, ALL)$ $(ALL, Chance, ALL)$

Exemple 2.16 - Le tableau 2.7 donne les bornes U_{camc}, U_{chc} L_{cmc} et L_{chc} du treillis cube convexe de la relation exemple en considérant la contrainte hybride 300 ≤ SUM(Quantité) ≤ 700.

De même que la caractérisation de l'ensemble des motifs fréquents comme étant un espace convexe nous permet de savoir, en ne connaissant que la bordure positive, si un motif est fréquent ou non, la représentation par bordure du treillis cube contraint de la relation Diablo3 permet de répondre directement à certaines requêtes comme par exemple :

— Est-ce que le nombre de personnages avec $Vitesse$ et $Bastion$ est supérieur à

300 ?
  — Oui, car le tuple $(ALL, Vitesse, Bastion)$, qui donnent les personnages avec comme `Propriété` $Vitesse$ et comme `Équipement` $Bastion$ pour n'importe quel `Type`, appartient à la bordure L_{cmc}.
— Est-ce que le nombre de personnages de `Type` $Sorcière$ avec $Vitesse$ est compris entre 300 et 700 ?
  — Oui, car $(Sorcière, Vitesse, ALL)$ spécialise $(Sorcière, ALL, ALL)$ appartenant à L_{chc} et généralise $(Sorcière, Vitesse, Bastion)$ appartenant à la bordure U_{chc}.

2.3.2.3. Treillis cube et treillis des parties

À notre connaissance, le treillis cube est la première approche visant à établir un fondement algébrique pour la fouille de bases de données multidimensionnelles. Précédemment, des chercheurs ont tenté d'étendre au contexte multidimensionnel, des solutions éprouvées dans un cadre de bases de données binaires, en utilisant le treillis des parties comme espace de recherche notamment pour les règles d'associations quantitatives (Srikant et Agrawal 1996) et la classification (Ganascia 1993 ; B. Liu, Hsu et Ma 1998 ; Dong et J. Li 2005 ; W. Li, Han et Pei 2001). Dans Mannila et Toivonen 1997, une approche par langage (langage de motifs) est introduite comme un modèle théorique général pour la fouille de données. Malheureusement, cette tentative de formalisation n'est pas la panacée car la propriété fondamentale de graduation, qui est nécessaire pour les algorithmes par niveau et la structure convexe, n'est pas toujours vérifiée. C'est pourquoi il apparaît qu'un cadre de travail spécifique, treillis des parties pour les bases de données binaires et treillis cube pour les bases de données multidimensionnelles, est indispensable.

Les auteurs du treillis cube (Casali, Cicchetti et Lakhal 2003a) ont montré la nécessité d'un fondement pour la fouille de bases multidimensionnelles qui ne soit pas celui de la fouille de bases binaires :

— un espace de recherche dont les éléments ne sont pas structurés est incomparablement plus volumineux qu'un espace dans lequel ils le sont. Compte tenu des volumes mis en jeu dans les entrepôts et en fouille de données, cette raison peut, à elle seule, suffire à convaincre.
— un espace de recherche dont les éléments ne sont pas structurés englobe un ensemble de combinaisons sémantiquement « impossibles » mais pour lesquelles des calculs doivent être effectués. Le nombre de ces combinaisons est exponentiel dans la cardinalité des dimensions (dont le domaine est réputé très large : Ross et Srivastava 1997 ; Beyer et Ramakrishnan 1999 ; Han, Kamber et Pei 2011). Le coût du calcul évoqué est donc très important. De plus :
  — dans le meilleur des cas, ces combinaisons sont élaguées de l'espace des solutions mais ajoutées à la bordure négative. Les résultats sont donc corrects, mais avec dégradation des performances ;
  — dans le pire des cas, ces combinaisons font partie de l'espace de solution du problème.

2.3.3. Définition formelle du cube de données fermé

Lors du calcul de cube de données, l'explosion combinatoire des résultats est un phénomène bien connu (HAN, KAMBER et PEI 2011). L'approche par cube fermé, présentée ici, vise à représenter un cube de données sans perte d'information et avec une réduction notable de l'espace de stockage nécessaire. L'idée sous-jacente est d'éliminer les redondances éventuelles en ne conservant qu'un représentant pour un ensemble de tuples issus des mêmes données de la relation d'origine. Cette représentation s'appuie sur le concept de connexion cubique. Dans le même esprit que la connexion de Galois (GANTER et WILLE 1999), la connexion cubique est un couple de fonctions (λ, σ), tel que λ est une application définie du treillis cube de r vers le treillis des parties de $Tid(r)$ (ensemble des identifiants des tuples de r) et σ est la fonction duale à λ. Finalement, un opérateur de fermeture de $CL(r)$ sur r est défini en utilisant la connexion cubique.

Définition 2.17 (Connexion cubique) - Soit $\texttt{RowId} : r \to \mathbb{N}^*$ une fonction qui associe à chaque tuple un unique entier positif et $Tid(r) = \{\texttt{RowId}(t) \mid t \in r\}$. Soit λ et σ deux fonctions définies comme suit :

$$\begin{array}{rrcl} \lambda : & CL(r) & \longrightarrow & \langle \mathscr{P}(Tid(r)), \subseteq \rangle \\ & t & \longmapsto & \bigcup \{Rowid(t') \in Tid(r) \mid t \preceq_g t' \text{ et } t' \in r\} \\ \sigma : \langle \mathscr{P}(Tid(r)), \subseteq \rangle & & \longrightarrow & CL(r) \\ & P & \longmapsto & +\{t \in r \mid \texttt{RowId}(t) \in P\} \end{array}$$

La fonction λ associe à tout tuple du treillis cube l'ensemble des identifiants des tuples qui le généralisent. La fonction σ quant à elle s'applique sur un ensemble d'identifiants de tuples et retourne la somme (cf. définition 2.10) de tous les tuples dotés de ces identifiants. En d'autres termes, elle renvoie le tuple le plus spécifique généralisant tous les tuples pourvus de ces identifiants.

Proposition 2.9 (Connexion cubique) - La connexion cubique $rc = (\lambda, \sigma)$ est une connexion de Galois entre le treillis cube de r et le treillis des parties de $Tid(r)$.

La connexion cubique étant un cas particulier de la connexion de Galois, la composition des deux fonctions, $\sigma \circ \lambda$, est un opérateur de fermeture dont une définition équivalente est donnée ci-dessous.

Définition 2.18 (Opérateur de fermeture cubique) - L'opérateur de fermeture cubique associe à tout tuple t du treillis cube un unique tuple appelé la fermeture de t. Il est obtenu en considérant l'ensemble des tuples plus spécifiques que t et en déterminant le tuple le plus général de cet ensemble grâce à l'opérateur de somme. Cet opérateur est noté $\mathbb{C}$ et défini comme suit :

$$\begin{array}{rcl} \mathbb{C} & : & CL(r) \to CL(r) \\ t & \mapsto & \begin{cases} +t' \mid t \preceq_g t' \text{ et } t' \in r \\ (\emptyset, \ldots, \emptyset) \text{ sinon.} \end{cases} \end{array}$$

Exemple 2.17 - En considérant l'espace multidimensionnel de la relation `Diablo3`, nous avons :

$$\begin{aligned}\mathbb{C}((Féticheur, ALL, ALL)) &= (Féticheur, Vitesse, Bastion)\\ &\quad + (Féticheur, Vitesse, Marche)\\ &= (Féticheur, Vitesse, ALL)\\ \mathbb{C}((Croisé, ALL, ALL)) &= (Croisé, Chance, Bastion)\end{aligned}$$

Corollaire 2.10 - $\mathbb{C}$ est un opérateur de fermeture de $CL(r)$ sur r. $\mathbb{C}$ satisfait les propriétés suivantes :

— $t \preceq_g t' \Rightarrow \mathbb{C}(t,r) \preceq_g \mathbb{C}(t',r)$ (isotonie) ;
— $t \preceq_g \mathbb{C}(t,r)$ (extensivité) ;
— $\mathbb{C}(t,r) = \mathbb{C}(\mathbb{C}(t,r),r)$ (idempotence).

Le système de fermeture contient l'ensemble des fermetures de tous les tuples du treillis cube. Chacun d'eux est nommé tuple fermé cubique. La propriété d'extensivité garantit que tout fermé cubique est plus général que les tuples qui l'ont généré. De plus, il est égal à sa propre fermeture de par la propriété d'idempotence. L'isotonie assure le respect de l'ordre des fermés par rapport à l'ordre des tuples qui les ont générés.

Définition 2.19 (Système de fermeture cubique) **-** Soit $\mathbb{C}(r) = \{t \in CL(r) \mid \mathbb{C}(t,r) = t\}$. $\mathbb{C}(r)$ est un système de fermeture sur r et l'opérateur de fermeture associé est $\mathbb{C}$. Tout tuple appartenant à $\mathbb{C}(r)$ est un tuple fermé ou un fermé cubique.

Exemple 2.18 - En considérant l'espace multidimensionnel de la relation exemple, nous avons :

$$\mathbb{C}(r) = \left\{ \begin{array}{l} (Féticheur, Vitesse, Bastion),\\ (Féticheur, Vitesse, Marche),\\ (Sorcière, Vitesse, Bastion),\\ (Sorcière, Chance, Marche),\\ (Croisé, Chance, Bastion),\\ (ALL, Vitesse, Bastion),\\ (ALL, ALL, Bastion),\\ (Féticheur, Vitesse, ALL),\\ (Sorcière, ALL, ALL),\\ (ALL, Vitesse, ALL),\\ (ALL, Chance, ALL),\\ (ALL, ALL, Marche),\\ (\emptyset, \ldots, \emptyset) \end{array} \right\}$$

Pour chaque tuple fermé cubique t, tous les tuples de $CL(r)$ dont la fermeture est t sont appelés les générateurs de t. Les tuples générateurs minimaux de t selon $\preceq_g$ sont les clefs cubiques de ce tuple.

Définition 2.20 (Clef cubique) **-** Soit t, un tuple fermé. $Key(t) = \min_{\preceq_g}(\{t' \in CL(r) \mid$

$t' \preceq_g t$ et $\mathbb{C}(t', r) = t\}$) est l'ensemble de générateurs minimaux de t. Chaque tuple de $Key(t)$ est une clef cubique.

Exemple 2.19 - En considérant l'espace multidimensionnel de la relation `Diablo3`, nous avons :

$$Key((Fétic heur, ALL, ALL)) = (Fétic heur, ALL, ALL)$$
$$Key((Croisé, Chance, Bastion)) = (Croisé, ALL, ALL)$$

La figure 2.11 représente l'ensemble de toutes les clefs cette relation. Les valeurs des différents attributs sont codés comme suit :

Type		Propriété		Équipement	
Fétic heur	$= F$	*Vitesse*	$= V$	*Bastion*	$= B$
Sorcière	$= S$	*Chance*	$= C$	*Marche*	$= M$
Croisé	$= C$	*ALL*	$= *$	*ALL*	$= *$
ALL	$= *$				

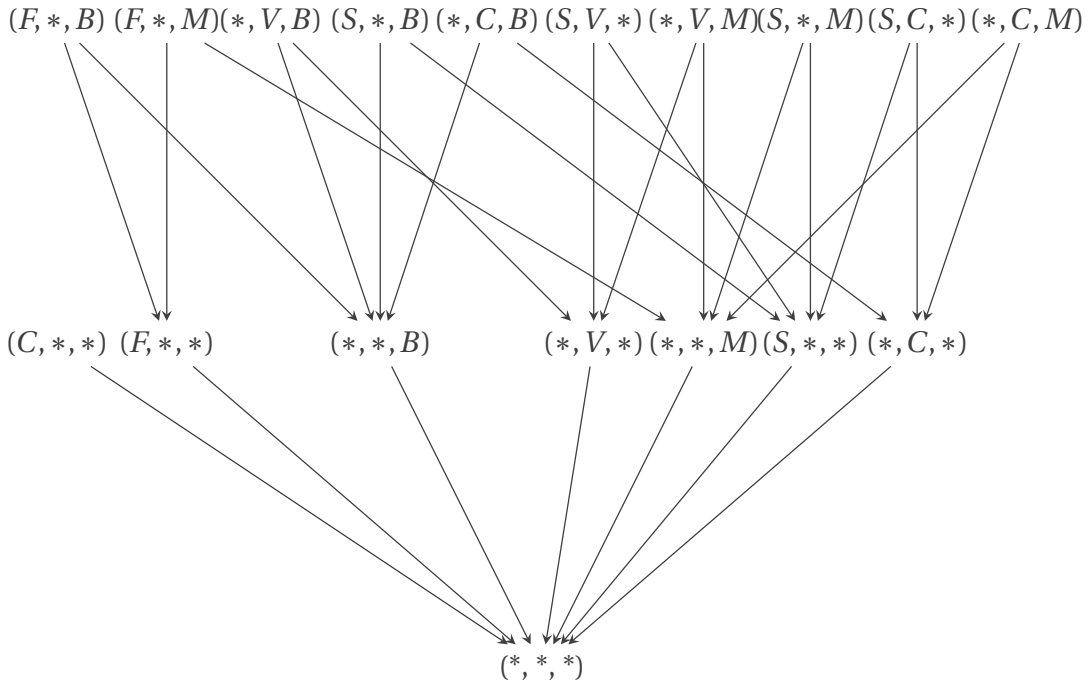


FIGURE 2.11. – Représentation de l'ensemble des clefs de la relation `Diablo3`

Le lemme suivant caractérise le nombre maximal de clefs cubiques que peut avoir un tuple fermé. Cependant, contrairement au modèle binaire, le nombre de clefs n'est plus exponentiel en fonction du nombre de valeurs, mais est exponentiel en fonction du nombre d'attributs dimensions.

Lemme 2.11 - Soit t un tuple fermé, le nombre maximal de clefs cubiques pour t est $\binom{|\Phi(t)|}{|\Phi(t)|/2}$.

En appliquant le théorème de Birkhoff (BIRKHOFF 1970) sur le treillis cube avec l'opérateur $\mathbb{C}$ comme opérateur de fermeture, il est possible de construire un treillis regroupant tous les fermés cubiques, l'ordre de généralisation étant la relation d'ordre de ce treillis. Cependant, contrairement aux autres treillis des fermés (GANTER et WILLE 1999) ce treillis n'est pas quelconque : il est co-atomique.

Théorème 2.12 - L'ensemble partiellement ordonné $CCL(r) = \langle \mathbb{C}(r), \preceq_g \rangle$ est un treillis complet et co-atomique appelé treillis cube fermé tel que :

— $\forall T \subseteq CCL(r), \bigwedge T = +_{t \in T} t$

— $\forall T \subseteq CCL(r), \bigvee T = \mathbb{C}(\bullet_{t \in T} t, r)$

Tous les tuples ayant la même fermeture généralisent les mêmes tuples de la relation originelle. Comme ils généralisent les mêmes tuples originaux, ils partagent la même valeur agrégée de la mesure (propriété du `GROUP BY`). Pour tout tuple du treillis cube, il suffit de calculer sa fermeture pour retrouver sa mesure. Ainsi le cube fermé est une couverture du cube de données.

Exemple 2.20 - La figure 2.12 représente le treillis cube fermé de la relation `Diablo3`. Les valeurs des différents attributs sont codés comme suit :

`Type`		`Propriété`		`Équipement`	
Féticheur	$= F$	*Vitesse*	$= V$	*Bastion*	$= B$
Sorcière	$= S$	*Chance*	$= C$	*Marche*	$= M$
Croisé	$= C$	*ALL*	$= *$	*ALL*	$= *$
ALL	$= *$				

2.3.4. Synthèse

Le cube de données fermé est la représentation la plus concise d'un cube de données. Il exploite les redondances généralement incluses dans un cube de données. En les éliminant, il permet de réduire la taille nécessaire sans perdre de données utiles. Lorsque les données sont très corrélées, la représentation par cube fermé montre tout son intérêt car la réduction apportée est réellement significative. Ces fortes corrélations apparaissent dans les jeux de données réelles et s'accroissent lorsque les données des dimensions sont très éparpillées sur des domaines très vastes. De telles caractéristiques sont typiques des entrepôts de données (ROSS et SRIVASTAVA 1997 ; BEYER et RAMAKRISHNAN 1999).

Il existe d'autres approches comme le cube de données quotient et le cube de données partition.

Le cube de données quotient (LAKSHMANAN, PEI et HAN 2002 ; Q. WANG, YOU, ZOU et al. 2021) est un résumé du cube de données pour certaines fonctions agrégatives comme `COUNT`, `SUM`, `MIN`, `MAX`, `AVG` et `TOP`$-k$. De plus, le cube de données quotient

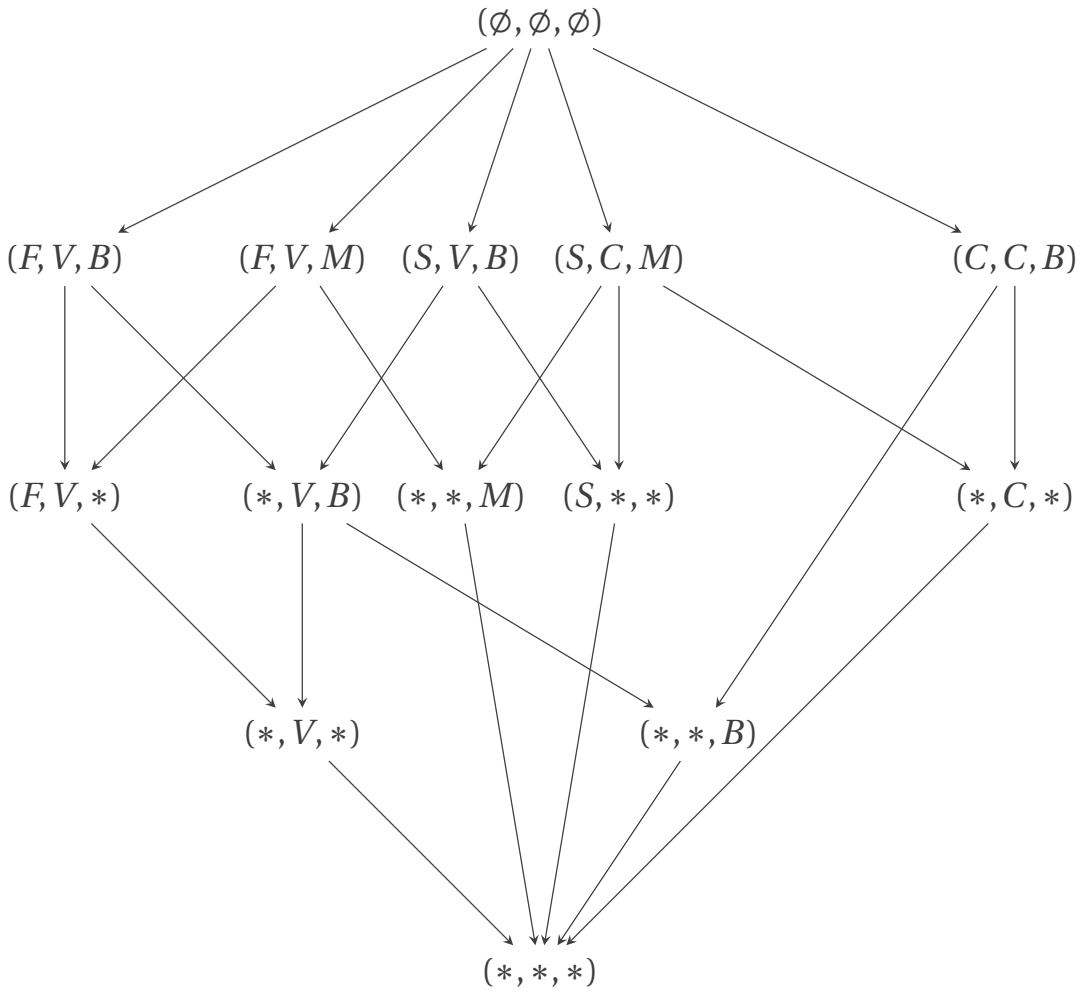


FIGURE 2.12. – Diagramme de Hasse du treillis cube fermé de la relation `Diablo3`

préserve la sémantique des opérateurs `ROLL UP` / `DRILL DOWN` sur le cube de données. Pour définir le cube de données quotient, le concept de classe d'équivalence (*i.e.* ensemble de tuples satisfaisant une relation d'équivalence) est utilisé.

Comme le cube de données fermé et le cube de données quotient, le cube de données partition (LAPORTE, Noel NOVELLI, CICCHETTI et al. 2002 ; CASALI, NEDJAR, CICCHETTI, LAKHAL et Noël NOVELLI 2009) propose une représentation des cubes plus concise que les cubes eux-mêmes. Cependant, elle n'exploite pas les redondances incluses dans les cubes de données comme le fait le cube de données fermé mais relève plutôt d'une optimisation au niveau logique. L'approche s'appuie sur le modèle partitionnel et son opérateur de produit de partitions (SPYRATOS 1987 ; Noel NOVELLI et MAABOUT 2003) pour définir une nouvelle structure réduite pour les cubes de données en revisitant le concept de classe d'équivalence en utilisant celui d'ensemble en accord (MANNILA et TOIVONEN 1996 ; LOPES, PETIT et LAKHAL 2002). C'est une méthode alternative qui permet aussi une réduction de l'espace de stockage, néanmoins moins importante que ne le fait le cube de données. Elle n'enregistre les valeurs des dimensions que dans la relation originelle et pas dans le cube de données, ce qui permet de limiter la taille de ce dernier. Pour cette raison, nous n'avons pas présenté cette approche.

Retenons cependant le concept d'ensemble en accord, car nos contributions s'inscrivent également dans le cadre de ce travail (cf. section 6.3 p. 188).

2.4. Skyline et Skycube

2.4.1. Introduction

Dans un contexte décisionnel, certaines requêtes ne renvoient aucun résultat. Dans ces requêtes, l'utilisateur recherche les tuples pour lesquels les valeurs de certains critères sont optimales. C'est l'usage de plusieurs critères dans ces interrogations qui les rend généralement infructueuses. En effet, tel tuple peut être optimal pour un critère mais pas pour un autre, il est alors éliminé du résultat alors qu'il aurait pu être pertinent pour l'utilisateur.

En considérant la relation exemple `Diablo3` étendue à de nouveaux attributs (cf. exemple 2.22), la recherche du héros « idéal » peut combiner des conditions sur la maximisation à la fois des valeurs de `Parangon`, de `Rareté`, de `Gemmes` et de `Succès`. Évidemment il est vraisemblable que ce héros idéal n'existe pas, d'où l'absence de réponse à ce type de requête. Pourtant certains personnages pourraient s'avérer pertinents pour le décideur parce que, ayant des valeurs de `Gemmes` et de `Succès` globalement élevées, mais sans qu'elles soient maximales, ils réunissent les critères de plus élevé niveaux de `Parangon` et de `Rareté`.

Afin d'apporter une réponse adéquate au type de requêtes décrites, l'opérateur Skyline (BÖRZSÖNYI, KOSSMANN et STOCKER 2001), précédemment connu sous le nom d'ensemble de Pareto et de vecteurs maximaux (BENTLEY, KUNG, SCHKOLNICK et al. 1978), a été introduit. Il considère l'ensemble des critères de choix d'une recherche

comme autant de préférences et extrait les tuples globalement optimaux pour cet ensemble de préférences. Ainsi plutôt que de rechercher une hypothétique solution idéale, il extrait les candidats les plus proches possibles des souhaits de l'utilisateur. Son principe général s'appuie sur la notion de dominance. Un objet ou un tuple est dit dominé par un autre si, pour tous les critères intéressant le décideur, il est moins optimal que cet autre. Un tel tuple est éliminé du résultat, non pas parce qu'il est non pertinent pour un des critères mais parce qu'il est non optimal selon la combinaison de tous les critères. En d'autres termes, il existe au moins une meilleure solution pour l'utilisateur qui, elle, sera retenue.

De la même manière que le cube de données permet de comprendre les liens existant entre plusieurs niveaux d'agrégation, une généralisation multidimensionnelle de l'opérateur Skyline a été proposée à travers le Skycube (YUAN, LIN, Q. LIU et al. 2005 ; PEI, JIN, ESTER et al. 2005). Cette structure réunit tous les Skylines possibles suivant les différentes combinaisons de critères. Il est alors possible de rechercher efficacement des entités dominantes selon différentes combinaisons de critères. De plus, grâce à cette structure, il devient possible d'observer le comportement des entités dominantes à travers l'espace multidimensionnel et ainsi d'analyser et de comprendre les différents facteurs de dominance. Ce concept étant inspiré du cube de données, il souffre des mêmes inconvénients de coût de calcul et d'explosion de l'espace de stockage. Ainsi il est naturel, comme pour le cube de données, d'essayer d'en proposer des représentations réduites et les algorithmes associés.

2.4.2. L'opérateur Skyline

2.4.2.1. Problèmes d'optimisation multicritère

Dans la plupart des cas, les problèmes d'optimisation sont décrits à l'aide de plusieurs objectifs, ou critères, souvent contradictoires (DHAENENS 2005) qui doivent pourtant être optimisés simultanément. L'optimisation multicritère consiste à déterminer, parmi les solutions optimales celles de meilleurs compromis possibles entre les différents critères. Toutes les solutions de cet ensemble sont optimales de sorte que, aucune solution ne domine les autres pour tous les critères considérés. Autrement dit, aucune amélioration ne peut être faite sur l'une des composantes du vecteur-solution sans dégradation d'au moins une autre composante de ce vecteur (ULUNGU EL TEGHEM 1994). Les approches peuvent être classées en deux groupes, les approches non-Pareto et les approches Pareto. Il existe de nombreux ouvrages décrivant en détail ces approches (MIETTINEN 1998 ; EHRGOTT et GANDIBLEUX 2000 ; DEB 2001 ; COLLETTE et SIARRY 2002 ; MAMMERI 2013). Nous nous intéresserons aux approches Pareto.

Optimum de Pareto Vilfredo Pareto est un mathématicien Italien ayant formulé le principe suivant (PARETO 1896) : dans un problème multicritère, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres.

Ce compromis, cet équilibre, est appelé *optimum* de Pareto. Un point x est dit

Pareto-optimal (ou solution non dominée) s'il n'est dominé par aucun autre point appartenant à l'espace de recherche $\mathscr{E}$.

L'optimum de Pareto fait partie de la théorie des jeux, présentée plus en détail à la section 4.6.2.

Dominance de Pareto La dominance au sens de Pareto a été proposée par (GOLDBERG 1989), pour répondre aux problématiques énoncées par Schaffer (SCHAFFER 1985), de la manière définie suivante.

Définition 2.21 (Dominance de Pareto) **-** Un point $x \in \mathscr{E}$ domine x' si :

$$\forall i, f_i(x) \leq f_i(x') \text{ tel que } \exists i, f_i(x) < f_i(x')$$

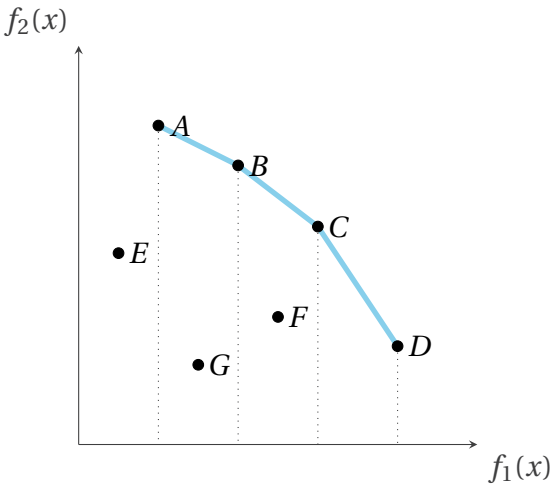


FIGURE 2.13. – Dominance et frontière de Pareto

Exemple 2.21 - Dans la figure 2.13, si l'on considère que les situations préférables sont celles où $f_1(x)$ et $f_2(x)$ sont les plus « fortes », les points A, B, C et D ne sont dominés par aucun autre point. Alors que les points E, F es G ne sont pas sur la frontière de Pareto (en couleur) parce qu'ils sont dominés par les autres points. On dit que A, B, C et D sont efficaces, et qu'ils sont Pareto-optimaux.

Frontière de Pareto La frontière, ou front, (d'efficacité) de Pareto est composée par l'ensemble des points Pareto-optimaux.

2.4.2.2. Cadre d'application

Avant de définir de manière formelle l'opérateur Skyline, il est important de bien situer le contexte dans lequel la problématique se pose. En effet, il ne s'applique pas à n'importe quelle relation d'une base de données. Pour qu'il puisse effectuer les

comparaisons nécessaires, il faut que les différents domaines sur lesquels sont définis les attributs, critères de choix de l'utilisateur, soient totalement ordonnés. Ainsi leurs valeurs pourront toujours être comparées (BÖRZSÖNYI, KOSSMANN et STOCKER 2001 ; PEI, YUAN, LIN et al. 2006). Par simplicité, dans la suite du document, les attributs seront systématiquement codés sous forme de valeurs numériques.

Les tuples de nos relations pouvant être utilisées par l'opérateur Skyline sont de la forme $t = (a_1, a_2, \ldots, a_k, c_1, c_2, \ldots, c_l)$ où les a_i sont les attributs inutiles pour le Skyline alors que les c_i sont les critères sur lesquels l'utilisateur se base pour porter son choix.

TABLEAU 2.8. – La relation `Diablo3`

RowId	Type	Propriété	...	Parangon	Rareté	Gemmes	Succès [a]
1	*Féticheur*	*Vitesse*	...	600	5	30	3
2	*Féticheur*	*Vitesse*	...	900	5	85	7
3	*Sorcière*	*Chance*	...	600	10	50	7
4	*Sorcière*	*Vitesse*	...	100	10	85	5
5	*Croisé*	*Chance*	...	900	10	50	7

a. * Par milliers, arrondi au nombre inférieur. Par exemple 5 équivaux à environ 5000.

Exemple 2.22 - La relation `Diablo3` illustrée par le tableau 2.8 est typique pour l'utilisation de l'opérateur Skyline. Les attributs conservés ici sont `Type`, `Propriété`, ... et les critères de choix pour trouver le « meilleur héros » sont le niveau de `Parangon` (`P`), la `Rareté` (`R`), ou qualité générale, d'un `Équipement` au format numérique (chacune des treize pièces d'inventaire possibles est comptabilisé si elle est un objet légendaire ou d'ensemble ancien ou primordial, et compte double en cas d'arme à deux mains), le niveau minimum de l'ensemble des `Gemmes` (`G`) légendaires et la valeur cumulée des `Succès` (`S`), ou hauts faits, obtenus par le personnage.

Pour des raisons de clarté, le tableau est divisée en trois parties. La première contient le `RowId`. Nous rappelons qu'il s'agit d'un attribut implicite ne faisant pas partie du schéma de la relation : sa valeur sert d'identifiant unique à chaque tuple. La deuxième contient toutes les informations qui ne rentrent pas en compte dans les critères de jugement du décideur. Il n'y a pas de restriction sur le domaine de ces dernières. Enfin, la troisième partie regroupe les attributs préférences qui vont être pris en compte pour le calcul de l'opérateur Skyline. Comme annoncé, leurs valeurs sont toutes numériques.

De plus, il est important de préciser que notre travail de recherche sur le calcul du Skyline reste le plus général possible. Nous ne prenons pas en compte certains cas particuliers qui sont exploitables pour calculer efficacement le Skyline. Il s'agit par exemple de calculs sur des attributs critères dont les domaines ont tous de faibles cardinalités (MORSE, PATEL et JAGADISH 2007) ou bien de Skylines portant sur une ou deux dimensions, cas traités dans BÖRZSÖNYI, KOSSMANN et STOCKER 2001. Nous ne considérerons pas non plus que l'entrée, les tuples de la relation qui nous intéresse,

est assez petite pour entrer en mémoire centrale, comme pour PREPARATA et SHAMOS 1985 et KUNG, LUCCIO et PREPARATA 1975.

2.4.2.3. Opérateur Skyline en SQL

Une syntaxe SQL de l'opérateur Skyline a été proposée pour exprimer des requêtes basées sur des préférences (BÖRZSÖNYI, KOSSMANN et STOCKER 2001).

Dans notre exemple, la recherche des personnages meilleurs candidats, ou l'élite des héros, s'exprime de la façon suivante :

```
01 |  SELECT *
02 |  FROM Diablo3
03 |  SKYLINE OF Parangon MAX, Rareté MAX,
04 |             Gemmes MAX, Succès MAX
```

Dans cette requête, `MAX` signifie que le critère correspondant doit être maximisé, celui de `Parangon`, de `Rareté`, de `Gemmes` et de `Succès`. De manière respective, nous aurions pu utiliser `MIN`, qui signifie que le critère correspondant doit être minimisé. C'est ce que l'on appelle la préférence Skyline.

2.4.2.4. Préférence Skyline

Soit r une relation composée des attributs $A_1, \ldots, A_m$. Une préférence Skyline sur A_j est une expression de l'une des deux formes suivantes : $Pref(A_i) = \texttt{MIN}$, ou $Pref(A_i) = \texttt{MAX}$. Soit t et t' deux tuples de r. Nous considérons que t domine t' (noté $t \prec_d t'$) si et seulement si $t.A_1 \leq t'.A_1, \ldots, t.A_m \leq t'.A_m$ et $\exists j \in [1..m] : t.A_j < t'.A_j$ avec :

$$(\preceq, \prec) = \begin{cases} (\leq, <) \equiv Pref(A_j) = \texttt{MIN} \\ (\geq, >) \equiv Pref(A_j) = \texttt{MAX} \end{cases}$$

2.4.2.5. Définition formelle de Skyline

Maintenant que nous avons décrit dans quelles conditions l'opérateur Skyline est applicable, nous allons le définir de façon formelle.

Définition 2.22 (L'opérateur de dominance $\succ$) - Soit $\mathscr{C} = \{c_1, c_2, \ldots, c_d\}$ l'ensemble des critères sur lesquels va porter l'opérateur Skyline. Sans perte de généralité, on considère qu'ils sont tous de préférence `MIN`. Soit deux tuples t et t' à comparer, on a alors :

$$t \succeq_{\mathscr{C}} t' \Leftrightarrow t[c_1] \leq t'[c_1] \text{ et } t[c_2] \leq t'[c_2] \text{ et } \ldots \text{ et } t[c_d] \leq t'[c_d]$$

On parle aussi de dominance stricte, notée $t \succ_{\mathscr{C}} t'$, lorsqu'on a :

$$t \succ_{\mathscr{C}} t' \text{ et } t \succeq_{\mathscr{C}} t' \text{ et } \exists t.c_i \neq t'.c_i$$

Lorsqu'un tuple t domine un tuple t' (*i.e.* $t \succeq_{\mathscr{C}} t'$), cela signifie que t est équivalent ou « meilleur » que le tuple t' pour tous les critères choisis. Comme nous considérons

que les critères sont tous de type `MIN`, les valeurs de t pour tous les critères sont inférieures ou égales à celles de t'. Ainsi dans le cadre d'une recherche multicritère, au moins un tuples dominés par d'autres ne sont pas pertinents et sont éliminés du résultat par l'opérateur Skyline.

Définition 2.23 (Skyline) - Soit r une relation adéquate, l'opérateur Skyline de r suivant $\mathscr{C}$ renvoie le sous-ensemble S de r tel que tous les éléments restants sont potentiellement intéressants pour l'utilisateur. Il s'agit de l'ensemble des tuples qui ne sont dominés par aucun autre, suivant l'ensemble de critères $\mathscr{C}$. Ce choix est fait à l'aide d'un opérateur de dominance $\succ$:

$$S_{\mathscr{C}}(r) = \{t \in r | \nexists t' \in r \setminus t, t' \succ_{\mathscr{C}} t\}$$

Si les tuples sont considérés comme des points dans un espace d-dimensionnel dont les coordonnées sont les valeurs respectives des attributs critères du Skyline, la dominance est définie sur les coordonnées de ces points, vues sous forme de vecteurs. Le problème se ramène alors à celui du *maximal vector computation* (Kung, Luccio et Preparata 1975 ; Preparata et Shamos 1985 ; Godfrey, Shipley et Gryz 2005 ; Godfrey, Shipley et Gryz 2007).

2.4.2.6. Calcul de Skyline

Il existe une solution naïve pour le calculer du Skyline : il suffit de traduire sa définition formelle en SQL. Nous demandons simplement au SGDB d'afficher les tuples qui ne sont pas dominés par d'autres, selon les critères voulus. Cette requête doit réaliser les opérations suivantes :

— elle doit comparer chaque tupe de r avec tous les autres. Une auto-jointure est donc nécessaire ;
— chaque tuple examiné t doit être écarté du résultat s'il existe un autre tuple t' tel que $t' \succ_{\mathscr{C}} t$;
— cette comparaison doit se faire sur les critères définis par l'utilisateur.

La requête est de la forme suivante. Remarquons que la dernière ligne permet d'obtenir une dominance stricte.

```
01 | SELECT *
02 | FROM R r
03 | WHERE NOT EXISTS (SELECT *
04 |                   FROM R s
05 |                    WHERE s.c1 <= r.c1
06 |                      AND ...
07 |                      AND s.cd <= s.cd
08 |                      AND (s.c1 < r.c1 OR ... OR s.cd < r.cd)
09 | );
```

On peut vérifier aisément que le résultat de la requête est bien celui attendu. Nous disposons donc d'un moyen simple de calculer le Skyline. Il faut maintenant évaluer son efficacité en examinant la complexité de la requête. Chaque tuple de la relation

est comparé avec tous les autres. Les comparaisons s'effectuant en temps constant, la complexité de la requête est en $O(n^2)$ (avec n le nombre de tuples de la relation). Nous allons voir maintenant que le résultat n'est pas satisfaisant dans le contexte des bases de données.

Une complexité en $O(n^2)$ est généralement acceptable lorsqu'on travaille en mémoire centrale. Notre calcul est polynomial en la taille de l'entrée, le nombre de tuples de la relation. Ce nombre est généralement très important et en pratique la relation complète ne tient pas dans la mémoire centrale de la machine. On rencontre fréquemment des bases de données de plusieurs dizaines voire centaines de giga-octets de données, sinon plus. Elles sont donc enregistrées sur un périphérique en mémoire secondaire. Cela a une conséquence immédiate très importante : le temps d'exécution se retrouve considérablement décuplé car les temps d'accès disques sont incomparablement plus longs que les accès mémoire. L'effet principal est que l'algorithme naïf, viable en mémoire centrale, devient tout simplement inexploitable en pratique.

Il existe des algorithmes de calcul de Skyline plus performants. KOSSMANN, RAMSAK et ROST 2002 propose une liste de critères utilisables pour évaluer et comparer ces algorithmes.

Les algorithmes peuvent se classer en deux catégories.

— Les algorithmes sans index :
les algorithmes sans index (Block-Nested-Loop « BNL » proposé par BÖRZSÖNYI, KOSSMANN et STOCKER 2001, extension de l'algorithme Nested-Loops GRUST, KRÖGER, GLUCHE et al. 1997 ; BERCHTOLD, BÖHM, KEIM et al. 1997 ; Divide-and-Conquer « DC » proposé par PREPARATA et SHAMOS 1985 ; KUNG, LUCCIO et PREPARATA 1975 et adapté aux bases de données par BÖRZSÖNYI, KOSSMANN et STOCKER 2001) n'utilisent pas de structure d'indexation particulière et n'ont donc pas besoin d'un pré-traitement pour fonctionner.

— Les algorithmes avec index :
Les algorithmes avec index regroupent tous ceux qui sont basés sur des index. Ils les exploitent pour améliorer l'efficacité du calcul et modifier leur comportement, vis-à-vis par exemple de leur progressivité. La contrepartie est toutefois qu'il faut disposer de l'index nécessaire, ou bien le cas échéant de devoir le calculer, avant de pouvoir d'utiliser ces algorithmes. Plusieurs types d'indexation ont été exploités pour mettre au point des algorithmes de calcul du Skyline (TAN, ENG et OOI 2001) :

  — Les index « classiques » :
  Il s'agit des index couramment utilisés dans les bases de données, tels que les B-Trees ou bien les index Bitmap. Deux algorithmes se basent sur ces index : le premier nommé Index et le second Bitmap, présentés dans PAPADIAS, TAO, G. FU et al. 2003. Ils présentent de nombreux désavantages. Le plus important d'entre eux est que l'index pré-calculé est adapté pour une demande particulière de Skyline.

  — Les index « spatiaux » :
  Appréhender le problème du calcul du Skyline sous son aspect géométrique permet d'exploiter certaines propriétés pour optimiser le temps de réponse

des algorithmes. Développés dans le contexte des bases de données géographiques, les index spatiaux, permettent d'exploiter pleinement certaines de ces propriétés (les R-Trees présentés dans GUTTMAN 1984, les X-Trees de BERCHTOLD, KEIM et KRIEGEL 1996, les SH-Trees de DANG 2006, R+-Trees de SELLIS, ROUSSOPOULOS et FALOUTSOS 1987, R*-Trees de BECKMANN, KRIEGEL, SCHNEIDER et al. 1990, Nearest Neighbor « NN » de ROUSSOPOULOS, KELLEY et VINCENT 1995 et le très efficace Branch-and-Bound Skyline « BBS » (PAPADIAS, TAO, G. FU et al. 2003 ; PAPADIAS, TAO, G. FU et al. 2005) et ses améliorations et optimisations HOSE et VLACHOU 2012 ; Z. LIU, L. LI, M. ZHANG et al. 2023).

2.4.3. Skycube

L'opérateur Skyline est un outil fondamental pour l'analyse utilisant plusieurs critères. Nous pouvons calculer un Skyline suivant un ensemble de critères définis par l'utilisateur, mais lorsqu'il s'agit d'en calculer plusieurs sur les mêmes données, aucun d'eux ne sait exploiter à son avantage les liens qui peuvent exister entre les différents Skylines. C'est pour cela qu'une structure nommée Skycube a été introduite (YUAN, LIN, Q. LIU et al. 2005 ; PEI, JIN, ESTER et al. 2005). On peut dire que le Skycube est au Skyline ce que le cube de données est au partitionnement : une généralisation multidimensionnelle. Ainsi, dès lors que l'on souhaite répondre rapidement à toutes les requêtes posées sur un Skycube, il vaut mieux opter pour une pré-matérialisation de ce cube de données.

2.4.3.1. Cadre formel du Skycube

Dans cette section, le concept de Skyline est étendu à l'espace multidimensionnel et nous présentons le concept de Skycube.

Définition 2.24 (Sous-espace Skyline) - Un sous-ensemble de dimensions $\mathscr{C} \subseteq \mathscr{D}$ ($B \neq \emptyset$) forme un sous-espace $|\mathscr{C}|$-dimensionnel de $\mathscr{D}$. Pour un tuple t dans l'espace D, la projection de t dans le sous-espace $\mathscr{C}$, dénoté par $t[\mathscr{C}]$, est un $|\mathscr{C}|$-tuple $(t.Di_1, \ldots, t.Di_{|\mathscr{C}|})$, avec $t.Di_1, \ldots, t.Di_{|\mathscr{C}|} \in \mathscr{C}$ et $i_1 < \ldots < i_{|\mathscr{C}|}$. La projection d'un tuple t ($t \in r$) dans un sous-espace $\mathscr{C} \subseteq \mathscr{D}$ appartient au sous-espace Skyline selon $\mathscr{C}$, si aucun tuple $t'[\mathscr{C}]$ (avec $t' \in r$) ne domine $t[\mathscr{C}]$ dans $\mathscr{C}$. t est appelé un objet d'un sous-espace Skyline selon $\mathscr{C}$. Nous notons $SKY_{\mathscr{C}}(r)$ le sous-espace Skyline selon $\mathscr{C}$ pour la relation r.

Définition 2.25 (Skycube) - Un Skycube est l'ensemble de tous les Skylines dans tous les sous-espaces non vides possibles de $\mathscr{C}$:

$$SKYCUBE(r, \mathscr{C}) = \{(C, SKY_C(r)) \mid C \subseteq \mathscr{C}\}$$

$SKY_C(r)$ est appelé le cuboïde Skyline (ou Skycuboïde) du sous-espace C. Par convention le cuboïde selon l'ensemble de critère vide est vide (*i.e.* $SKY_{\emptyset}(r) = \emptyset$).

La structure du Skycube peut être représentée par un treillis semblable à celui utilisé pour le cube de données (cf. figure 2.14). Les cuboïdes du Skycube sont regroupés par niveau en fonction de leur nombre de critères. Ces niveaux sont numérotés en partant du bas du treillis (cuboïdes portant sur un seul critère) et en remontant vers le sommet (cuboïde suivant tous les critères possibles).

Exemple 2.23 (Skycube) - La figure 2.14 donne le Skycube associé à la relation `Diablo3` (cf. tableau 2.8, `RowId` est symbolisé par `Id`, et les critères par leur initiale).

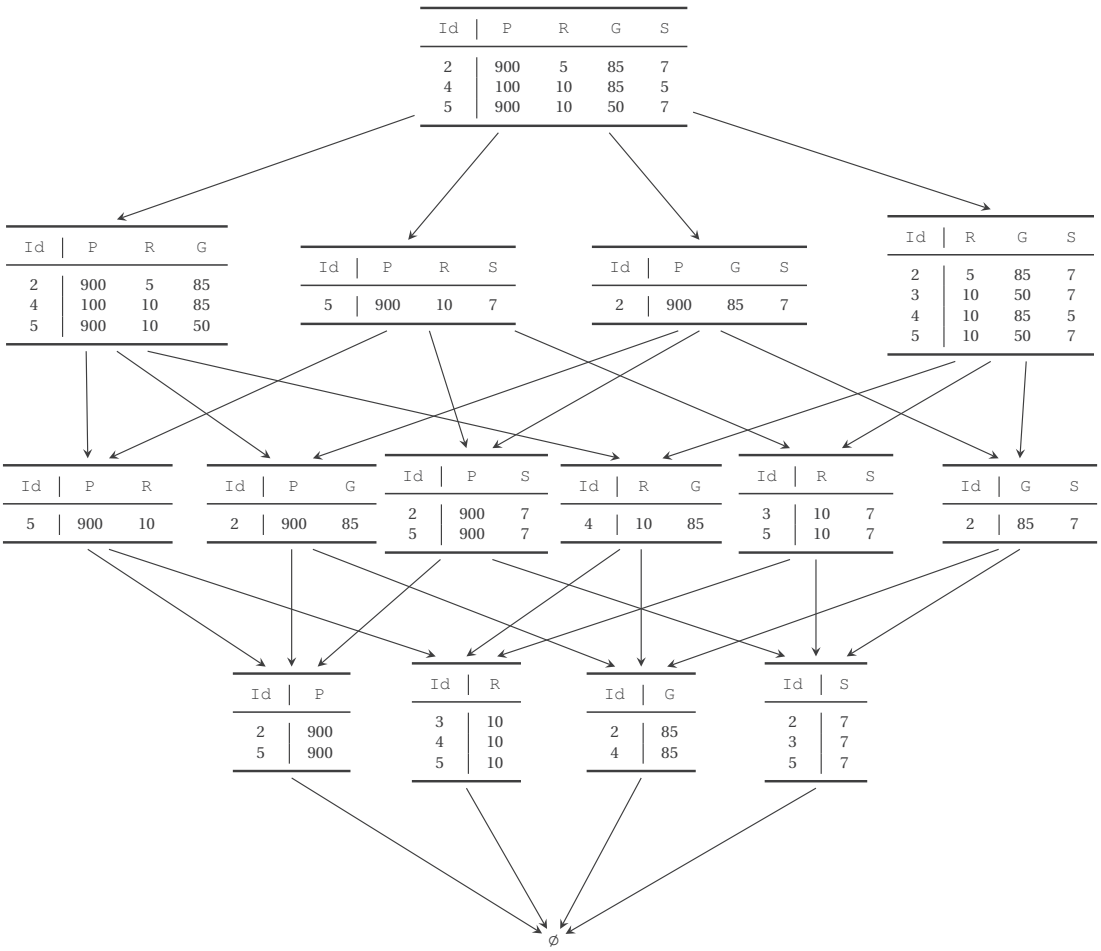


FIGURE 2.14. – Représentation du treillis du Skycube de la relation `Diablo3`

Une requête Skyline multidimensionnelle retourne le sous-ensemble de tuples de la relation originelle formant le Skyline dans un sous-espace donné. Une fois le Skyline calculé, il est possible de répondre à toute requête efficacement.

2.4.3.2. Problèmes associés aux Skylines

En général, si un tuple t est dans les Skylines des sous-espaces C_1 et C_2 tels que $C_1 \subset C_2$, pouvons-nous affirmer que t appartiendra aussi au Skyline de n'importe

quel sous-espace C situé entre C_1 et C_2 ($C_1 \subset C \subset C_2$)? Une telle propriété serait fort attrayante puisqu'elle pourrait considérablement simplifier la détermination des Skylines multidimensionnels. Malheureusement elle n'est pas vérifiée dans le cas général.

Exemple 2.24 - Avec la relation `Diablo3`, comme le montre la figure 2.14, en considérant les Skylines selon (`Parangon`, `Rareté`, `Succès`) et (`Rareté`, `Succès`) on a : (`Parangon`, `Rareté`, `Succès`) $\supseteq$ (`Rareté`, `Succès`) et SKY_{PRS}(`Diablo3`) $\subseteq$ SKY_{RS}(`Diablo3`).

En revanche, si l'on considère les Skylines selon (`Parangon`, `Rareté`, `Gemmes`, `Succès`) et (`Parangon`, `Rareté`, `Gemmes`), nous avons alors (`Parangon`, `Rareté`, `Gemmes`, `Succès`) $\supseteq$ (`Parangon`, `Rareté`, `Gemmes`) et, de même, nous avons alors SKY_{PRGS}(`Diablo3`) $\supseteq$ SKY_{PRG}(`Diablo3`).

L'observation mise en évidence par cet exemple est la suivante : l'appartenance à un Skyline n'est pas monotone, c'est-à-dire qu'un tuple t appartenant à un cuboïde $SKY_{\mathscr{U}}(r)$ n'est pas automatiquement contenu dans les ancêtres de ce cuboïde.

2.4.3.3. Calcul de Skycube

L'analyse des données multidimensionnelles est à l'origine de nombreux efforts de recherche au cours des dernières années. L'un des aspects qui a été abordé jusqu'à présent est celui de permettre aux utilisateurs d'analyser leurs données à partir de différentes perspectives, chacune correspondant à un sous-ensemble sélectionné de dimensions. Pour optimiser ces analyses, les requêtes, le pré-calcul et la matérialisation sont parmi les solutions les plus étudiées. Pour rendre le Skycube utile en pratique, deux principaux axes de recherche ont été poursuivis jusqu'à présent.

- Le premier axe vise à proposer des algorithmes efficaces pour le calculer. Comme dans le cas du cube de données, le Skycube peut contenir des informations superflues. Le nombre de ces skylines est exponentiel par rapport à $\mathscr{D}$. Le temps d'exécution et l'espace de stockage rendent ces solutions difficiles à mettre en œuvre même avec des ensembles de données de taille modérée.
- Le second axe de recherche, motivé par la problématique de l'axe précédent, a motivé la proposition de représentations réduites du Skycube faite par Pei, Yuan, Lin et al. 2006 afin de réduire la consommation de temps et d'espace. Parmi les algorithmes de calcul performants de Skycube réduits, nous pouvons citer :
  - la famille des algorithmes basée sur les sous-espace décisifs à stratégie de mutualisation des résultats et des tris comme le Sort First SkyLine de Chomicki, Godfrey, Gryz et al. 2005 ainsi que le Bottom-Up Skycube (BUS) et le Top-Down Skycube de Yuan, Lin, Q. Liu et al. 2005, ou plus efficace, en utilisant un Skytree, représentant un partitionnement récursif de l'espace basé sur des points, comme le QSkycube (Lee et S. Hwang 2010) ou encore en utilisant les dimensions hiérarchiques comme le HSky (Hierarchical Skyline Queries) de Bouadi, Cordier et Quiniou 2014, ou le très efficace algorithme Orion

et ses variantes (RAÏSSI, PEI et KISTER 2010 ; RAÏSSI 2012) ;
— la famille des algorithmes de RkSB (pour reverse k-Skyband, ou k-Skyband inversé) comme le Branch-Bound-Based Reverse k-Skyband (BRkSB), le Pre-computation-Based Reverse k-Skyband (PRkSB) ou le Optimized PRkSB Algorithm (OPRkSB) de Q. LIU, GAO, G. CHEN et al. 2012 ;
— le complémentaire du Skycube, appelé Skycube negative (HANUSSE, WANKO et MAABOUT 2016 ; ALAMI, HANUSSE, WANKO et al. 2020).

2.4.4. Synthèse

Dans cette section, nous nous sommes penchés sur l'analyse utilisant plusieurs critères à travers l'opérateur Skyline. Il permet à l'utilisateur d'extraire les solutions satisfaisant au mieux un ensemble de critères spécifiés. Après une description du contexte d'utilisation de cet opérateur, nous avons étudié sa définition, ainsi qu'une solution sous forme d'une requête SQL bien que celle-ci est trop peu performante pour être utilisée efficacement.

Le concept de Skycube partage avec le cube de données les mêmes avantages mais aussi les mêmes défauts dont celui d'un énorme volume. Pour remédier à cet inconvénient, il nous faudra étudier une représentation visant à réduire les redondances.

2.5. Conclusion

Dans ce chapitre, nous avons parcouru l'analyse des base de données traditionnelles selon trois prismes : le cube de données et ses méthodes de calcul, le cube de données fermé et ses treillis, et l'opérateur Skyline et sa généralisation multidimensionnelle, le Skycube.

La leçon à retenir de l'état de l'art du cube de données est que le treillis des cuboïdes ne peut pas constituer un espace de recherche satisfaisant pour calculer efficacement les cubes de données. En effet, en examinant l'algorithme BUC et plus précisément son parcours, pourtant présenté comme celui du treillis des cuboïdes, il apparaît que le niveau de granularité requis est celui des tuples et non des cuboïdes. C'est de cette particularité que BUC tire sa très grande efficacité. Il est assez étonnant de constater qu'à ce stade, le cube de données n'est vu que comme le résultat d'un calcul (soit celui d'une requête soit celui plus performant d'un algorithme) et n'est pas précisément défini.

Nous avons présenté une approche de représentation des cubes de données qui réduit le volume des données sans aucune perte d'information : le cube de données fermé. Le cube de données émergent, dont nous nous servirons largement, décrit au chapitre suivant (cf. chapitre 3 p. 86) réutilise largement cette structure et ses propriétés.

Outre la réponse à des requêtes bien particulières, la problématique de l'opérateur Skyline nous a intéressé car ce concept a été couplé à celui de cube de données. Ainsi, une nouvelle problématique OLAP a t-elle fait son apparition. Comme pour le cube

de données, il s'agit de combiner, au sein d'une même structure, tous les Skylines possibles selon un ensemble de critères.

3. Cube de données émergent

Sommaire

3.1. Introduction 86
3.2. Concepts 87
3.3. Bordures pour le cube de données émergent 91
3.3.1. Bordures [L; U] 93
3.4. Calcul du cube émergent et de ses bordures 95
3.4.1. Cube de données émergent en SQL 95
3.4.2. L'algorithme du cube de données émergent : E-IDEA 98
3.4.2.1. Description de l'algorithme E-IDEA 103
3.4.3. F-IDEA 104
3.5. Cube de données fermé émergent 108
3.5.1. L-cube de données fermé émergent 110
3.5.2. L'algorithme de cube de données fermé émergent : ℂ-IDEA . . . 116
3.6. Conclusion 117

3.1. Introduction

Le cube de données émergent (NEDJAR, CASALI, CICCHETTI et al. 2007 ; NEDJAR, CASALI, CICCHETTI et al. 2010b ; NEDJAR, CASALI, CICCHETTI et al. 2009 ; NEDJAR, CICCHETTI et LAKHAL 2011 ; NEDJAR, LAKHAL et CICCHETTI 2013) permet de capturer les renversements de tendances dans les base de données OLAP. Ce concept résulte du couplage de deux structures intéressantes, le cube de données (GRAY, CHAUDHURI, BOSWORTH et al. 1997) et les motifs émergents (DONG et J. LI 1999), dans le même esprit que le Skycube (PEI, YUAN, LIN et al. 2006) qui combine le concept de cube de données et la dominance de Pareto (BÖRZSÖNYI, KOSSMANN et STOCKER 2001). À partir des cubes de données de deux relations d'une base de données pourvues de dimensions (ou d'attributs catégories), le cube de données émergent rassemble tous les tuples satisfaisant une double contrainte d'émergence : la valeur de leur mesure est faible dans une relation (contrainte C_1) et significative dans l'autre (contrainte C_2). Nous rappelons le concept du cube de données émergent et proposons une représentation condensée à travers les bordures $[L;U]$ (pour respectivement les bordures *lower* et *upper*) (DONG et J. LI 1999 ; RAEDT et KRAMER 2001 ; BONCHI et LUCCHESE 2004). Après la caractérisation du cube de données émergent et de ses représentations, nous nous intéressons aux différentes approches permettant de calculer ces dernières. Dans un premier temps, nous étudions les requêtes SQL qui renvoient le cube de données

émergent partiellement ou complètement. Elles apportent une solution au problème sans investissement de développement et peuvent être mises en œuvre facilement à condition toutefois que la taille des données initiales soit raisonnable et que l'on ne souhaite pas calculer les bordures. Lorsque cette double condition n'est pas remplie, les approches algorithmiques s'imposent. Pour éviter de développer un nouvel algorithme, le choix s'est porté sur la réutilisation de BUC (BEYER et RAMAKRISHNAN 1999) pour son extrême efficacité et sa capacité rare d'intégration dans un système relationnel. Dans un premier temps, nous présentons l'adaptation du schéma algorithmique de BUC au calcul des cubes émergents avec l'algorithme appelé E-IDEA. Ensuite nous proposons un autre algorithme, F-IDEA, permettant d'obtenir les représentations par bordures dans un contexte multidimensionnel. Cet algorithme représente la première alternative de calcul des bordures dans un contexte multidimensionnel sans utiliser les approches binaires telles que Max-Miner (JR. 1998), Mafia (BURDICK, CALIMLIM, FLANNICK et al. 2005) et GenMax (GOUDA et ZAKI 2005).

Dans ce chapitre, nous présentons le cube de données émergent après un bref rappel du contexte dans lequel il s'inscrit. Nous étudions ensuite les bornes $[L;U]$. Nous trouverons une présentation complète du cube de données émergent dans NEDJAR 2009.

3.2. Concepts

Dans cette section, nous présentons la problématique étudiée en proposant le concept de cube de données émergent. Ces cubes capturent les tendances non significatives pour l'utilisateur (parce que leur mesure est en dessous d'un certain seuil) mais qui croissent suffisamment pour le devenir ou, respectivement, des tendances avérées qui s'essoufflent sans nécessairement disparaître.

Par exemple, elles sont d'un intérêt certain dans l'analyse de flots de données, car elles révèlent des renversements de tendances. Dans une application Web où les flots continus de données reçues décrivent de manière détaillée la navigation des utilisateurs (HAN, Y. CHEN, DONG et al. 2005), connaître l'engouement (ou au contraire le désintérêt) pour telle ou telle URL est particulièrement important pour l'administrateur afin d'allouer au mieux les ressources disponibles selon des besoins réels et fluctuants.

La caractérisation du cube de données émergent s'inscrit dans le contexte plus général du treillis cube de la relation r : $CL(r)$ (CASALI, CICCHETTI et LAKHAL 2003a). Ce dernier, décrit dans la section 2.3, est un espace de recherche bien adapté au calcul du data cube de r. Il organise les tuples, solutions possibles du problème, selon un ordre de généralisation/spécialisation, noté $\preceq_g$ (LAKSHMANAN, PEI et HAN 2002). Ces tuples sont structurés selon les attributs dimensions de r et ces derniers peuvent prendre la valeur *ALL* (GRAY, CHAUDHURI, BOSWORTH et al. 1997). De plus, est ajouté à ces tuples un tuple virtuel ne contenant que des valeurs vides afin de fermer la structure. Tout tuple du treillis cube généralise le tuple de valeurs vides. Pour manipuler les tuples de $CL(r)$, l'opérateur + est défini. À partir d'un couple de tuples,

il retourne le tuple le plus spécifique de $CL(r)$ qui généralise les deux opérandes.

TABLEAU 3.1. – Relation exemple Diablo3

Type	Propriété	Équipement	Compagnon	Saison	Quantité
Féticheur	*Vitesse*	*Bastion*	*Enchanteresse*	19	100
Féticheur	*Vitesse*	*Bastion*	*Enchanteresse*	20	300
Féticheur	*Vitesse*	*Marche*	*Templier*	19	100
Sorcière	*Vitesse*	*Bastion*	*Templier*	20	300
Sorcière	*Vitesse*	*Marche*	*Templier*	19	100
Sorcière	*Vitesse*	*Marche*	*Enchanteresse*	20	300
Sorcière	*Chance*	*Marche*	*Templier*	20	300
Sorcière	*Chance*	*Marche*	*Enchanteresse*	19	100
Croisé	*Vitesse*	*Marche*	*Enchanteresse*	20	100
Croisé	*Chance*	*Bastion*	*Enchanteresse*	20	200
Croisé	*Chance*	*Marche*	*Enchanteresse*	19	600
Croisé	*Chance*	*Marche*	*Enchanteresse*	20	200

Exemple 3.1 - Considérons, à nouveau, la relation Diablo3 (cf. tableau 3.1), étendue et mise à jour, donnant la Quantité de personnages par Type, Propriété, Équipement, Compagnon et, auxquels nous ajoutons également, par Saison. Dans CL(Diablo3), prenons les personnages de Type *Féticheur* avec *Vitesse* quels que soient Équipement, Compagnon et Saison, autrement dit le tuple (*Féticheur*, *Vitesse*, *ALL*, *ALL*, *ALL*). Nous remarquons que ce tuple est spécialisé par le tuple (*Féticheur*, *Vitesse*, *Bastion*, *Enchanteresse*, 20) mais aussi par le tuple (*Féticheur*, *Vitesse*, *Marche*, *Templier*, 19) et (*Féticheur*, *Vitesse*, *ALL*, *ALL*, *ALL*) $\preceq_g$ (*Féticheur*, *Vitesse*, *Bastion*, *Enchanteresse*, 20) illustre l'ordre de généralisation entre tuples. Pour finir nous avons (*Féticheur*, *Vitesse*, *Marche*, *Templier*, 20)+ (*Féticheur*, *Vitesse*, *Bastion*, *Enchanteresse*, 20) = (*Féticheur*, *Vitesse*, *ALL*, *ALL*, *ALL*).

Dans la suite de ce chapitre, nous considérons seulement les fonctions agrégatives étudiées par PEI, HAN et LAKSHMANAN 2004, comme par exemple COUNT et SUM.

Définition 3.1 (Fonction de mesure) - Soit f une fonction agrégative, r une relation et t un tuple (ou cellule) de $CL(r)$. Nous notons $f_{val}(t,r)$ la valeur de la fonction agrégative f associée au tuple t dans $CL(r)$.

Exemple 3.2 - Si nous considérons les personnages de Type *Féticheur* avec *Vitesse*, pour n'importe quels Équipement, Compagnon et Saison, autrement dit le tuple (*Féticheur*, *Vitesse*, *ALL*, *ALL*, *ALL*) de CL(Diablo3), nous avons :

$$SUM_{val}((Féticheur, Vitesse, ALL, ALL, ALL), \texttt{Diablo3}) = 500$$

Un tuple émergent de r_1 vers r_2 a une valeur de mesure faible dans r_1 alors qu'elle est significative dans r_2.

Définition 3.2 (Tuple émergent) - Un tuple $t \in CL(r_1 \cup r_2)$ est dit émergent de r_1 vers r_2 si et seulement s'il satisfait les deux contraintes C_1 et C_2 :

$$\begin{cases} f_{val}(t, r_1) < SeuilMin_1(C_1) \\ f_{val}(t, r_2) \geq SeuilMin_2(C_2) \end{cases}$$

TABLEAU 3.2. – Relation $\texttt{Diablo3}_1$, personnages de la `Saison` 19

Type	Propriété	Équipement	Compagnon	Quantité
Féticheur	*Vitesse*	*Bastion*	*Enchanteresse*	100
Féticheur	*Vitesse*	*Marche*	*Templier*	100
Sorcière	*Chance*	*Marche*	*Enchanteresse*	100
Croisé	*Chance*	*Marche*	*Enchanteresse*	600
Sorcière	*Vitesse*	*Marche*	*Templier*	100

TABLEAU 3.3. – Relation exemple $\texttt{Diablo3}_2$, personnages de la `Saison` 20

Type	Propriété	Équipement	Compagnon	Quantité
Féticheur	*Vitesse*	*Bastion*	*Enchanteresse*	300
Sorcière	*Vitesse*	*Bastion*	*Templier*	300
Sorcière	*Vitesse*	*Marche*	*Enchanteresse*	300
Sorcière	*Chance*	*Marche*	*Templier*	300
Croisé	*Vitesse*	*Marche*	*Enchanteresse*	100
Croisé	*Chance*	*Marche*	*Enchanteresse*	200
Croisé	*Chance*	*Bastion*	*Enchanteresse*	200

Exemple 3.3 - Imaginons que nous cherchions les renversements de tendances s'étant produits entre la `Saison` 19 et la `Saison` 20 pour les personnages répertoriés dans la relation `Diablo3` (cf. tableau 3.1). La relation $\texttt{Diablo3}_1$ (cf. tableau 3.2) correspond aux personnages de la `Saison` 19 et $\texttt{Diablo3}_2$ (cf. tableau 3.3) sélectionne les tuples de `Diablo3` pour lesquels `Saison` = 20. Soit $SeuilMin_1 = 200$ le seuil pour la relation $\texttt{Diablo3}_1$ et $SeuilMin_2 = 200$ celui pour la relation $\texttt{Diablo3}_2$, le tuple $t_1 = (Sorcière, Vitesse, ALL, ALL)$ est émergent de $\texttt{Diablo3}_1$ vers $\texttt{Diablo3}_2$ car $SUM_{val}(t_1, \texttt{Diablo3}_1) = 100$ ($< SeuilMin_1$) et $SUM_{val}(t_1, \texttt{Diablo3}_2) = 600$ ($\geq SeuilMin_2$). En revanche, le tuple $t_2 = (Croisé, Vitesse, ALL, ALL)$ n'est pas émergent car $SUM_{val}(t_2, \texttt{Diablo3}_2) = 100$.

Dans l'objectif de pouvoir quantifier clairement un renversement de tendances, nous introduisons la mesure du taux d'emergence.

Définition 3.3 (Taux d'emergence) - Soit r_1 et r_2 deux relations unicompatibles, $t \in CL(r_1 \cup r_2)$ un tuple et f une des fonctions agrégatives citées. Le taux d'émergence de

t de r_1 vers r_2, noté $ER(t)$, est défini par :

$$ER(t) = \begin{cases} 0 \text{ si } f_{val}(t, r_1) = 0 \text{ et } f_{val}(t, r_2) = 0 \\ \infty \text{ si } f_{val}(t, r_1) = 0 \text{ et } f_{val}(t, r_2) \neq 0 \\ \dfrac{f_{val}(t, r_2)}{f_{val}(t, r_1)} \text{ sinon.} \end{cases}$$

Le taux d'émergence varie en fonction des seuils $SeuilMin_1$ et $SeuilMin_2$ fixés par l'utilisateur. Nous distinguons trois cas possibles.

1. Si $SeuilMin_1 = 0$ et $SeuilMin_2 > 0$, alors $ER(t) = \infty$. On cherche donc des tuples ne généralisant aucun tuple de r_1. Ces tuples émergents correspondent aux nouvelles tendances de r_1 vers r_2.
2. Si $SeuilMin_1 \leq SeuilMin_2$, alors $ER(t) \geq 1$. Les tuples résultats sont alors positivement émergents. Ils sont caractérisés par une nette augmentation de leur mesure entre r_1 et r_2.
3. Si $SeuilMin_1 > SeuilMin_2$, alors $ER(t) \in [0, 1[$. Les tuples solutions sont négativement émergents (ou immergents). Ces tuples ont pour caractéristique une diminution de leur mesure de r_1 vers r_2.

Observons que lorsque $ER(t) \geq 1$, il caractérise des tendances significatives dans r_2 mais insuffisamment marquée dans r_1. Au contraire, quand $ER(t) < 1$, il souligne les tendances immergentes, pertinentes dans r_1 mais pas dans r_2.

Exemple 3.4 - À partir des relations $\texttt{Diablo3}_1$ et $\texttt{Diablo3}_2$, nous pouvons calculer $ER((Sorcière, Vitesse, ALL, ALL)) = 600/100$. Logiquement, plus le taux d'émergence est élevé, plus le renversement de tendance est marqué. Ainsi le tuple cité signifie un bond pour les personnages de `Type` $Sorcière$ avec $Vitesse$ entre $\texttt{Diablo3}_1$ et $\texttt{Diablo3}_2$.

Propriété 3.1 (MinRatio) - Soit $MinRatio = \frac{SeuilMin_2}{SeuilMin_1}$,

$$\forall t \in EC(r_1, r_2), \text{ nous avons } ER(t) > MinRatio$$

La conséquence de cette propriété est que les tuples émergents ont un taux d'émergence « avantageux ». Cela constitue une première aide pour l'utilisateur afin de bien spécifier ses contraintes d'émergence. Par exemple considérons $SeuilMin_1 = 300$ et $SeuilMin_2 = 500$, alors tous les tuples émergents ont un taux d'émergence supérieur à 5/3.

Définition 3.4 (Cube de données émergent) - Nous appelons cube de données émergent l'ensemble de tous les tuples de $CL(r_1 \cup r_2)$ émergents de r_1 vers r_2. Le cube de données émergent, noté $EC(r_1, r_2)$, est défini par :

$$EC(r_1, r_2) = \{t \in CL(r_1 \cup r_2) \mid C_1(t) \wedge C_2(t)\}$$

Avec $C_1(t) = f_{val}(t, r_1) < SeuilMin_1$ et $C_2(t) = f_{val}(t, r_2) \geq SeuilMin_2$.

Exemple 3.5 - Le tableau 3.4 donne le cube de données émergent de la relation $\texttt{Diablo3}_1$ vers la relation $\texttt{Diablo3}_2$ avec les seuils $SeuilMin_1 = 200$ et $SeuilMin_2 = 200$.

Les coûts de calcul et d'enregistrement du cube de données émergent dépendent du nombre de tuples émergents. Ce nombre est borné par la taille de l'espace de recherche (l'ensemble des tuples possibles). L'espace de recherche dans lequel s'inscrit le cube de données émergent est lié aux relations en entrée ($CL(r_1 \cup r_2)$). Le second paramètre qui va influer sur le nombre de solutions est la double contrainte d'émergence. C'est le décideur qui fixe cette contrainte en donnant les seuils ($SeuilMin_1$ et $SeuilMin_2$). Plus la contrainte est forte plus le cube de données émergent associé est réduit. Le cas extrême étant atteint quand $SeuilMin_1 = 0^+$ et $SeuilMin_2 = +\infty$. Avec de tels seuils, aucun tuple n'est émergent et le résultat est toujours l'ensemble vide. Inversement, lorsque la contrainte est plus lâche moins de tuples sont filtrés. Le cube de données émergent est donc plus volumineux. La contrainte d'émergence la plus faible possible est obtenue pour $SeuilMin_1 = +\infty$ et $SeuilMin_2 = 0^+$. Dans ce cas, pour qu'un tuple soit émergent il faut que la valeur de sa mesure dans r_2 soit différente de 0. Le tuple doit donc appartenir au cube de données de r_2. Ainsi, dans le pire des cas (du point de vue de sa taille), le calcul du cube de données émergent se ramène au calcul du cube de données de r_2.

Le calibrage des contraintes pour obtenir un cube de données émergent d'un volume réellement exploitable est important. En ayant identifié les cas extrêmes, nous avons vu que le problème de calcul d'un cube de données émergent est au moins aussi difficile (du point de vue de la complexité) que le problème de calcul d'un cube de données (*c.f* paragraphe 2.2.5 p. 44). La spécification des contraintes est donc un élément clef pour que l'utilisateur puisse appréhender puis exploiter les résultats.

3.3. Bordures pour le cube de données émergent

Dans ce paragraphe, nous présentons les bordures $[L;U]$. Les bordures offrent une représentation des cubes émergents et ont des avantages supplémentaires. Tout d'abord, les renversements de tendances peuvent être isolés très efficacement et à un moindre coût car il n'est pas nécessaire de calculer et d'enregistrer les deux cubes sous-jacents. Ainsi, nous économisons temps d'exécution et espace de stockage. De plus, nous pouvons de manière immédiate savoir, par une requête, si une tendance donnée est émergente. Afin de gérer systématiquement ce type de requête, nous pouvons concevoir des classifieurs efficaces pour décider si une tendance est émergente ou pas. De plus, en se focalisant sur des tuples particuliers, les bordures des cubes émergents offrent un point de départ intéressant pour la navigation à travers un cube de données. Enfin, nous tenons compte des bordures décrites afin de caractériser la taille exacte des cubes émergents.

TABLEAU 3.4. – Cube de données émergent de $\texttt{Diablo3}_1$ vers $\texttt{Diablo3}_2$

Tuple émergent	*ER*
(*ALL*, *ALL*, *ALL*, *Templier*)	3
(*ALL*, *ALL*, *Bastion*, *ALL*)	8
(*ALL*, *ALL*, *Bastion*, *Templier*)	∞
(*ALL*, *ALL*, *Bastion*, *Enchanteresse*)	5
(*ALL*, *ALL*, *Marche*, *Templier*)	1.5
(*ALL*, *Chance*, *ALL*, *Templier*)	∞
(*ALL*, *Chance*, *Marche*, *Templier*)	∞
(*ALL*, *Vitesse*, *ALL*, *Templier*)	1.5
(*ALL*, *Vitesse*, *ALL*, *Enchanteresse*)	7
(*ALL*, *Vitesse*, *Bastion*, *ALL*)	6
(*ALL*, *Vitesse*, *Bastion*, *Templier*)	∞
(*ALL*, *Vitesse*, *Bastion*, *Enchanteresse*)	3
(*ALL*, *Vitesse*, *Marche*, *ALL*)	2
(*ALL*, *Vitesse*, *Marche*, *Enchanteresse*)	∞
(*Féticheur*, *ALL*, *ALL*, *ALL*)	1.5
(*Féticheur*, *ALL*, *ALL*, *Enchanteresse*)	3
(*Féticheur*, *ALL*, *Bastion*, *ALL*)	3
(*Féticheur*, *ALL*, *Bastion*, *Enchanteresse*)	3
(*Féticheur*, *Vitesse*, *ALL*, *ALL*)	1.5
(*Féticheur*, *Vitesse*, *ALL*, *Enchanteresse*)	3
(*Féticheur*, *Vitesse*, *Bastion*, *ALL*)	3
(*Féticheur*, *Vitesse*, *Bastion*, *Enchanteresse*)	3
(*Sorcière*, *ALL*, *ALL*, *ALL*)	4.5
(*Sorcière*, *ALL*, *ALL*, *Templier*)	6
(*Sorcière*, *ALL*, *ALL*, *Enchanteresse*)	3
(*Sorcière*, *ALL*, *Bastion*, *ALL*)	∞
(*Sorcière*, *ALL*, *Bastion*, *Templier*)	∞
(*Sorcière*, *ALL*, *Marche*, *ALL*)	3
(*Sorcière*, *ALL*, *Marche*, *Templier*)	3
(*Sorcière*, *ALL*, *Marche*, *Enchanteresse*)	3
(*Sorcière*, *Chance*, *ALL*, *ALL*)	3
(*Sorcière*, *Chance*, *ALL*, *Templier*)	∞
(*Sorcière*, *Chance*, *Marche*, *ALL*)	3
(*Sorcière*, *Chance*, *Marche*, *Templier*)	∞
(*Sorcière*, *Vitesse*, *ALL*, *ALL*)	6
(*Sorcière*, *Vitesse*, *ALL*, *Templier*)	3
(*Sorcière*, *Vitesse*, *ALL*, *Enchanteresse*)	∞
(*Sorcière*, *Vitesse*, *Bastion*, *ALL*)	∞
(*Sorcière*, *Vitesse*, *Bastion*, *Templier*)	∞
(*Sorcière*, *Vitesse*, *Marche*, *ALL*)	3
(*Sorcière*, *Vitesse*, *Marche*, *Enchanteresse*)	∞

3.3.1. Bordures [L; U]

Étant donnée la définition des contraintes d'émergence C_1 et C_2, qui sont respectivement monotone et antimonotone, pour chaque tuple t satisfaisant la contrainte C_1, nous savons que tous les tuples plus spécifiques que t satisfont aussi la contrainte C_1. Cela implique que les tuples les moins spécifiques (les minimaux selon $\preceq_g$) satisfaisant la contrainte C_1 sont les plus importants car ils délimitent l'espace des solutions. Inversement, dans le cas de la contrainte C_2, ce sont les tuples les plus spécifiques (les maximaux selon $\preceq_g$) qui sont les plus importants. Ces deux ensembles de tuples forment les frontières du cube de données émergent.

Définition 3.5 (Bordures [L; U]) **-** Le cube de données émergent peut être représenté par les bordures : U qui englobe les tuples émergents maximaux, et L qui contient tous les tuples émergents minimaux selon l'ordre de généralisation $\preceq_g$:

$$\begin{cases} L = \min_{\preceq_g}(\{t \in CL(r_1 \cup r_2) \mid C_1(t) \wedge C_2(t)\}) \\ U = \max_{\preceq_g}(\{t \in CL(r_1 \cup r_2) \mid C_1(t) \wedge C_2(t)\}) \end{cases}$$

Propriété 3.2 (Représentation du cube de données émergent par les bordures [L; U]) **-** Les bordures $[L;U]$ sont une représentation du cube de données émergent.

La figure 3.1 offre une illustration des bordures $[L;U]$ en les situant dans le treillis cube de $r_1 \cup r_2$. La partie hachurée correspond au cube de données émergent. Comme le montre cette figure, les bordures $[L;U]$ forment les frontières du cube de données émergent.

Exemple 3.6 - Avec nos relations exemples $\texttt{Diablo3}_1$ et $\texttt{Diablo3}_2$, le tableau 3.5 donne les bordures $[L;U]$ pour le cube de données émergent, en conservant une valeur similaire pour les seuils ($SeuilMin_1 = 200, SeuilMin_2 = 200$). À partir des bordures, nous savons que le tuple $(Féticheur, ALL, Bastion, Enchanteresse)$ est émergent car il spécialise le tuple $(Féticheur, ALL, ALL, ALL)$ qui appartient à la bordure L tout en généralisant le tuple $(Féticheur, Vitesse, Bastion, Enchanteresse)$ de la bordure U. De plus, le tuple $(ALL, Chance, Marche, ALL)$ n'est pas émergent. Même s'il généralise le tuple $(Sorcière, Chance, Marche, Templier)$ qui appartient à la bordure U, il ne spécialise aucun tuple de la bordure L.

La figure 3.2 représente les bordures $[L;U]$ du cube de données émergent avec $SeuilMin_1 = 200$ et $SeuilMin_2 = 200$ sur le treillis cube de la relation exemple. Pour raison de lisibilité du dessin, nous restreignons le calcul de ce cube aux dimensions `Type`, `Propriété` et `Équipement`. Les valeurs des différentes dimensions sont codées comme suit :

`Type`		`Propriété`		`Équipement`	
$Féticheur$	$= F$	$Vitesse$	$= V$	$Marche$	$= M$
$Sorcière$	$= S$	$Chance$	$= C$	$Bastion$	$= B$
$Croisé$	$= C$	ALL	$= *$	ALL	$= *$
ALL	$= *$				

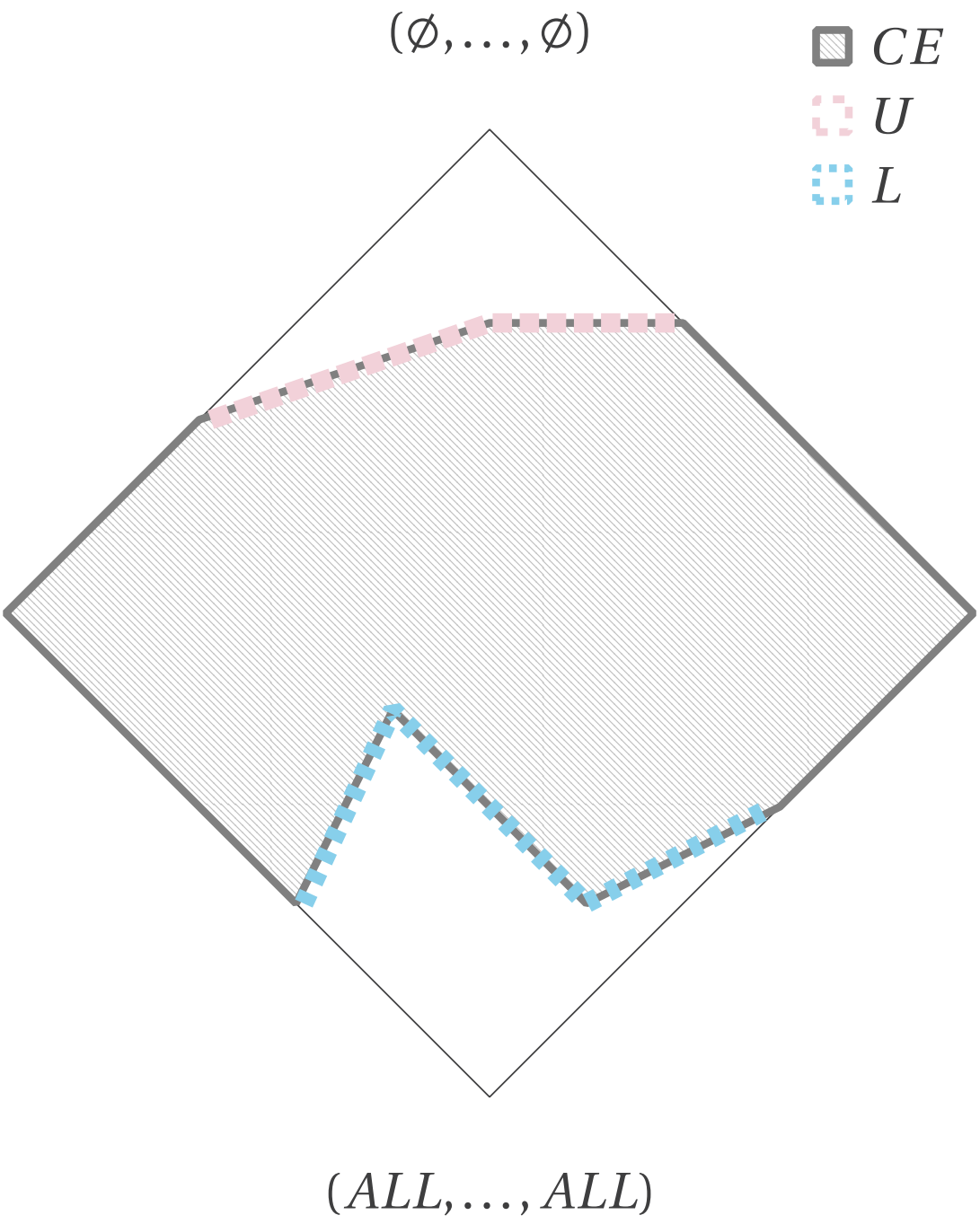


FIGURE 3.1. – Représentation du cube de données émergent par les bordures [L; U]

TABLEAU 3.5. – Bordures $[L;U]$ du cube de données émergent

U	$(Féticheur, Vitesse, Bastion, Enchanteresse)$ $(Sorcière, Vitesse, Bastion, Templier)$ $(Sorcière, Chance, Marche, Templier)$ $(Sorcière, Vitesse, Marche, Enchanteresse)$
L	$(Féticheur, ALL, ALL, ALL)$ $(Sorcière, ALL, ALL, ALL)$ $(ALL, ALL, Bastion, ALL)$ $(ALL, ALL, ALL, Templier)$ $(ALL, Vitesse, ALL, Enchanteresse)$ $(ALL, Vitesse, Marche, ALL)$

Le principe de représentation d'un ensemble convexe par des bordures Min/Max (comme $[L;U]$) est classique en apprentissage automatique (RAEDT et KRAMER 2001) mais n'est qu'assez peu étudié pour la représentation des cubes de données contraints (CASALI, NEDJAR, CICCHETTI et LAKHAL 2007). Pourtant, ces bordures offrent un résumé qui est à la fois bien plus réduit et beaucoup moins coûteux à calculer que les cubes émergents qu'elles représentent.

3.4. Calcul du cube émergent et de ses bordures

Dans ce paragraphe nous présentons les méthodes permettant d'obtenir le cube de données émergent ainsi que ses bordures. Nous les abordons avec un double point de vue, bases de données et algorithmique. Pour l'approche base de données, nous tirons profit des possibilités de SQL et spécifions différentes requêtes de calcul du cube de données émergent. Outre leur intérêt pédagogique, elles peuvent facilement être mises en œuvre lorsque les volumes de données traités sont de taille raisonnable. En revanche, c'est l'efficacité que nous visons avec l'approche algorithmique.

3.4.1. Cube de données émergent en SQL

Dans l'objectif de donner une expression SQL la plus intuitive possible du cube de données émergent, nous allons dans un premier temps donner la requête SQL calculant uniquement l'ensemble des tuples émergents. Nous utilisons dans notre requête l'opérateur `GROUP BY CUBE` (implémenté dans de nombreux SGBD), pour calculer les cubes de données de r_1 et de r_2. Une fois ces deux cubes de données calculés, nous en faisons la différence. Nous obtenons donc les tuples du cube de données de r_2, satisfaisant la seconde contrainte d'émergence C_2, privés des tuples du cube de données de r_1 ne satisfaisant pas la première contrainte d'émergence C_1.

```
01 |  SELECT  D_1, D_2, ..., D_n
02 |  FROM r_2
```

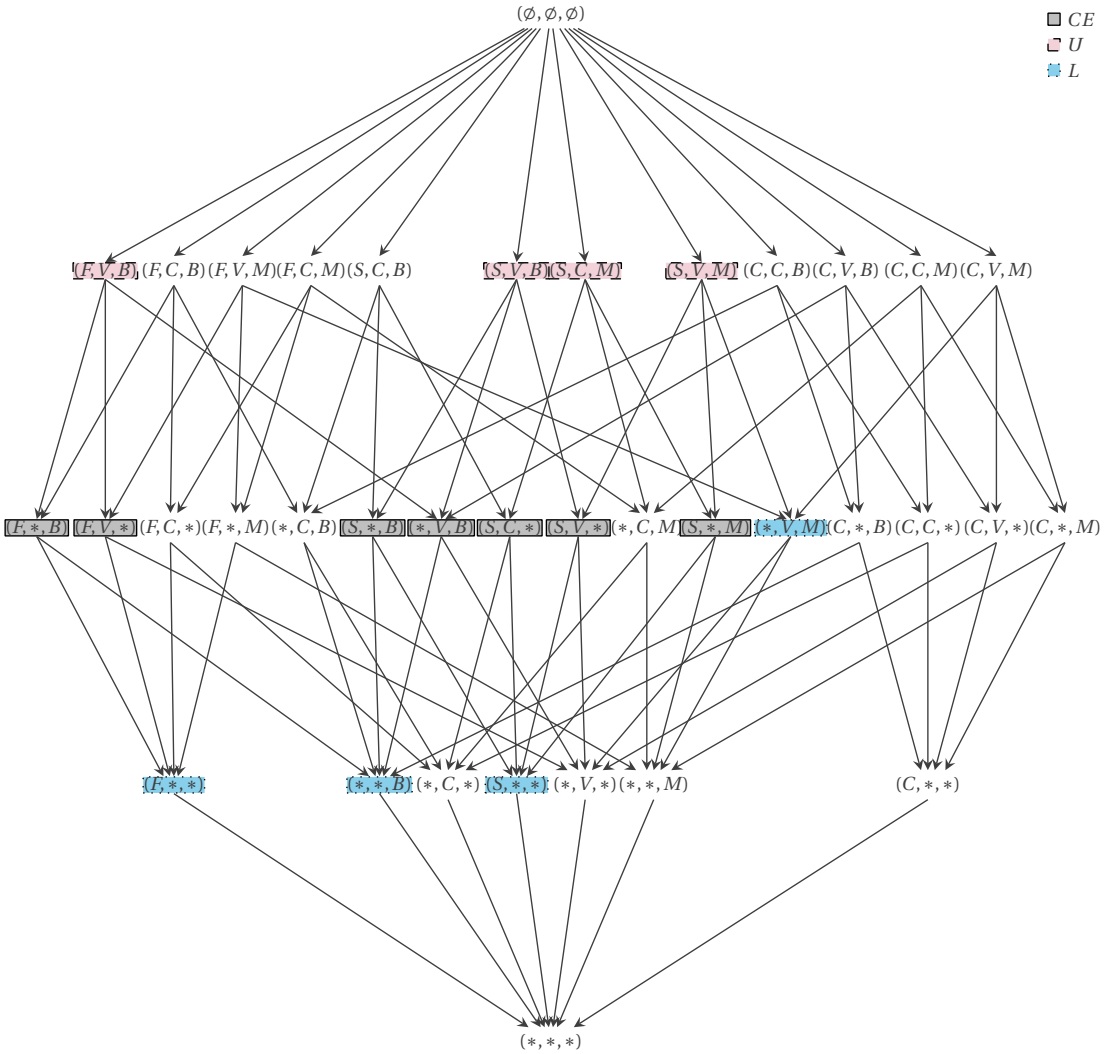


FIGURE 3.2. – Représentation du cube de données émergent et de ses bordures $[L;U]$

```
03 |  GROUP BY CUBE (D_1, D_2, ..., D_n)
04 |  HAVING f(M) >= SeuilMin_2
05 |  MINUS
06 |  SELECT D_1, D_2, ..., D_n
07 |  FROM r_1
08 |  GROUP BY CUBE (D_1, D_2, ..., D_n)
09 |  HAVING f(M) >= SeuilMin_1
```

Malheureusement, avec cette simple requête, nous ne pouvons pas retrouver la valeur du taux d'émergence. En fait, pour les tuples émergents qui sont significatifs dans r_2 et qui n'existent pas dans r_1, nous ne pouvons pas calculer la valeur du taux d'émergence à cause de l'absence de mesure pour ce tuple dans r_1. Pour contourner ce problème, nous présentons une autre requête qui calcule l'intégralité du cube de données émergent. Cette nouvelle requête est plus complexe que la précédente car elle fait une distinction entre deux cas (les deux opérandes de l'opérateur `UNION`).

1. Les tuples émergents dont on connaît toutes les mesures car ils sont présents dans les cubes de données des deux relations r_1 et r_2. Ces tuples correspondent aux basculements de tendances entre ces cubes de données.
2. Les tuples émergents qui n'existent pas dans r_1 ni dans son cube de données. Ils correspondent aux nouveautés apparaissant dans le cube de données de r_2.

```
01 |  SELECT C_2.D_1, ..., C_2.D_n, C_2.M_2/C_1.M_1 ER
02 |  FROM (SELECT D_1, ..., D_n, f({M|*}) M2
03 |  FROM r_2
04 |  GROUP BY CUBE (D_1, ..., D_n)
05 |  HAVING f({M|*}) >= SeuilMin_2) C_2,
06 |  (SELECT D_1, ..., D_n, f({M|*}) M1
07 |  FROM r_1
08 |  GROUP BY CUBE (D_1, ..., D_n)
09 |  HAVING f({M|*}) < SeuilMin_1) C_1
10 |  WHERE NVL(C_1.D_1,'ALL') = NVL(C_2.D_1,'ALL') AND ...
11 |  NVL(C_1.D_n,'ALL') = NVL(C_2.D_n,'ALL')
12 |  UNION
13 |  SELECT D_1, ..., D_n, Inf
14 |  FROM (SELECT D_1, ..., D_n
15 |  FROM r_2
16 |  GROUP BY CUBE (D_1, ..., D_n)
17 |  HAVING f({M|*}) >= SeuilMin_2)
18 |  WHERE (NVL(D_1,'ALL'), ..., NVL(D_n,'ALL')) NOT IN
19 |  (SELECT NVL(D_1,'ALL'), ..., NVL(D_n,'ALL')
20 |  FROM r_1
21 |  GROUP BY CUBE (D_1, ..., D_n)
22 |  HAVING f({M|*}) >= SeuilMin_1)
```

Dans les clauses `WHERE`, la fonction `NVL` est utilisée pour substituer la valeur *ALL* à la valeur *NULL*, qui est généralement utilisée à la place de la valeur symbolique *ALL*.

Il existe également une troisième démarche SQL, dont la requête est donnée ci-après, qui simplifie beaucoup l'expression de la requête SQL. L'idée de base est de se dispenser du calcul des deux cubes de données en effectuant une « fusion » des deux

relations opérandes (DIOP, GIACOMETTI, LAURENT et al. 2002). Le résultat de cette fusion est une nouvelle relation r dont le schéma est le suivant : $\mathscr{R} = \mathscr{D} \cup \mathscr{M}_1 \cup \mathscr{M}_2$ où $\mathscr{M}_1$ est l'attribut mesure de r_1 et $\mathscr{M}_2$ celui de r_2. Tout tuple de r_1 est considéré dans r avec la valeur 0 pour l'attribut mesure M_2 et inversement tout tuple de r_2 se voit affecter la valeur 0 pour la mesure M_1. Le fait de dissocier les deux mesures en considérant deux attributs différents permet de réaliser l'agrégation de r en conservant les agrégats obtenus pour r_1 et ceux obtenus pour r_2 en quelque sorte en les concaténant.

```
01 | SELECT D_1, D_2, ..., D_n, SUM(M2)/NULLIF(SUM(M1),0) ER
02 | FROM (SELECT D_1, D_2, ..., D_n, M M1, 0 M2
03 | FROM r_1
04 | UNION
05 | SELECT D_1, D_2,..., D_n, 0 M1, M M2
06 | FROM r_2)
07 | GROUP BY CUBE (D_1, D_2,..., D_n)
08 | HAVING SUM(M1) < SeuilMin_1 AND SUM(M2) >= SeuilMin_2
```

Remarquons que la relation fusionnée r est une relation temporaire calculée dans la clause `FROM` en réalisant l'union des tuples de r_1 et de r_2. Il suffit alors, dans le premier bloc, de procéder au calcul du cube de données et de spécifier la double contrainte d'émergence dans la clause `HAVING`. Pour éviter le problème de la division par 0 lors du calcul du taux d'émergence, la fonction `NULLIF` qui substitue au dénominateur la valeur *NULL* lorsqu'il est égal à 0, est utilisée. Ainsi, nous obtenons la valeur *NULL* correspondant à un taux d'émergence infini.

Exemple 3.7 - Avec nos relations exemples $\texttt{Diablo3}_1$ et $\texttt{Diablo3}_2$, la requête par fusion est la suivante :

```
01 | SELECT Type, Propriété, Équipement, Compagnon,
02 |        SUM(M2)/NULLIF(SUM(M1),0) ER
03 | FROM (SELECT Type, Propriété, Équipement, Compagnon,
04 |              Quantité M1, 0 M2
05 |       FROM Diablo3_1
06 |       UNION
07 |       SELECT Type, Propriété, Équipement, Compagnon,
08 |              0 M1, Quantité M2
09 |       FROM Diablo3_2)
10 | GROUP BY CUBE (Type, Propriété, Équipement, Compagnon)
11 | HAVING SUM(M1) < 300 AND SUM(M2) >= 300
```

Les résultats de cette requête sont donnés dans le tableau 3.6.

3.4.2. L'algorithme du cube de données émergent : E-IDEA

Pour l'instant, il n'existe aucun moyen intégré aux SGBD pour calculer les bordures. Même à partir des ensembles de tuples retournés par les requêtes proposées, on ne sait pas extraire les maximaux/minimaux selon l'ordre de généralisation.

Pour que le cube de données émergent soit un opérateur directement exploitable par l'utilisateur, les algorithmes de calcul que nous présentons doivent être intégrables

TABLEAU 3.6. – Cube émergent calculé par la requête avec fusion des relations

Tuple émergent	*ER*
(*ALL*, *ALL*, *ALL*, *Templier*)	3
(*ALL*, *ALL*, *Bastion*, *ALL*)	8
(*ALL*, *ALL*, *Bastion*, *Enchanteresse*)	5
(*ALL*, *Vitesse*, *ALL*, *ALL*)	3.3
(*ALL*, *Vitesse*, *ALL*, *Enchanteresse*)	7
(*ALL*, *Vitesse*, *Bastion*, *ALL*)	6
(*ALL*, *Vitesse*, *Marche*, *ALL*)	2
(*ALL*, *Vitesse*, *Marche*, *Enchanteresse*)	*NULL*
(*Sorcière*, *ALL*, *ALL*, *ALL*)	4.5
(*Sorcière*, *ALL*, *ALL*, *Templier*)	6
(*Sorcière*, *ALL*, *Marche*, *ALL*)	3
(*Sorcière*, *Vitesse*, *ALL*, *ALL*)	6

directement dans les SGBD. Avec une telle approche relationnelle, il est possible tirer pleinement partie des outils d'analyse ROLAP [1] existants. Le cube de données émergent n'est ainsi qu'un cube de données particulier et, comme cela est aussi faisable avec le cube de données originel, il est possible de l'interroger, l'explorer et y naviguer.

L'algorithme BUC (BEYER et RAMAKRISHNAN 1999) et ses différentes variantes (MORFONIOS et IOANNIDIS 2008 ; PHANLUONG 2016) ont montré toute leur efficacité aussi bien pour le calcul des cubes de données iceberg que pour des cubes de données complets. E-IDEA s'appuie sur l'algorithme BUC pour proposer une solution logicielle homogène capable de calculer le cube de données émergent aussi bien que ses différentes représentations. L'algorithme BUC est présenté dans le paragraphe 2.2.5.2 p. 46.

Dans cette sous-section, nous nous intéressons à l'adaptation du schéma algorithmique de BUC pour le calcul du cube de données émergent. Cette approche tire profit de la structure de la relation « fusionnée » présentée à la sous-section 3.4.1. C'est véritablement cette structure de relation concaténant les différentes mesures qui nous permet d'adapter des algorithmes de calcul de cube de données au contexte de cube de données émergent.

Exemple 3.8 - Avec nos relations exemples $\texttt{Diablo3}_1$ et $\texttt{Diablo3}_2$, la relation « fusionnée » est donnée dans le tableau 3.7 (avec $\texttt{Q}_1$ qui correspond à la mesure `Quantité` de la relation exemple $\texttt{Diablo3}_1$ et $\texttt{Q}_2$ qui correspond à la mesure `Quantité` de la relation exemple $\texttt{Diablo3}_2$). La relation résultat a les même attributs dimensions que les relations en entrée, mais possède deux attributs mesures. L'attribut mesure

1. ROLAP pour *relational* OLAP, méthodologie reposant sur la manpipulation de données enregistrées en base de données relationnelle afin de donner l'impression des fonctionnalités de partitionnement de OLAP.

Quantité$_1$ correspond à Quantité pour la relation Diablo3$_1$ et Quantité$_2$ correspond à Quantité pour la relation Diablo3$_2$. C'est cette relation qui sert d'entrée aux algorithmes proposés.

TABLEAU 3.7. – Relation « fusionnée » Diablo3$_{Fus}$

RowId	Type	Propriété	Équipement	Compagnon	Q_1	Q_2
1	*Féticheur*	*Vitesse*	*Bastion*	*Enchanteresse*	100	0
2	*Féticheur*	*Vitesse*	*Bastion*	*Enchanteresse*	0	300
3	*Féticheur*	*Vitesse*	*Marche*	*Templier*	100	0
4	*Sorcière*	*Vitesse*	*Bastion*	*Templier*	0	300
5	*Sorcière*	*Vitesse*	*Marche*	*Enchanteresse*	0	300
6	*Sorcière*	*Vitesse*	*Marche*	*Templier*	100	0
7	*Sorcière*	*Chance*	*Marche*	*Templier*	0	300
8	*Sorcière*	*Chance*	*Marche*	*Enchanteresse*	100	0
9	*Croisé*	*Vitesse*	*Marche*	*Enchanteresse*	0	100
10	*Croisé*	*Chance*	*Marche*	*Enchanteresse*	600	0
11	*Croisé*	*Chance*	*Marche*	*Enchanteresse*	0	200
12	*Croisé*	*Chance*	*Bastion*	*Enchanteresse*	0	200

L'algorithme E-IDEA est une solution algorithmique inspirée de l'approche de BUC prenant en compte la contrainte d'émergence. Cette dernière est la conjonction de deux contraintes : l'une est monotone (C_1) et l'autre antimonotone (C_2). Le parcours du treillis cube utilisé par BUC favorise la contrainte antimonotone. En effet, comme il traite les tuples des plus agrégés vers les moins agrégés, c'est la seconde contrainte qui impose l'arrêt de l'algorithme.

Au début de son traitement, les tuples multidimensionnels considérés ne satisfont pas la contrainte C_1 mais vérifient bien C_2 ; ces tuples sont donc non émergents, ils doivent être ignorés. Une fois tous ces tuples traités, E-IDEA passe aux tuples émergents qui, eux, devront être retenus. Puis, il sort de l'espace des résultats lorsque C_2 n'est plus satisfaite pour retomber sur des tuples non émergents. La contrainte C_2 n'étant plus vérifiée, tous les successeurs ne sont plus émergents, l'algorithme s'arrête. La figure 3.3 illustre sur un treillis les différentes étapes rencontrées par l'algorithme durant le parcours de l'espace de recherche.

Pour traiter les différents cas possibles, l'algorithme fonctionne par phases et pour chacune de ces phases, il a un comportement spécifique. Comme le montre la figure 3.3, il y a en tout trois phases principales (P_1, P_2 et P_3), chacune correspondant à l'état (vérifié ou non vérifié) des contraintes C_1 et C_2. Le tableau ci-dessous donne la correspondance entre état des contraintes et la phase dans laquelle l'algorithme se trouve :

	C_2	$\neg C_2$
C_1	P_2	P_3
$\neg C_1$	P_1	

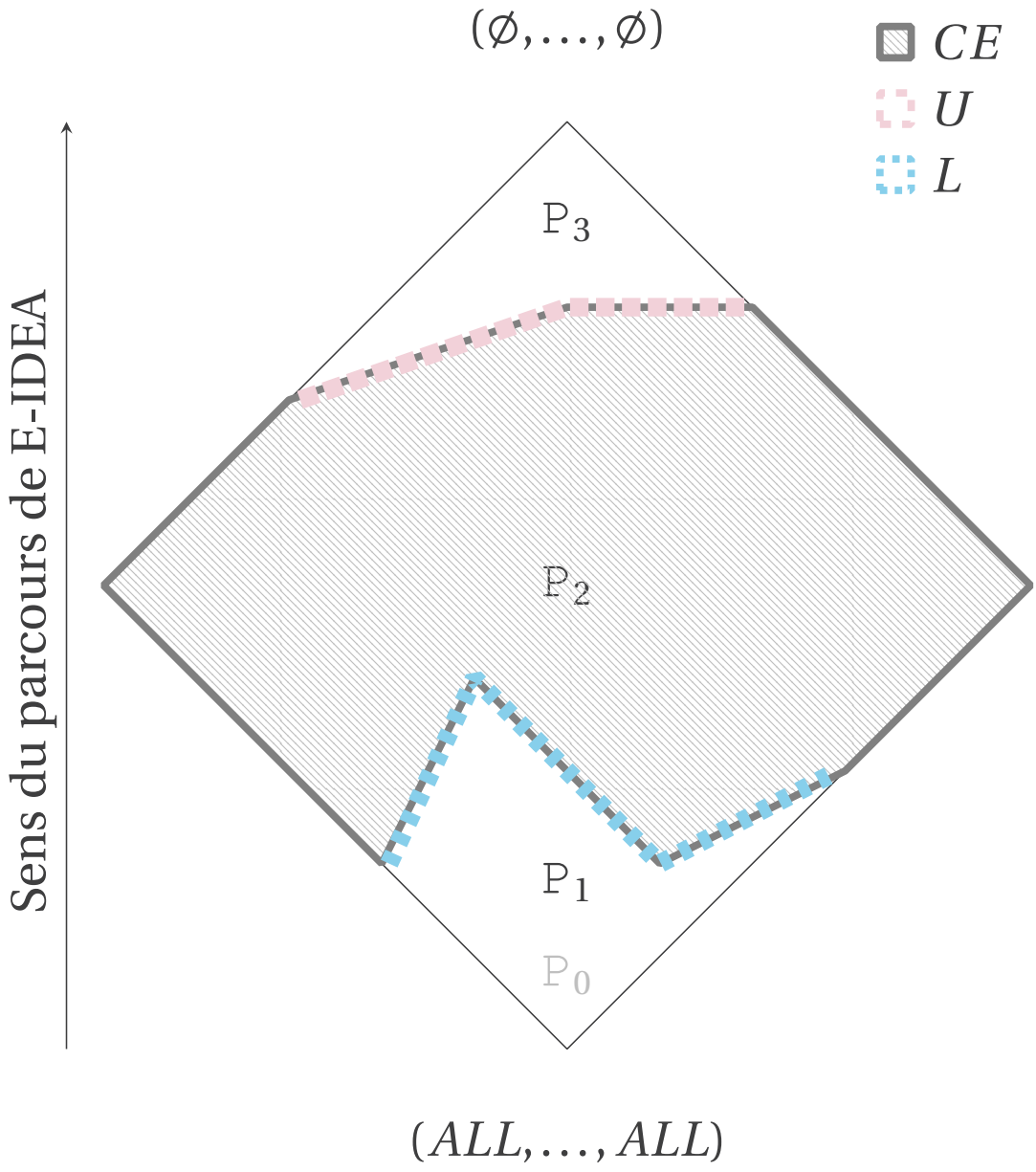


FIGURE 3.3. – Illustration des différentes phases d'E-IDEA

Comme souligné ci-dessus, C_1 et C_2 ne sont pas de nature quelconque. Le parcours en profondeur de I-IDEA implique que la contrainte monotone n'est pas satisfaite au début de l'algorithme puis le deviendra plus tard. Inversement, la contrainte antimonotone est satisfaite au début puis, par la suite, ne l'est plus. Ainsi, les transitions de phases ne peuvent toutes se produire. Par exemple si l'algorithme est dans la phase P_3 (contrainte C_2 n'est plus satisfaite), il ne pourra plus revenir dans l'une ou l'autre des phases. L'automate donné dans la figure 3.4 décrit les différentes transitions possibles.

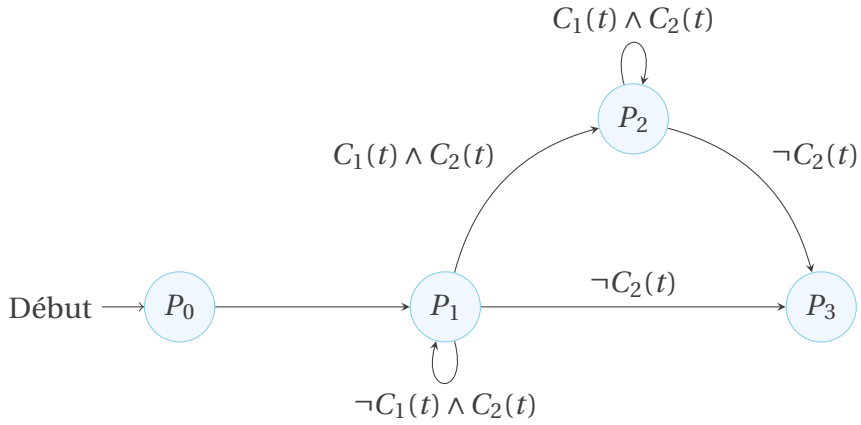


FIGURE 3.4. – Automate de transitions de phases de l'algorithme E-IDEA

Voici les phases de l'algorithme :

— P_0 : c'est la phase d'élimination des duplicats, qui n'existe que lors du premier appel de l'algorithme pour une même relation ; contrairement au cas du cube de données pour lequel les relations contiennent rarement des tuples dupliqués, la relation fusionnée d'E-IDEA en contient le plus souvent un grand nombre (en effet, lorsque l'on fusionne les deux relations, les tuples qui sont présents à la fois dans r_1 et r_2 se retrouvent deux fois dans la relation résultat r, la première occurrence avec une valeur uniquement pour l'attribut mesure $\mathscr{M}_1$ et la seconde uniquement pour $\mathscr{M}_2$), ce qui a pour effet d'annuler la forte optimisation obtenue par l'appel d'ÉcrireSuivants vu que cette procédure n'est appelée que lorsque les partitions ne contiennent plus qu'un seul élément (aussi appelé *base single tuple*) : pour supprimer ces duplicats, il suffit de les pré-agréger en les combinant en un seul et même tuple ayant une mesure pour les deux attributs $\mathscr{M}_1$ et $\mathscr{M}_2$.

— P_1 : c'est la véritable phase initiale, l'état dans lequel se trouve l'algorithme au démarrage lors de tous les appels ultérieurs au premier pour une même relation ; durant cette phase, les tuples considérés ne sont pas émergents car la contrainte C_1 n'est pas encore vérifiée ; l'algorithme se contente juste de partitionner la relation sans écrire aucun résultat ; E-IDEA change de phase dès lors que l'une des deux contraintes change d'état.

— P_2 : c'est la phase principale durant laquelle tous les tuples rencontrés sont émergents ; E-IDEA doit donc écrire chacun d'entre eux sur disque ; ces opérations

d'entrées/sorties étant très coûteuses, c'est l'étape la plus consommatrice en temps de l'algorithme ; cette étape se termine quand un tuple ne vérifiant plus la contrainte C_2 est rencontré ; dans ce cas, l'algorithme passe en phase 3.

— P_3 : c'est la phase finale, l'algorithme arrive dans cette phase lorsque la contrainte C_2 n'est plus vérifiée pour le tuple courant t ; cette contrainte étant antimonotone elle ne peut plus être vérifiée pour tous les tuples plus spécifiques que t ; il n'y a donc plus besoin de parcourir l'espace de recherche, et l'algorithme se termine alors.

Exemple 3.9 - Concernant la phase P_0, avec les relations exemples, nous avons le tuple $(Féticheur, Vitesse, Bastion, Enchanteresse)$ dans $\texttt{Diablo3}_1$ et dans $\texttt{Diablo3}_2$. Il figure donc en double dans la relation fusionnée (cf. tableau 3.7). Pour supprimer ce doublon, il faut remplacer $(Féticheur, Vitesse, Bastion, Enchanteresse)100 : 0$ et $(Féticheur, Vitesse, Bastion, Enchanteresse)0 : 300$ par le tuple « combiné » $(Féticheur, Vitesse, Bastion, Enchanteresse)100 : 300$. La relation fusionnée dédoublonnée est donnée dans le tableau 3.8 (avec $\texttt{Q}_1$ pour $\texttt{Quantité}_1$ et $\texttt{Q}_2$ pour $\texttt{Quantité}_2$).

TABLEAU 3.8. – Relation fusionnée dédoublonnée $\texttt{Diablo3}^{+}_{Fus}$

RowId	Type	Propriété	Équipement	Compagnon	$\texttt{Q}_1$	$\texttt{Q}_2$
1	*Féticheur*	*Vitesse*	*Bastion*	*Enchanteresse*	100	300
2	*Féticheur*	*Vitesse*	*Marche*	*Templier*	100	0
3	*Sorcière*	*Vitesse*	*Bastion*	*Templier*	0	300
4	*Sorcière*	*Vitesse*	*Marche*	*Enchanteresse*	0	300
5	*Sorcière*	*Vitesse*	*Marche*	*Templier*	100	0
6	*Sorcière*	*Chance*	*Marche*	*Templier*	0	300
7	*Sorcière*	*Chance*	*Marche*	*Enchanteresse*	100	0
8	*Croisé*	*Vitesse*	*Marche*	*Enchanteresse*	0	100
9	*Croisé*	*Chance*	*Marche*	*Enchanteresse*	600	200
10	*Croisé*	*Chance*	*Bastion*	*Enchanteresse*	0	200

3.4.2.1. Description de l'algorithme E-IDEA

A l'exception de sa phase d'élimination des duplicats, E-IDEA est une adaptation du schéma algorithmique de BUC dédié au calcul de cube de données émergent. Il utilise les mêmes algorithmes que BUC (cf. paragraphe 2.2.5.2 p. 46) :

— BUC (cf. algorithme 4 p. 50), l'algorithme lui-même ;
— Agréger (cf. algorithme 2 p. 49), qui calcule la valeur de la mesure pour la partition courante ;
— ÉcrireSuivants (cf. algorithme 3 p. 49), qui permet d'écrire les résultats directement sur disque lorsque le cube de données de la partition courante est trivial à calculer ;

— Partitionner (cf. algorithme 5 p. 51) qui découpe efficacement la relation en partitions en fonction des valeurs de l'attribut courant (cette opération est réalisée grâce à un tri par comptage de complexité linéaire).

Des modifications ne sont apportées qu'à Agréger ainsi que BUC, et l'élimination des duplicats fait son apparition :

— l'algorithme Agréger modifié est donné dans l'algorithme 6 ; la relation fusionnée ayant deux attributs mesures, il faut calculer la valeur de la fonction agrégative sur chacun de ces attributs ; à partir de ces deux valeurs, il est toujours possible de dériver le taux d'émergence associé au tuple multidimensionnel courant ;

— l'algorithme E-IDEA (cf. algorithme 7) reprend le même schéma algorithmique que BUC mais en adaptant son comportement en fonction de la phase dans laquelle il se trouve grâce à PhaseSuiv ;

— pour éviter de devoir ajouter un test inutile dans tous les appels sauf le premier, une nouvelle procédure nommée E-IDEA_P_0 est ajoutée : elle est pratiquement la même que celle d'IDEA, mais l'appel à Partitionner est remplacée par celui à PartitionnerEtDédupliquer, qui comme son nom l'indique, va à la fois partitionner et profiter du tri effectué (lors du premier partitionnement) pour éliminer les duplicats. La procédure E-IDEA_P_0 est donnée dans l'algorithme 8.

Algorithme 6 Algorithme Agréger (pour E-IDEA)

```
Entrée :
   Une relation fusionnée r.
   La cellule courante CelluleCour.
   Une fonction agrégative additive f.
01: soit r1 la projection de r sur D ∪ M1
02: soit r2 la projection de r sur D ∪ M2
03: pour tout t ∈ r faire
04:   CelluleCour.M1 = CelluleCour.M1 + f(t, r1);
05:   CelluleCour.M2 = CelluleCour.M2 + f(t, r2);
06: fin pour
```

3.4.3. F-IDEA

A notre connaissance, F-IDEA (pour *frontier integrable database algorithm*) est une des premières solutions logicielles génériques permettant d'obtenir les bordures du cube de données émergent. F-IDEA est également basé sur BUC, comme E-IDEA, pour les même raisons. Lors du calcul du cube de données émergent, E-IDEA utilise la majeure partie du temps d'exécution pour réaliser les sorties, *i.e.* l'écriture du cube de données sur disque. E-IDEA a donc été adapté de manière à n'écrire que les tuples des bordures. De plus, grâce à une structure de données adaptée, l'écriture multiple des mêmes tuples est évitée. Dans un premier temps, nous présentons l'algorithme calculant la bordure U puis nous montrons les modifications à lui apporter pour obtenir la bordure L.

Algorithme 7 Algorithme E-IDEA

Entrée :
Une relation fusionnée r.
L'ensemble des dimensions $\mathscr{D} = \{d_1, \ldots, d_n\}$ à traiter.
La cellule courante $CelluleCour$.
La phase courante de l'algorithme P_{Cour}.
Sortie :
Le cube de données émergent de r.
La phase de sortie de l'algorithme.

```
01: DimCour := d_1;
02: Agréger(r, CelluleCour, f); //Calcule les valeurs de la
    fonction mesure et les écrit dans CelluleCour
03: soit P_Suiv := PhaseSuiv(P_Cour, CelluleCour); //cf. figure 3.4
04: si P_Suiv = P_3 alors
05:   retourner P_Suiv;
06: sinon si P_Suiv = P_2 alors
07:   si |r| = 1 alors
08:     ÉcrireSuivants(r[0], D, CelluleCour); //La relation n'a
        qu'un tuple, on écrit les résultats sans agréger
09:     retourner P_3; //Les successeurs sont tous écrits,
        on passe à la phase finale P_3
10:   fin si
11:   Écrire(CelluleCour);
12: fin si
13: pour tout d_j ∈ D faire
14:   C := |r(d_j)|;
15:   Partitionner(r, d_j); //r est partitionnée suivant d_j en
      C partitions : r_1, ..., r_C
16:   pour i = 1, ..., C faire
17:     //CelluleCour reçoit la valeur de d_j pour r_i
18:     CelluleCour.d_j := r_i[0].d_j;
19:     E-IDEA(r_i, {d_{j+1}, ..., d_n}, CelluleCour, P_Suiv);
20:   fin pour
21:   CelluleCour.d_j := ALL; //d_j est entièrement traitée
22: fin pour
23: retourner P_Suiv;
```

Algorithme 8 Algorithme E-IDEA_P_0

Entrée :
Une relation fusionnée r.
L'ensemble des dimensions $\mathscr{D} = \{d_1, \ldots, d_n\}$ à traiter.
La cellule courante $CelluleCour$.
Sortie :
Le cube de données émergent de r.
La phase de sortie de l'algorithme.

```
01: DimCour := d_1;
02: Agréger(r, CelluleCour, f); //Calcule les valeurs de la
    fonction mesure et les écrit dans CelluleCour
03: si C_1((ALL,...,ALL)) alors
04:   BUC(r, D, (ALL,...,ALL));
05:   //Si C_1 est satisfaite dès (ALL,...,ALL), le résultat
      est le cube de données iceberg de r
06:   retourner P_0;
07: fin si
08: pour tout d_j ∈ D faire
09:   C := |r(d_j)|;
10:   PartitionnerEtDédupliquer(r, d_j); //r est partitionnée
      suivant d_j en C partitions : r_1,...,r_C
11:   pour i = 1,...,C faire
12:     CelluleCour.d_j := r_i[0].d_j; //CelluleCour reçoit la valeur
        de d_j pour r_i
13:     E-IDEA(r_i, {d_{j+1},...,d_n}, CelluleCour, P_1);
14:   fin pour
15:   CelluleCour.d_j := ALL; //d_j est entièrement traitée
16: fin pour
17: retourner P_0;
```

Sur la figure 3.3, on remarque que les différentes bordures délimitent les phases de l'algorithme E-IDEA. En d'autres termes, les tuples des bordures correspondent à des tuples pour lesquels l'algorithme change de phase. Nous présentons à quel type de transition est associé chaque bordure après avoir donné la définition de plusieurs concepts nous permettant de mieux les formaliser, et qui serviront également pour la suite de ce document.

Définition 3.6 (Ensemble en accord) - Soit $P \subseteq r$ un sous-ensemble de tuples de la relation r et $D \subseteq \mathscr{D}$ un ensemble de dimensions, P est en accord sur l'ensemble D si et seulement si $\forall t, t' \in P$ on a $t[D] = t'[D]$.

Définition 3.7 (IDEA-Successeur) - Soit $D = \{d_1, \ldots, d_j\}$ un sous-ensemble ordonné de dimensions ($D \subseteq \mathscr{D}$) et soit P un ensemble de tuples de la relation r en accord sur D. Un ensemble de tuples P' est dit IDEA-Successeur de P par rapport à D si et seulement si $P' \subseteq P$ et $\exists d_k \in \mathscr{D} \backslash D$ (avec $k > j$) tel que $\forall t, t' \in P'$, $t.d_k = t'.d_k$.

Autrement dit, P' est un IDEA-Successeur de P par rapport à D si P' est un sous-ensemble de P qui est en accord sur un sur-ensemble direct de D dont la dimension supplémentaire d_k est plus grande (selon $\geq_{\mathscr{D}}$) que la plus grande des dimensions de D (d_j dans la définition ci-dessus).

Exemple 3.10 - En prenant la relation exemple fusionnée dédoublonnée $\texttt{Diablo3}^{+}_{Fus}$ (cf. tableau 3.8), soit $P = \texttt{Diablo3}^{+}_{Fus} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ un ensemble de tuples en accord sur $D = \emptyset$, l'ensemble de tuples $P' = \{t_1, t_2, t_3, t_4, t_5\}$ est en accord sur $D' = \{\texttt{Propriété}\}$ un sur-ensemble direct de D. P' est donc un IDEA-Successeur de P par rapport à D.

Définition 3.8 (IDEA-Successeur maximal) - Soit $D = \{d_1, \ldots, d_j\}$ un sous-ensemble ordonné de dimensions ($D \subseteq \mathscr{D}$) et soit P un ensemble de tuples d'une relation r tels que $\forall t, t' \in P$ on a $t[D] = t'[D]$. Une partition P' est dit IDEA-Successeur maximal de P par rapport à D si et seulement si P' est un IDEA-Successeur de P par rapport à D et $\nexists t \in P \backslash P'$ tel que $t \cup P'$ soit un IDEA-Successeur de P par rapport à D.

Cette notion de IDEA-Successeur maximal permet de formaliser clairement l'ordre dans lequel les algorithmes BUC et E-IDEA parcourent les partitions de la relation d'entrée (cf. figure 2.9 p. 46). Ainsi, dire qu'une partition P' est un IDEA-Successeur maximal de P par rapport à D revient à dire que si l'on fait un appel à E-IDEA avec comme arguments P et D, alors on est certain que l'un des appels récursifs directs prend comme paramètre le fragment P' et l'ensemble de dimensions $D \cup d_k$. Si l'on se préoccupe de l'arbre des appels récursifs, cela revient à dire qu'il existe une arête qui relie les deux appels d'arguments (P, D) et (P', $D \cup d_k$).

Exemple 3.11 - D'après l'exemple précédent, $P' = \{t_1, t_2, t_3, t_4, t_5\}$ est IDEA-Successeur de $P = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ par rapport à $D = \emptyset$. P' n'est pas IDEA-Successeur maximal car t_8 partage les même valeurs que les tuples de P' sur D'. $P' \cup t_8$ est, quant à lui, un IDEA-Successeur maximal de P par rapport à D car il n'existe au-

cun tuple partageant les mêmes valeurs pour les attributs de D'. Le tuple multidimensionnel associé à P est (ALL, ALL, ALL, ALL). Celui associé à $P' \cup t_8$ est le tuple $(ALL, Vitesse, ALL, ALL)$.

Lors de l'exécution d'E-IDEA, les transitions de phases ont lieu juste après un appel récursif à l'algorithme. Si pour un appel à E-IDEA avec comme arguments P et D, l'algorithme est en phase P_i et que lors du traitement de l'un de ses IDEA-Successeurs maximaux P', il est en phase P_j, nous considèrerons que P' est issu d'une transition $\mathtt{P}_i \rightarrow \mathtt{P}_j$ et que P est source d'une transition $\mathtt{P}_i \rightarrow \mathtt{P}_j$.

Proposition 3.1 - Soit un tuple $t \in L$, l'ensemble de tuples P' ayant servi au calcul de t est issu d'une transition $\mathtt{P}_1 \rightarrow \mathtt{P}_2$.

Proposition 3.2 - Soit un tuple $t \in U$, l'ensemble de tuples P ayant servi au calcul de t n'est source que de transitions $\mathtt{P}_2 \rightarrow \mathtt{P}_3$.

Ces deux propriétés sont fondamentales pour caractériser les algorithmes capables de dériver les différentes bordures. Intégrer l'une de ces propriétés au sein de l'algorithme E-IDEA nous permet de calculer la bordure correspondante. L'algorithme découlant de cette intégration est appelé F-IDEA. F-IDEA n'est pas seulement un algorithme capable de calculer une unique bordure, mais il apporte une solution algorithmique générique capable de calculer n'importe quelle bordure ou combinaison de bordures, dès l'intégration des propriétés correspondantes.

Nous ne présentons la démarche d'adaptation de E-IDEA en F-IDEA uniquement pour la bordure L (cf. algorithme 9). La première des modifications à apporter est de supprimer toutes les opérations effectuées pour les tuples générés pendant la phase P_2 car tous les tuples de L sont produits durant la phase P_1. La deuxième des différences se situe au niveau de la gestion des *base single tuples*. La partition de ces tuples est constituée d'un simple tuple, ainsi pour un tel élément il n'est pas besoin de considérer tous ses successeurs mais juste le plus grand d'entre eux. La projection du tuple sur l'ensemble des dimensions restant à traiter est alors ajoutée à la bordure. La dernière des modifications à faire concerne la vérification de la condition donnée par la propriété 3.1. Après chaque appel récursif on vérifie que la transition est $\mathtt{P}_1 \rightarrow \mathtt{P}_2$. Si aucune transition de ce type n'est rencontrée, le tuple courant est testé et ajouté à la bordure L. La procédure AjouterABordure se contente de vérifier la « maximalité » du tuple et de l'ajouter au résultat. Comme nous pouvons le voir sur cette version de F-IDEA, les adaptations à faire sont toutes induites par les propositions 3.1 et 3.2. Il y a effectivement des optimisations plus spécifiques au calcul de chaque bordure, mais l'algorithme F-IDEA perdrait en généricité. Malgré cela, cet algorithme est très efficace (cela a été démontré expérimentalement dans NEDJAR 2009).

3.5. Cube de données fermé émergent

Offrir aux décideurs des cubes de données émergents est loin d'être trivial car deux cubes de données, probablement volumineux donc coûteux à obtenir, à enregistrer et

Algorithme 9 Algorithme F-IDEA (pour la bordure L)

Entrée :
Une relation fusionnée r.
L'ensemble des dimensions $\mathscr{D} = \{d_1, \ldots, d_n\}$ à traiter.
La cellule courante $CelluleCour$.
La phase courante de l'algorithme P_{Cour}.
Sortie :
La bordure L du cube de données émergent de r.
La phase sortie de l'algorithme.

```
01: DimCour := d_1;
02: Agréger(r, CelluleCour, f); //Calcule les valeurs de la
    fonction mesure et les écrit dans CelluleCour
03: soit P_Suiv := PhaseSuiv(P_Cour, CelluleCour); //cf. figure 3.4
04: si (P_Suiv = P2) ∨ (P_Suiv = P3) alors
05:   retourner P_Suiv; //On ne cherche que les tuples de L,
      pas besoin de continuer
06: sinon si |r| = 1 alors
07:   soit t := r[0];
08:   AjouterABordure(t[D]); //r ne contient que t, on
      l'ajoute dans L sans traiter ses prédécesseurs
09:   retourner P_1;
10: fin si
11: pour tout d_j ∈ D faire
12:   C := |r(d_j)|;
13:   Partitionner(r, d_j); //r est partitionnée suivant d_j en
      C partitions : r_1, ..., r_C
14:   soit TestL : booléen du test de la proposition 3.1
15:   TestL := (P_Suiv == P_1); //La source de la transition est P_1
16:   pour i = 1, ..., C faire
17:     //CelluleCour reçoit la valeur de d_j pour r_i
18:     CelluleCour.d_j := r_i[0].d_j;
19:     P_Res := F-IDEA(r_i, {d_{j+1}, ..., d_n}, CelluleCour, P_Suiv);
20:     TestL := TestL ∧ (P_Res = P2) ;
21:   fin pour
22:   si TestL alors
23:     AjouterABordure(CelluleCour);
24:   fin si
25:   CelluleCour.d_j := ALL; //d_j est entièrement traitée
26: fin pour
27: retourner P_Suiv;
```

à gérer, doivent être calculés pour ensuite être comparés. Nous avons déjà présenté des représentations réduites pour les cubes de données émergents qui s'appuient sur les bordures (NEDJAR, CASALI, CICCHETTI et al. 2008 ; NEDJAR, CASALI, CICCHETTI et al. 2006a ; NEDJAR, CASALI, CICCHETTI et al. 2006b). Néanmoins, ces représentations ne permettent pas de retrouver les valeurs des mesures et donc ne peuvent pas être utilisées pour répondre à n'importe quelle requête OLAP.

Dans cette section, nous présentons le cube de données fermé émergent, qui offre selon nous les représentations de cube de données émergent éliminant les redondances sans perte d'information parmi les plus réduites, et qui s'appuie sur le cube de données fermé (cf. paragraphe 2.3.3 p. 69). À partir de cette représentation, il est possible de retrouver les valeurs des mesures associées aux différentes tendances. Ainsi, le résultat de toute requête OLAP peut en être dérivé. Cette représentation englobent l'ensemble des tuples fermés émergents auquel est ajoutée toute l'information nécessaire pour garantir une représentation sans perte. Cette information est la bordure L (cf. paragraphe 3.3.1 p. 93), d'où le L-cube de données fermé émergent (NEDJAR, CASALI, CICCHETTI et al. 2010b).

3.5.1. L-cube de données fermé émergent

Le L-cube de données fermé émergent inclut à la fois :
— l'ensemble des tuples fermés émergents ;
— la bordure L.

Cette approche est conçue dans le même esprit que celle proposée par GOLLAPUDI et SIVAKUMAR 2004 dans le contexte de bases de données de transactions et qui englobe les motifs fermés contraints et la bordure minimale L.

Pour une raison de simplicité, nous utilisons, à partir de maintenant t au lieu de $(t, f_{val}(t,r))$ pour indiquer un tuple complet avec ses valeurs des dimensions et sa mesure.

Cette représentation a pour objectif d'éliminer les redondances existant au sein des cubes de données émergents. Effectivement, certains tuples véhiculent la même sémantique que d'autres plus agrégés. En fait, les uns et les autres sont construits en agrégeant exactement les mêmes tuples de la relation originale mais à des niveaux de granularité différents. Donc un seul tuple, le plus spécifique de tous, peut représenter tout l'ensemble. L'opérateur de fermeture cubique est conçu pour calculer ce tuple représentatif. Pour définir cette représentation, nous avons besoin des concepts suivants :

— connexion cubique (cf. définition 2.17 p. 69) ;
— opérateur de fermeture cubique(cf. définition 2.18 p. 69) : soit $T \subseteq CL(r)$ un ensemble de tuples, l'opérateur de fermeture cubique $\mathbb{C} : CL(r) \rightarrow CL(r)$ selon T, peut également être défini ainsi :

$$\mathbb{C}(t,T) = \sigma \circ \lambda(t) = (\emptyset, \ldots, \emptyset) + \sum_{\substack{t' \in T, \\ t \preceq_g t'}} t'$$

Où l'opérateur $\sum$ a la même sémantique que l'opérateur +; considérons tous les tuples t' dans T, agrégeons les ensemble en utilisant l'opérateur + : nous obtenons un nouveau tuple qui généralise tous les tuples t' et qui est le plus spécifique, ce nouveau tuple est la Fermeture Cubique de t.

Exemple 3.12 - Dans les exemples de cette section, nous cherchons les renversements de tendances s'étant produits entre la `Saison` 19 et la `Saison` 20 pour les personnages répertoriés dans la relation `Diablo3` (cf. tableau 3.9), que nous reprenons et à laquelle nous ajoutons le `Compagnon` *Brigand*. La relation $\texttt{Diablo3}_1$ (cf. tableau 3.10) correspond aux héros pour lesquels `Saison` = 19 et $\texttt{Diablo3}_2$ (cf. tableau 3.11) sélectionne les tuples de `Diablo3` pour lesquels `Saison` = 20. Nous effectuons la fermeture cubique du tuple multidimensionnel (*Féticheur*, *ALL*, *ALL*) dans la relation $\texttt{Diablo3}_1$ en agrégeant tous les tuples qui le spécialisent grâce à l'opérateur +.

$$\begin{aligned}\mathbb{C}((\textit{Féticheur}, \textit{ALL}, \textit{ALL}), \texttt{Diablo3}_1) &= (\emptyset, \ldots, \emptyset)\\ &+ (\textit{Féticheur}, \textit{Vitesse}, \textit{Templier})\\ &+ (\textit{Féticheur}, \textit{Vitesse}, \textit{Enchanteresse})\\ &= (\textit{Féticheur}, \textit{Vitesse}, \textit{ALL})\end{aligned}$$

TABLEAU 3.9. – Relation exemple `Diablo3`

Type	Propriété	Compagnon	Saison	Quantité
Féticheur	*Vitesse*	*Templier*	19	100
Féticheur	*Vitesse*	*Enchanteresse*	19	100
Sorcière	*Chance*	*Enchanteresse*	19	100
Croisé	*Chance*	*Enchanteresse*	19	100
Sorcière	*Vitesse*	*Templier*	20	200
Sorcière	*Chance*	*Enchanteresse*	20	100
Féticheur	*Vitesse*	*Templier*	20	100
Croisé	*Chance*	*Enchanteresse*	20	100
Croisé	*Chance*	*Brigand*	20	300

TABLEAU 3.10. – Relation exemple $\texttt{Diablo3}_1$

Type	Propriété	Compagnon	Quantité
Féticheur	*Vitesse*	*Templier*	100
Féticheur	*Vitesse*	*Enchanteresse*	100
Sorcière	*Chance*	*Enchanteresse*	100
Croisé	*Chance*	*Enchanteresse*	100

TABLEAU 3.11. – Relation exemple $\texttt{Diablo3}_2$

Type	Propriété	Compagnon	Quantité
Sorcière	*Vitesse*	*Templier*	200
Sorcière	*Chance*	*Enchanteresse*	100
Féticheur	*Vitesse*	*Templier*	100
Croisé	*Chance*	*Enchanteresse*	100
Croisé	*Chance*	*Brigand*	300

Voici également d'autres définitions utiles pour définir cette représentation.

Définition 3.9 (Fonction mesure compatible avec la fermeture cubique) - Une fonction mesure, f_{val}, relative à une fonction agrégative f, de $CL(r) \rightarrow \mathbb{R}$ est compatible avec l'opérateur de fermeture $\mathbb{C}$ sur T si et seulement si $\forall t, u \in CL(r)$, elle satisfait les propriétés suivantes :

— $t \preceq_g u \Rightarrow f_{val}(t,T) \geq f_{val}(u,T)$ ou $f_{val}(t,T) \leq f_{val}(u,T)$;
— $\mathbb{C}(t,T) = \mathbb{C}(u,T) \Rightarrow f_{val}(t,T) = f_{val}(u,T)$;
— $t \preceq_g u$ et $f_{val}(t,T) = f_{val}(u,T) \Rightarrow \mathbb{C}(t,T) = \mathbb{C}(u,T)$.

Cette fonction est une adaptation, spécifique au cadre de travail du treillis cube de données, de la fonction poids introduite dans STUMME, TAOUIL, BASTIDE et al. 2002 pour n'importe quel système de fermeture de l'ensemble des parties. Par exemple, les fonctions mesures `COUNT` et `SUM` sont compatibles avec l'opérateur de fermeture cubique.

Donc, en utilisant les fonctions mesures précédentes, nous pouvons en donner une autre définition. L'opérateur de fermeture cubique selon T peut être défini comme suit :

$$\mathbb{C}(t,T) = t \bullet \{t' \in \mathscr{A}\, t(r)\} \text{ tels que } f_{val}(t,T) = f_{val}(t \bullet t', T).$$

Définition 3.10 (Tuple fermé émergent) - Soit $t \in CL(r)$ un tuple, t est un tuple fermé émergent si et seulement si :

— t est un tuple émergent ;
— $\mathbb{C}(t, r_1 \cup r_2) = t$.

Exemple 3.13 - Soit $SeuilMin_1 = 200$ le seuil pour la première contrainte d'émergence (C_1) et $SeuilMin_2 = 200$ le seuil pour la seconde (C_2). Le cube de données émergent de $\texttt{Diablo3}_1$ vers $\texttt{Diablo3}_2$ est donné dans la table 3.12. La figure 3.5 donne la représentation de ce cube de données sous forme de treillis. Sur ce treillis on voit aussi apparaître les bordures classique $[L;U]$. Les valeurs des différentes dimensions sont codées comme suit :

Type		Propriété		Compagnon	
Féticheur	$= F$	*Vitesse*	$= V$	*Enchanteresse*	$= E$
Sorcière	$= S$	*Chance*	$= C$	*Templier*	$= T$
Croisé	$= C$	*ALL*	$= *$	*Brigand*	$= B$
ALL	$= *$			*ALL*	$= *$

TABLEAU 3.12. – Cube de données émergent de $\texttt{Diablo3}_1$ vers $\texttt{Diablo3}_2$

Tuples émergent	*ER*
$(ALL, ALL, Templier)$	3
$(ALL, Vitesse, Templier)$	3
$(Sorcière, ALL, ALL)$	3
$(Croisé, ALL, ALL)$	4
$(Croisé, Chance, ALL)$	4
$(ALL, ALL, Brigand)$	∞
$(ALL, Chance, Brigand)$	∞
$(Sorcière, ALL, Templier)$	∞
$(Sorcière, Vitesse, ALL)$	∞
$(Sorcière, Vitesse, Templier)$	∞
$(Croisé, ALL, Brigand)$	∞
$(Croisé, Chance, Brigand)$	∞

Le tuple $(Sorcière, Vitesse, Templier)$ est un tuple fermé émergent car :

1. $(Sorcière, Vitesse, Templier)$ est un tuple émergent (cf. tableau 3.12).
2. $\mathbb{C}((Sorcière, Vitesse, Templier), \texttt{Diablo3}_1 \cup \texttt{Diablo3}_2)$
   $= (Sorcière, Vitesse, Templier)$.

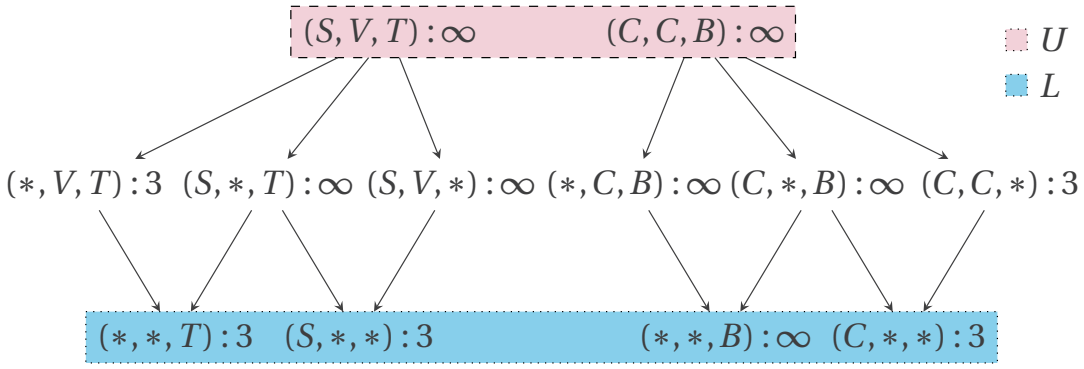


FIGURE 3.5. – Illustration du cube de données émergent avec ses bordures $[L; U]$

L'ensemble des tuples fermés émergents n'est pas une représentation sans perte du cube de données émergent car pour certains tuples il est impossible de décider s'ils

sont émergents ou pas. Ce sont tous les tuples plus généraux que les tuples fermés émergents les plus généraux.

Exemple 3.14 - Considérons l'ensemble de tous les tuples émergents (T) dans le tableau 3.13. Les tuples ($ALL, Chance, Brigand$) et ($ALL, Chance, ALL$) partagent la même fermeture sur T : ($Croisé, Chance, Brigand$), qui est émergent. Le premier tuple est aussi émergent alors que le second ne l'est pas.

TABLEAU 3.13. – Ensemble des tuples fermés émergents

Tuple fermé émergent	ER
($ALL, Vitesse, Templier$)	3
($Sorcière, ALL, ALL$)	3
($Croisé, Chance, ALL$)	4
($Sorcière, Vitesse, Templier$)	∞
($Croisé, Chance, Brigand$)	∞

Afin d'obtenir une représentation sans perte, sont combinés, d'une part, l'ensemble des tuples fermés émergents à partir desquels les valeurs de la mesure peuvent être retrouvées et, d'autre part, les bordures qui délimitent l'espace des solutions. Cependant, la bordure U est déjà incluse dans l'ensemble des tuples fermés car les éléments de U sont les tuples émergents les plus détaillés (spécifiques). Ils sont donc obligatoirement des tuples fermés.

Définition 3.11 (L-cube de données fermé émergent) **-** Le L-cube de données fermé émergent $L-Ecc$ est défini de la manière suivante :

$$L-Ecc(r_1, r_2) = \{t \in CL(r_1 \cup r_2) \text{ tel que } t \text{ est un tuple fermé émergent}\} \cup L$$

Exemple 3.15 - Le L-cube de données fermé émergent est représenté par le tableau 3.13 donnant l'ensemble des tuples fermés émergents et le tableau 3.14 qui propose la bordure L. Il est illustré par la figure 3.6. Les valeurs des différentes dimensions sont codées comme suit :

Type		Propriété		Compagnon	
$Féticheur$	$= F$	$Vitesse$	$= V$	$Enchanteresse$	$= E$
$Sorcière$	$= S$	$Chance$	$= C$	$Templier$	$= T$
$Croisé$	$= C$	ALL	$= *$	$Brigand$	$= B$
ALL	$= *$			ALL	$= *$

Les propositions suivantes permettent de s'assurer que le L-cube de données fermé émergent est une représentation sans perte d'information pour le cube de données émergent.

TABLEAU 3.14. – Bordure $[L;U]$ du cube émergent de $\texttt{Diablo3}_1$ vers $\texttt{Diablo3}_2$

U	$(Sorcière, Vitesse, Templier)$
	$(Croisé, Chance, Brigand)$
L	$(ALL, ALL, Templier)$
	$(Sorcière, ALL, ALL)$
	$(Croisé, ALL, ALL)$
	$(ALL, ALL, Brigand)$

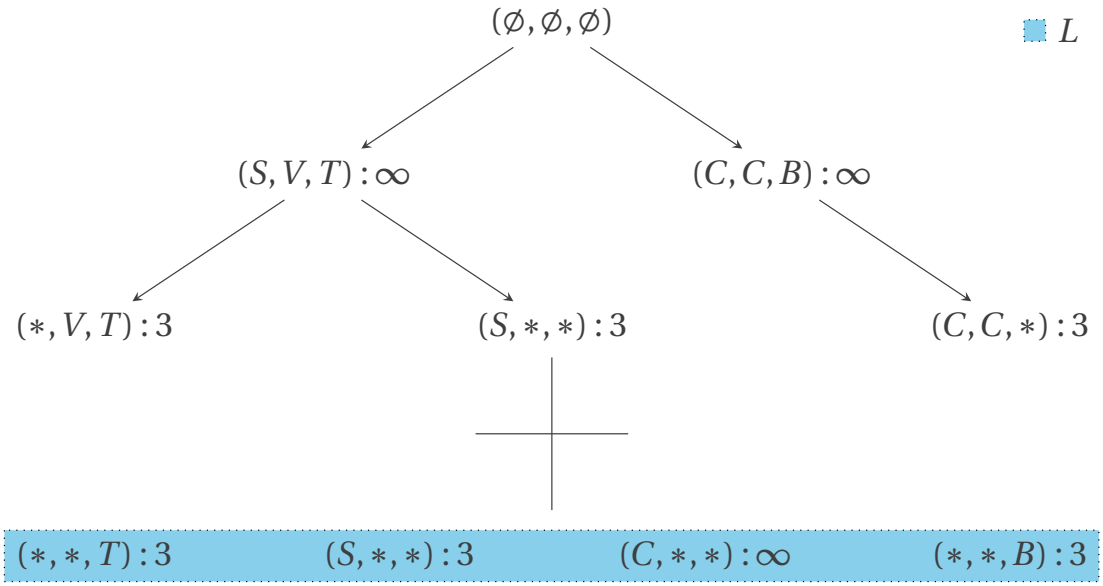


FIGURE 3.6. – Illustration du L-cube de données fermé émergent

Proposition 3.3 - Pour tout tuple fermé émergent t, $\mathbb{C}(t, L-Ecc(r_1, r_2)) = \mathbb{C}(t, r_1 \cup r_2)$

La proposition précédente montre que, pour tout tuple émergent, nous pouvons calculer sa fermeture cubique à partir soit de $r_1 \cup r_2$, soit du L-cube de données fermé émergent et obtenir, dans les deux cas, le même résultat.

Proposition 3.4 - Soit t et t' deux tuples de $EC(r_1, r_2)$, si t et t' ont la même fermeture cubique sur $r_1 \cup r_2$, alors leur taux d'émergence est le même. $\forall\, t, t' \in EC(r_1, r_2)$ tels que $\mathbb{C}(t, r_1 \cup r_2) = \mathbb{C}(t', r_1 \cup r_2)$, nous avons $ER(t) = ER(t')$.

Avec cette deuxième proposition, nous savons que deux tuples ayant la même fermeture cubique ont le même taux d'émergence.

Proposition 3.5 - $\forall\ t \in CL(r_1 \cup r_2)$, t est un tuple émergent $\Rightarrow \mathbb{C}(t, L-Ecc(r_1, r_2))$ est un tuple fermé émergent.

Afin de faire du L-cube de données fermé émergent une représentation sans perte, il est nécessaire de calculer la fermeture cubique sur $r_1 \cup r_2$ car deux tuples peuvent avoir les mêmes fermetures cubiques sur r_1 et sur r_2, mais des fermetures cubiques différentes sur $r_1 \cup r_2$. Cette dernière proposition est importante et nous permet de savoir qu'il est possible avec le L-cube de données fermé émergent de dériver la mesure de tous les tuples émergents.

Le L-cube de données fermé émergent contient les bordures $[L; U]$, nous pouvons donc toujours savoir si un tuple est émergent ou pas. De plus quand un tuple est émergent grâce aux propositions précédentes, nous pouvons dériver sa mesure.

Exemple 3.16 - Dérivons le taux d'émergence du tuple $(ALL, Chance, Brigand)$. Nous savons que ce tuple est émergent parce qu'il appartient à l'intervalle $[L; U]$ (cf. figure 3.5). En calculant sa fermeture cubique sur $L-Ecc(\texttt{Diablo3}_1, \texttt{Diablo3}_2)$, nous obtenons le tuple $(Croisé, Chance, Brigand)$. Puisque le taux d'émergence du tuple précédent est ∞, nous savons que le taux d'émergence de $(ALL, Chance, Brigand)$ est ∞ et donc nous retrouvons le résultat présenté dans la table 3.12.

3.5.2. L'algorithme de cube de données fermé émergent : $\mathbb{C}$-IDEA

La représentation L-cube de données fermé émergent utilise la bordure L et les tuples fermés émergents, de ce fait F-IDEA est utilise comme base pour la construction d'une nouvelle famille d'algorithmes dédiés au calcul de cette représentation : $\mathbb{C}$-IDEA (pour *close integrable database algorithm*). Cet algorithme utilise certaines propriétés de F-IDEA pour calculer efficacement l'ensemble des tuples fermés émergents. Préalablement à la présentation de l'algorithme $\mathbb{C}$-IDEA, nous donnons les propositions sur lesquelles il se base.

Proposition 3.6 (Tuple émergent non-fermé) **-** Soit P un ensemble de tuples de r en accord sur l'ensemble de dimensions $D \subseteq \mathscr{D}$ généré lors d'un partitionnement de l'algorithme F-IDEA et P' l'un de ses IDEA-Successeurs maximaux par rapport à D (P'

est un ensemble en accord selon $D \cup d_k$). Si $|P| = |P'|$ alors le tuple t associé à P est un tuple non-fermé.

Cette proposition est la première intégrée au sein de F-IDEA pour éliminer la majorité des tuples non-émergents. Malheureusement, cette proposition ne garantit pas, à elle-seule, l'élimination de tous les tuples non fermés. Pour ne pas générer de solutions erronées, un second test permettant de garantir la présence d'un vrai tuple fermé est nécessaire.

Propriété 3.3 (Tuple fermé candidat) **-** Un tuple non éliminé grâce à la proposition des tuples émergents non-fermés (3.6) sont appelés des tuples fermés candidats.

Proposition 3.7 (Fermeture d'une partition IDEA) **-** Soit P l'une des partitions non vides générées lors d'un partitionnement de l'algorithme F-IDEA. La fermeture cubique du tuple multidimensionnel t associé P peut être obtenue comme suit :

$$\mathbb{C}(t, r_1 \cup r_2) = \underset{t' \in P}{+} t'$$

Cette proposition est particulièrement pertinente car le coût de calcul d'une fermeture est ainsi largement minimisée. Ce calcul s'effectuant sur une partition minimale de la relation d'origine, on évite ainsi de devoir balayer toute la relation inutilement. En plus grâce à cette proposition, le nombre de calculs est lui aussi réduit.

Proposition 3.8 (Test de fermeture) **-** Soit t un tuple fermé candidat et P la partition de r correspondant. t est un fermé si et seulement si $+_{t \in P}\, t = t$.

L'algorithme $\mathbb{C}$-IDEA (cf. algorithme 10) est une déclinaison de F-IDEA utilisant les propositions ci-dessus durant la phase P_2 pour calculer efficacement les tuples fermés émergents et ainsi obtenir les représentations réduites du cube de données émergent. Cette approche tire profit du mode de fonctionnement de l'algorithme par partitionnement successif, pour optimiser autant que possible le calcul des cubes de données fermés émergents. En outre, comme tous les autres algorithmes de la famille IDEA, cette approche reste directement intégrable au sein d'un SGBD.

3.6. Conclusion

Dans ce chapitre, nous avons présenté le cube de données émergent afin de capturer les renversements de tendances entre deux cubes de données de deux relations comparables. Calculer un cube de données émergent est coûteux et les résultats peuvent s'avérer très volumineux. Aussi nous nous sommes intéressés aux représentations réduites de tels cubes de données. Les cubes de données émergents peuvent être représentés par les bordures $[L; U]$, ce qui a pour avantage : d'éviter de calculer les deux cubes de données sous-jacents à comparer, de permettre de savoir si un renversement de tendance est effectif, d'offrir un outil de classification de tendances et de focaliser l'attention de l'utilisateur sur un ensemble particulier de tuples remarquables. Cet

Algorithme 10 Algorithme ℂ-IDEA

Entrée :
la relation fusionnée r.
L'ensemble des dimensions $\mathscr{D} = \{d_1, \ldots, d_n\}$ à traiter.
La cellule courante $CelluleCour$.
La phase courante de l'algorithme P_{Cour}.
Sortie :
L'ensemble des tuples fermés émergents de r.
La phase sortie de l'algorithme.

```
01: DimCour := d_1;
02: Agréger(r, CelluleCour, f); //Calcule les valeurs de la
    fonction mesure et les écrit dans CelluleCour
03: soit P_Suiv := PhaseSuiv(P_Cour, CelluleCour); //cf. figure 3.4
04: si (P_Suiv = P_3) alors
05:   retourner P_Suiv;
06: sinon si (P_Suiv = P_2) alors
07:   si |r| = 1 alors
08:     AjouterFermé(r[0]);
09:     retourner P_3;
10:   fin si
11: fin si
12: pour tout d_j ∈ 𝒟 faire
13:   C := |r(d_j)|;
14:   Partitionner(r, d_j); //r est partitionnée suivant d_j en
      C partitions : r_1, ..., r_C
15:   pour i = 1, ..., C faire
16:     si (P_Suiv = P_2) ∧ (|r_i| ≠ |r|) alors
17:       t := Somme(r); //Calcul de la fermeture
18:       si t = CelluleCour alors
19:         AjouterFermé(CelluleCour); //cf. proposition 3.8
20:       fin si
21:     fin si
22:     //CelluleCour reçoit la valeur de d_j pour r_i
23:     CelluleCour.d_j := r_i[0].d_j;
24:     ℂ-IDEA(r_i, {d_{j+1}, ..., d_n}, CelluleCour, P_Suiv);
25:   fin pour
26:   CelluleCour.d_j := ALL; //d_j est entièrement traitée
27: fin pour
28: retourner P_Suiv;
```

ensemble peut être le point de départ pour l'exploration des résultats. Des approches permettant l'obtention du cube de données émergent et de ses bordures ont aussi été présentées. Elles englobent des solutions bases de données, via des requêtes SQL, et des solutions algorithmiques efficaces, E-IDEA et F-IDEA. Parmi les requêtes SQL proposées, la plus simple et la plus efficace s'appuie sur une relation qui fusionne les deux relations considérées pour le calcul du cube de données émergent. Les solutions algorithmiques présentées, exploitent, quant à elles, une approche existante, performante et intégrable au cœur d'un SGBD : BUC. E-IDEA permet le calcul de cubes de données émergents, et F-IDEA permet en plus le calcul de ses bordures [L;U].

Complétant la représentation concise par bordures mais ne permettant pas de dériver les valeurs des mesures, le L-cube de données fermé émergent est une représentation plus réduite ne générant pas de perte d'information. En revanche, cette représentation est plus volumineuse, mais en contrepartie elle permet de répondre à toute requête sur le cube de données émergent tout en enregistrant le minimum d'information. Donc, si l'utilisation du cube de données émergent est la formulation de requêtes alors le L-cube de données fermé émergent est la représentation pour optimiser l'espace de stockage.

4. Conception de jeux vidéo

Sommaire

4.1. Introduction . . . 121
4.2. Jouet, jeu et jeu vidéo . . . 122
4.3. Approche formelle de la conception de jeux vidéo . . . 123
4.3.1. Mécaniques . . . 124
4.3.2. Dynamiques . . . 124
4.3.3. Esthétiques . . . 124
4.4. Classification . . . 124
4.4.1. Classification de modèles de joueurs . . . 124
4.4.1.1. Bartle . . . 125
4.4.1.2. Yee . . . 126
4.4.2. Classification de jeux vidéo . . . 126
4.4.2.1. Catégories de Caillois . . . 127
4.4.2.2. Typologie de la théorie des jeux . . . 127
4.4.2.3. Genre de jeux vidéo . . . 127
4.4.2.4. Approche atomique . . . 128
4.5. Gameplay . . . 130
4.5.1. Méthodologie de conception de jeu vidéo . . . 130
4.5.2. Fun . . . 130
4.5.3. Flow . . . 132
4.5.4. Défi, incertitude, difficulté et virtuosité . . . 133
4.5.4.1. Défi . . . 133
4.5.4.2. Incertitude . . . 133
4.5.4.3. Difficulté . . . 135
4.5.4.4. Virtuosité . . . 135
4.5.5. Boucle de jeu . . . 135
4.5.5.1. Interaction jeu-joueur . . . 135
4.5.5.2. Boucle OCR . . . 136
4.5.6. Motivations . . . 136
4.6. Équilibrage . . . 137
4.6.1. Domaines d'équilibrage . . . 138
4.6.2. La théorie des jeux . . . 139
4.6.2.1. Forme normale d'un jeu . . . 139
4.6.2.2. Meilleure réponse . . . 139
4.6.2.3. Équilibre de Nash . . . 139

4.6.2.4. Dominance stratégique 140
4.7. Conclusion 140

4.1. Introduction

Il n'existe pas, à notre connaissance, de théorie unifiée pour la conception de jeux vidéo (ou *game design*), et pas de formule simple montrant comment réaliser de « bons » jeux vidéo. L'équivalent de la classification périodique des éléments pour la Chimie n'a pas encore été établi pour les jeux vidéo (SCHELL 2014).

Le concepteur de jeu vidéo (ou *game designer*) a pour rôle de décider ce que sera l'expérience de jeu [1] et de la mettre en œuvre (FULLERTON 2008) de manière plaisante et motivante. La phase de conception d'un jeu vidéo est une suite d'itérations, d'étapes répétées où chaque changement est implémenté, puis éprouvé afin d'en juger la qualité, l'équilibre et la nécessité. Ces modifications sont testées d'abord par les équipes de développements, mais de plus en plus, il est fait appel à des éléments externes pour diversifier les points de vue, les habitudes de jeu et les profils de joueurs. Depuis la démocratisation des moyens de communication informatiques à haut débit, ce sont le plus souvent des joueurs du monde entier qui sont sélectionnés pour faire des sessions de jeu (*playtest*) ou bien pour des durées plus longues sur des versions plus abouties (*bêta-test*). Ces tests sont généralement suivis avec la plus grande attention par les équipes de développement, lors de séances d'observation, et le prototype du jeu vidéo peut évoluer rapidement pour recevoir de nouvelles modifications qui seront ou non conservées. Cela permet d'altérer des éléments majeurs du jeu vidéo, mais aussi de modifier quelques subtilités ayant trait à l'équilibre entre ces éléments. Ce processus itératif a généralement lieu dans des laboratoires de tests au cœur même des acteurs majeurs de l'industrie vidéo ludique. Mettre en œuvre ces tests est généralement coûteux et les contraintes associées les rendent difficile à appliquer. Trouver des profils de testeurs représentatifs, faire durer les tests suffisamment longtemps pour disposer de suffisamment de données, disposer de suffisamment de testeurs et de matériels . . . Les laboratoires ont alors besoin de moyens démesurés pour pouvoir se faire une idée de ce que sera le produit dans les mains d'utilisateurs divers. Avec la démocratisation d'Internet, le taux de consoles et d'ordinateurs connectés n'a cessé de croître. De même, les jeux vidéo mobiles (sur téléphone intelligent ou *smartphone*) nécessitent souvent une connexion Internet, ne serait-ce que pour les télécharger. Si bien qu'il est ainsi devenu particulièrement aisé de récolter des informations sur l'ac-

1. L'expérience de jeu désigne l'ensemble des sensations, émotions et interactions vécues par un joueur lorsqu'il joue à un jeu vidéo ou à tout autre type de jeu. C'est la façon dont le joueur perçoit, interagit et apprécie le jeu. L'expérience de jeu est influencée par de nombreux facteurs tels que le *gameplay*, la narration, les graphismes, la musique, l'ambiance sonore, la difficulté, l'immersion, le sentiment de progression, la liberté d'action, les interactions sociales dans un contexte multijoueur, etc. L'expérience de jeu englobe toutes les impressions et les sensations que le joueur ressent tout au long de son sa pratique ludique, allant de la satisfaction de maîtriser de nouvelles compétences à l'excitation d'explorer un monde virtuel, en passant par l'immersion dans une histoire captivante ou la connexion sociale avec d'autres joueurs.

tivité des joueurs. C'est comme cela qu'est née l'idée de faire tester les jeux vidéo par les joueurs directement. Analyser les comportements de joueurs réels sur une version non finale mais suffisamment proche du produit fini permet de détecter rapidement tous les comportements et corriger tout défaut. Pour certains jeux vidéo modernes pour lesquels il est prévu une durée de « jouabilité » extrême (en milliers d'heures par joueur) il est impossible à un panel restreint de couvrir l'ensemble des comportements potentiels. À l'heure actuelle, il n'existe pas de modèle capable de prendre en compte la complexité de ces données dépendantes du temps et de traiter un si grand volume d'information, tout en étant facilement interprétable par les concepteurs de jeux vidéo.

Dans ce chapitre, nous introduisons des théories de conception de jeux, que nous appliquons au jeux vidéo. Nous pouvons les trouver plus en détail dans SALEN et ZIMMERMAN 2004 ; SCHELL 2014 ; ALBINET 2015.

4.2. Jouet, jeu et jeu vidéo

De manière basique, un jouet est un accessoire de jeu, et un jeu a le plus souvent des règles, des objectifs et il est interactif. Sa version informatique, dotée d'une interface utilisateur sur un dispositif vidéo permettant une interaction ludique avec le joueur est appelée jeu vidéo. La particularité d'un jeu vidéo vient aussi du fait que sa boucle de jeu (cf. sous-section 4.5.5) est orchestrée par un système logique au lieu de personnes.

Les jeux ne sont donc pas toujours réglés et peuvent être approchés d'un point de vue du développement psychomoteur (PIAGET 1945) :

— sensori-moteurs : phase de développement où le joueur s'attache surtout à observer, retrouver et manipuler des objets ;
— symboliques : phase où le joueur rattache aux objets une dimension symbolique (comme utiliser un bâton comme une épée) ;
— règles : phase où le système de règle fait son apparition.

Le jeu est une notion difficile à circonscrire. Les tentatives visant à le définir sont nombreuses, et le langage y fait parfois obstacle. Nous pouvons organiser les tentatives de définitions en deux groupes (ROY 2019) :

— les théories qui appréhendent le jeu comme une activité (CHÂTEAU 1950), une expérience vécue (CAILLOIS 1958) ou une attitude (CHÂTEAU 1989) ; autrement dit, une activité humaine, voire même un indicateur du propre de l'humain ;
— les théories qui approchent le jeu comme un objet matériel, un dispositif technologique ou un ensemble de règles (JUUL 2011) ; autrement dit, une entité technique.

Pourtant, plutôt que d'opposer l'humain et la technique, la théorie de la technique anthropologiquement constitutive et constituante (ou TAC) de GOODY et BENSA 1979 décrit la technique, non seulement comme une production de l'humain, mais comme son corrélat.

Dans certaines langues, comme en anglais et en portugais, il y a deux mots bien distincts pour désigner le jeu : respectivement *play* et *game*, et *brincar* et *jogar* (ALVAREZ

2007). Dans chaque cas, cela décrit deux conceptions différentes du jeu (BROUGÈRE 2005). Ces conceptions ne sont pas sans rappeler les modalités de jeu *paidia* et *ludus* de CAILLOIS 1958, grandement inspirées de HUIZINGA 1951 :

— *paidia* : puissance primaire d'improvisation et d'allégresse ; la *paidia* est, en quelque sorte, la recherche d'un moment ludique où le jeu est improvisé, et désigne ce qui appartient à l'enfant mais ne se limite pas pour autant au jeu des enfants. Ce terme ainsi que ses dérivés, désigne le jeu en général jusqu'à celui le plus noble, et le plus saint (KELLNER 2000) ;
— *ludus* : goût de la difficulté gratuite ; le *ludus* désigne un défi à relever, des objectifs et un but pour un joueur.

Une des définitions les plus précises du jeu, celle de SALEN et ZIMMERMAN 2004, liste les approches suivantes :

1. Principes fondés sur des règles qui limitent les joueurs ;
2. Conflit ou challenge ;
3. Objectif orienté / résultat orienté ;
4. Activité, processus, ou évènement ;
5. Implique la prise de décision ;
6. Pas sérieux et prenant ;
7. Jamais associé à des gains matériels ;
8. Artificiel / sûr / hors de la vie ordinaire ;
9. Crée des groupes sociaux spéciaux ;
10. Volontaire ;
11. Incertain ;
12. Fait croire / représentatif ;
13. Inefficace ;
14. Système de répartition / ressources et prises ;
15. Une forme d'Art.

Même si elles restent tout à fait subjectives, la plupart des définitions s'accordent sur des combinaison de sous-ensembles des ces approches :

— pour CAILLOIS 1958, c'est 1, 7, 8, 10, 11 et 12 ;
— pour ABT 1970, c'est 1, 3, 4 et 5 ;
— pour CRAWFORD 1984, c'est 1, 2, 5, 8, 12 et 14 ;
— pour BROUGÈRE 2005, c'est 1, 4, 5, 8 et 11.

4.3. Approche formelle de la conception de jeux vidéo

Le travail d'un concepteur de jeu vidéo n'est pas de directement créer une expérience pour le joueur mais bien de lui fournir les mécaniques qui lui permettront de façonner son expérience personnelle avec le jeu vidéo. Même dans des univers très

contraints, des joueurs trouveront toujours des façons atypiques d'utiliser le jeu vidéo (ou *gameplay* émergent) et de s'écarter des cas d'utilisation attendues par les équipes de conception (SALEN et ZIMMERMAN 2004).

Nous pouvons envisager selon la structure MDA (de *mechanics, dynamics and aesthetics* pour mécaniques, dynamiques et esthétiques) la décomposition d'un jeu vidéo en trois parties (HUNICKE, LEBLANC et ZUBEK 2004). Chaque composant de MDA peut être considéré comme une « vision » du jeu vidéo, séparée, mais liée de manière causale (LEBLANC 2004).

4.3.1. Mécaniques

Les mécaniques sont les éléments basiques d'un jeu vidéo, les représentations des données, les algorithmes gérant notamment la physique du jeu vidéo ou les comportements des différents composants ludiques. C'est la partie logique du jeu vidéo d'où peut être extraite la majorité des données d'analyse de l'expérience utilisateur.

4.3.2. Dynamiques

Les dynamiques émergent des mécaniques. Ce sont les interactions entre les actions des joueurs et les mécaniques ainsi que leurs résultantes au cours de l'expérience de jeu. Ces interactions font émerger des comportements qui mettent parfois en lumière des déséquilibres, jusqu'alors non discernables, qui appelleront des corrections de la part des développeurs.

4.3.3. Esthétiques

L'esthétique du jeu vidéo décrit les émotions ressenties par le joueur à travers l'expérience du jeu, telles que le *fun* (cf. section 4.5.2), le sentiment de liberté, la tristesse ou la peur. La difficulté pour les concepteurs consiste à générer des émotions en agissant uniquement sur les mécaniques du jeu vidéo. Un des leviers les plus exploités par les *game designers* consiste à construire la difficulté du jeu vidéo de manière à ce qu'elle permette au joueur d'entretenir un sentiment de progression, de réussite et de maîtrise.

De nombreux auteurs proposent des listes composées de toutes sortes d'expériences esthétiques, comme celles de BERGSTRÖM et BJÖRK 2009, qui témoignent de la profusion des expériences de jeu.

4.4. Classification

4.4.1. Classification de modèles de joueurs

Une compréhension poussée des joueurs d'un jeu vidéo est essentiel afin de pouvoir lui offrir la meilleure expérience de jeu possible, des défis de difficulté idéales par

rapport à ses capacités et ses attentes. Pour cela, avoir un cadre, créer des modèles comportementaux, motivationnels et émotionnels de joueurs est primordial.

Nombreux sont ceux qui se sont prêtés à l'exercice de la classification de modèles de joueurs (NACKE, BATEMAN et MANDRYK 2013 ; SZWOCH 2015), nous n'en présenterons que deux, représentatifs, ici.

4.4.1.1. Bartle

Il est rapidement apparu nécessaire de classer les joueurs pour limiter le nombre de profils à étudier pour développer un jeu vidéo. La première topologie proposée, appelée MUD (pour *multi-user dungeon*), se base sur quatre profils différents selon les objectifs principaux des joueurs (BARTLE 1996). Les deux premiers sont les profils asociaux, dont les objectifs sont purement liés au jeu vidéo et non aux autres joueurs. Les deux suivants sont les profils sociaux, dont les objectifs sont principalement liés aux autres joueurs et les interactions avec eux (cf. figure 4.1) de la manière suivante :

— si l'objectif est d'explorer au plus l'univers du jeu vidéo, trouver les éléments cachés, alors on classera le joueur comme explorateur ;
— si l'objectif est de compléter au plus possible toutes les tâches et quêtes du jeu vidéo, totaliser un maximum de points, alors on classera le joueur comme *power gamer* ou *achiever* ;
— si l'objectif est de collaborer, discuter, échanger pour progresser dans le jeu vidéo et faciliter le jeu des autres, alors on classera le joueur comme socialisateur ;
— si l'objectif est de surpasser les autres, en obtenant de meilleurs scores ou bien en les éliminant d'une compétition liée à un jeu vidéo, alors on classera le joueur comme *killer*.

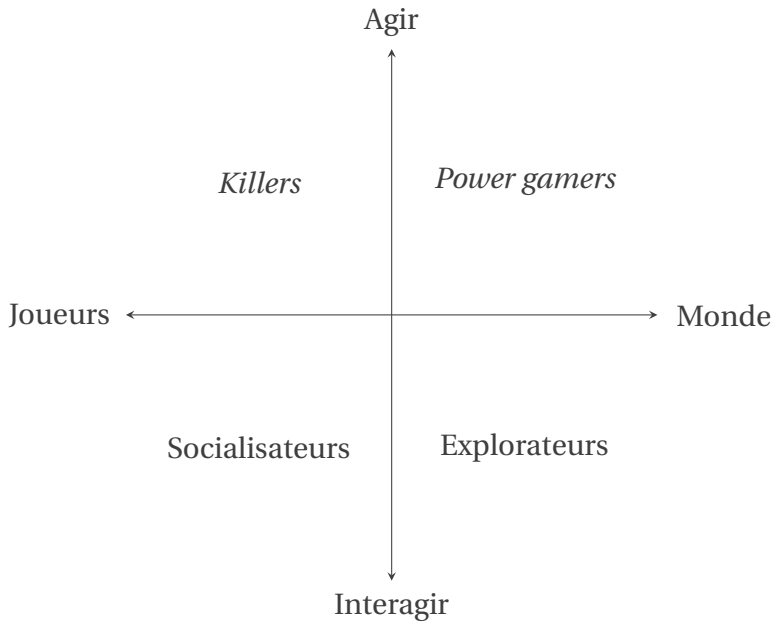


FIGURE 4.1. – Classification des joueurs de Bartle

4.4.1.2. Yee

Après une étude importante sur un grand nombre de joueurs de MMORPG[2], un nouveau format de classification a été établi, qui repose sur le fait d'attribuer, pour chaque joueur un score dans trois composantes découpées en sous composantes (YEE 2006) :

- accomplissement :
  - avancement : le désir de pouvoir, de progrès rapide, d'accumuler des richesses ou du statut ;
  - mécaniques : le besoin d'analyse des règles et du système pour optimiser le personnage ;
  - compétition : l'envie d'affronter et de se mesurer à d'autres joueurs ;
- social :
  - socialisation : l'intérêt d'aider et de discuter avec les autres joueurs ;
  - relations : le désir de former des relations durable et pleines de sens ;
  - facteur humain : le fait de prendre du plaisir à faire progresser le groupe ;
- immersion :
  - découverte : l'envie de découvrir le monde, trouver des éléments secrets ;
  - jeu de rôle : l'intérêt de créer une personnalité avec un historique cohérent et interagir avec d'autre joueur pour créer une histoire vivante ;
  - personnalisation : le désir de personnalisation de l'apparence du personnage ;
  - évasion : le besoin de fuir le monde réel.

TABLEAU 4.1. – Classification de joueurs de Yee

Accomplissement	Social	Immersion
Avancement	Socialiser	Découverte
Mécaniques	Relations	Jeu de rôle
Compétition	Facteur humain	Personnalisation
		Évasion

4.4.2. Classification de jeux vidéo

En dehors de toutes autres considérations, le nombre de joueurs et la nature de leur relation au sein d'un jeu vidéo peut constituer un critère de classification d'un jeu vidéo (NATKIN et VEGA 2004 ; FULLERTON 2008).

Nous présentons dans cette sous-section les propositions taxonomiques que nous avons retenu dans notre travail de recherche.

2. Le jeu de rôle en ligne massivement multijoueur (JDRMM ou MMORPG pour *massively multiplayer online role-playing game*) est un type de jeu vidéo hybride associant le jeu de rôle et le jeu en ligne massivement multijoueur, permettant, simultanément, la réunion d'un grand nombre de joueurs dans un monde virtuel.

4.4.2.1. Catégories de Caillois

La catégorisation atypique proposée par CAILLOIS 1958 n'est pas faite de manière formelle, et se base peu sur d'autres classifications, mais utilise les modalités du jeu *paidia* et *ludus* :

- *agôn* : jeux de compétition qui fait la part belle à l'égalité des chances, au besoin, et lorsque cela est possible, avec des parties à handicap en cas de fortes différences de niveaux ;
- *alea* : jeux de pur hasard ne faisant pas appel à une quelconque virtuosité ; à l'opposer de *l'agôn* qui demande aux joueurs d'user de leurs compétences ;
- *mimicry* : jeux de simulacre, qui font appel à l'incarnation de personnages et des jeux de rôles ;
- *illinx* : jeux vertigineux, qui donne des sensations d'ivresse et mettent à l'épreuve les sens des joueurs.

4.4.2.2. Typologie de la théorie des jeux

La théorie des jeux (cf. sous-section 4.6.2) offre une classification des jeux en catégories en fonction de leurs approches de résolution :

- jeux à information complète et information incomplète : une information est dite complète si la prise de décision se fait en connaissance :
  - des possibilités d'action ;
  - des possibilités d'action des autres joueurs ;
  - des gains résultants pour chacune de ces actions ;
  - les motivations des autres joueurs ;
- jeux à somme nulle et jeux à somme non nulle : les gains d'un joueur correspondent exactement aux pertes d'un autre ;
- jeux déterminés : cas particulier de jeu à somme nulle appeler jeux de Nim (de l'allemand *Nimm* « prends ! ») ou jeux de stratégies pure ;
- jeux coopératifs et jeux non coopératifs : besoin ou non de coalition ;
- jeux à mémoire parfaite et mémoire imparfaite : tous les anciens coups sont connus ou non ;
- jeux simultanés : stratégies décidées en même temps ;
- jeux séquentiels : stratégies basées sur l'ordre des décisions ;
- jeux finis : ensemble fini de stratégies ;
- jeux répétés : répétition d'un jeu, avec connaissance des résultats intermédiaires, et phénomènes de réputation.

4.4.2.3. Genre de jeux vidéo

Les genres de jeux vidéo sont déterminés par le *gameplay* : quels défis le joueur doit relever et quelles actions il entreprend pour dépasser ces défis. Les quatre genres principaux sont (DIBERDER A LE DIBERDER FRÉDÉRIC. L'UNIVERS DES et VIDÉO 1998 ; ROLLINGS et ADAMS 2003 ; NATKIN et VEGA 2004 ; ROGERS 2014) :

- action : commandes répétées frénétiquement ;

— aventure : histoire au cœur du jeu vidéo ;
— *puzzle* : pensée analytique ;
— stratégie : choix non-triviaux ;

Les genres principaux sont découpés en sous-genres, et certains jeux vidéo sont multi-genres.

Bien que les propositions de classification par genre peuvent être considéré comme références, certains considèrent que ces types de classifications sont, par nature, subjectives et condamnées à la désuétude du fait de la multitude de nouvelles perspectives offertes en permanence par les évolutions, notamment celles technologiques (GENVO et LETOURNEUX 2011).

4.4.2.4. Approche atomique

Dans une approche atomique du jeu vidéo permettant d'établir des classifications à différents niveaux, des atomes eux-même à des objets plus complexes composés d'atomes. Nous désignons généralement chaque atome comme étant « un système de jeu », mais cette définition est jusqu'à un certain point récursif. Une fois que vous avez assemblé plusieurs atomes dans une molécule, la molécule dans son ensemble doit également répondre à tous les critères de ce qui rend un atome amusant (KOSTER 2012).

Briques gameplay Contrairement aux autres jeux, les règles d'un jeu vidéo sont rarement explicites. Il est alors fréquent de les retrouver précisément en cours de partie par des successions de tests et d'échecs. Fort de ce constat, un principe de classification qui se base sur des « schémas de règle de jeu » a été élaboré. Ces schémas étant formalisés dans un élément appelé brique *gameplay*. Sur ce modèle, actuellement douze briques *gameplay* (cf. figure 4.2) ont été recensées (ALVAREZ 2018) :

1. `Atteindre` (`At`) : atteindre une valeur ou une position donnée ;
2. `Éviter` (`Ev`) : donne une pénalité en cas d'un élément interdit touché ;
3. `Créer` (`Cr`) : admet la génération d'éléments ;
4. `Détruire` (`Dt`) : admet la suppression d'éléments ;
5. `Gérer` (`Gr`) : conditionne la gestion d'une ou plusieurs ressources limitées ;
6. `Choisir` (`Ch`) : offre un choix ;
7. `Déplacer` (`Dp`) : permet de déplacer un élément ;
8. `Écrire` (`Ec`) : supporte une commande d'écriture ;
9. `Randomiser` (`Rn`) : génère une ou plusieurs valeurs aléatoires ;
10. `Shooter` (`Sh`) : tir sur une cible à déplacement non nul ;
11. `Vocaliser` (`Vc`) : produit une onde sonore, un chant, un sifflement, . . . ;
12. `Transformer` (`Tr`) : modifie une ou plusieurs propriétés de représentation, d'interaction ou de contenu.

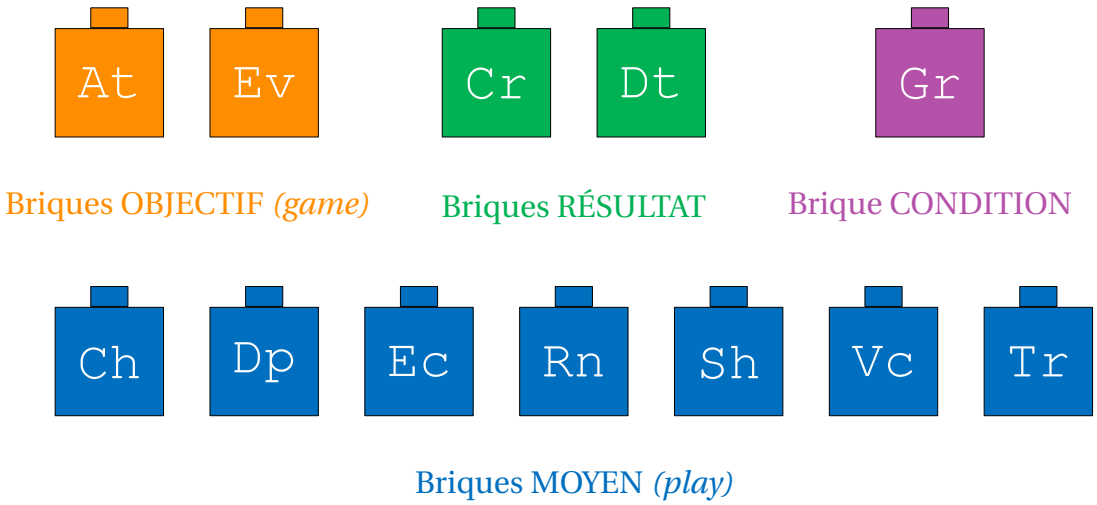


FIGURE 4.2. – Briques *gameplay*

Métabriques Les briques se combinent souvent par paires au sein des jeux vidéo. Une telle association est appelée métabrique. Elle est essentielle car elle permet une véritable classification par les briques *gameplay* en limitant de manière importante le nombre de familles de jeux (combinaisons d'éléments). Il a été établi quatre règles pour définir ce concept de métabrique (ALVAREZ 2007) :

1. sont appelées métabriques la combinaison de deux briques *gameplay* de nature complémentaires qui donne ainsi naissance à un défi ;
2. ajouter une brique *gameplay* à une métabrique, confère au défi porté par cette dernière, une variante, qui cependant n'affecte pas sa nature profonde ;
3. si l'on ajoute plusieurs briques *gameplay* à une métabrique, le point 2 reste probablement vrai tant que la mise en présence de ces briques *gameplay* ne constitue pas à leur tour une autre métabrique ;
4. Associer des métabriques revient à associer leur défi respectif.

Le tableau 4.2 donne la combinaison des métabriques répertoriées à ce jour.

TABLEAU 4.2. – Liste des métabriques

Briques *gameplay*	Atteindre	Éviter
Choisir	*Puzzle*	*Quiz*
Déplacer	*Solitary*	*Driver*
Écrire	*Adventure*	*Hangman*
Randomiser	*Jackpot*	*Roulette*
Shooter	*Killer*	*Mine*
Vocaliser	*Karaoké*	*Wave*
Transformer	*Safe*	*Bomb*

4.5. Gameplay

Le *gameplay* est une notion difficile à appréhender et à traduire en français, réunissant tout à la foi les concepts de « jouabilité », de maniabilité mais aussi de difficulté, aussi nous le préservons sous ce vocable dans ce mémoire.

Du point de vue des règles du jeu vidéo, le *gameplay*, est composé par des règles définissant des objectifs à accomplir associées à d'autres règles spécifiant des moyens et des contraintes pour atteindre ces objectifs (ALVAREZ 2007).

Le *gameplay* peut être défini comme toutes les actions (cognitives ou physiques, performantes et stratégies) effectuées par le joueur influençant positivement ou négativement le résultat de la situation de jeu incertaine dans laquelle il se trouve (GUARDIOLA et NATKIN 2005).

Les règles sont une composante vraiment essentielle d'un jeu vidéo, elles sont le résultat de la conception du jeu vidéo et visent à maitriser les objectifs, les obstacles et les moyens de résolutions mise à la disposition du joueur (Jesper JUUL 2003 ; HUNICKE, LEBLANC et ZUBEK 2004).

4.5.1. Méthodologie de conception de jeu vidéo

La plupart des méthodes de conception de jeux vidéo n'ont jamais été évaluées scientifiquement (NEIL 2012). Et bien qu'il n'y ait pas de consensus sur les méthodes ou les étapes de conception de jeux vidéo, nombre d'auteurs s'accordent sur un processus itératif en trois étapes : prototype, *playtest*, révision (SALEN et ZIMMERMAN 2004 ; ROLLINGS et ADAMS 2003 ; FULLERTON 2008). Le *game design* est un processus clairement anthropocentré : le concepteur de jeu vidéo défini la cible et l'expérience de jeu souhaitée, puis s'attache ensuite à faire en sorte que le jeu délivre cette expérience de jeu. Le *game designer* intervient, de manière partiellement prédictible, durant toute la durée de création du jeu vidéo (FULLERTON 2008 ; ALBINET 2015) :

— phase de conception : élaboration du *gameplay* ;
— phase de pré-production : création du prototype final par sa fondation suivi de sa structure, ses détails formels et son affinage ;
— phase de production : livraison de plusieurs versions alpha et bêta ;
— phase de post-production, de qualité : obtention d'un master, d'un « gold master »[3].

4.5.2. Fun

Dans le domaine des jeux vidéo, le *fun* peut être employé pour décrire le plaisir éprouvé suite à l'apprentissage de nouveaux motifs et à la résolution mentale d'un défi.

3. Lorsqu'un un jeu vidéo offre toutes les fonctionnalités prévues et que le nombre de bogues est jugé minimal, on considère qu'il a atteint sa version « finale », « stable » ou « or », qui peut alors être publiée.

Exemple 4.1 - Le jeu du tic-tac-toe (aussi appelé « morpion » ou « oxo », cf. figure 4.3) est un dispositif ludique de stratégie pure apprécié tant que ses joueurs n'ont pas la connaissance des stratégies dominantes (suite de coups forçant la victoire ou le nul). Dès lors, il est possible de prévoir la victoire, et le jeu perd de son intérêt car il n'offre plus rien à apprendre. Il n'est donc plus « *fun* » (KOSTER 2014).

FIGURE 4.3. – Glow Tic Tac Toe, exemple de jeu vidéo de tic-tac-toe [a]

a. Capture d'écran par Arclite Systems Ltd sur Google Play (https://play.google.com).

L'organisme libère naturellement de la dopamine, molécule associée au plaisir, lorsque nous apprenons (KOSTER 2014). Cette mécanique physiologique est un des éléments décisifs dans l'évolution humaine. En effet, apprendre une nouvelle connaissance, compétence ou capacité à permis aux individus de s'adapter à des situations nouvelles ou à des changements environnementaux. C'est cette même dopamine qui est libérée après avoir surpassé un objectif qui semblait complexe, cela aussi est un héritage de notre évolution. La recherche de dopamine ayant motivé l'évolution jusqu'à nos jours, il n'est pas étonnant que de nombreux utilisateurs fassent maintenant des jeux vidéos et de leurs nouvelles mécaniques à apprendre et défis toujours plus

créatifs leur domaine d'expérimentation.

4.5.3. Flow

Nous pouvons considérer que l'équilibre entre la difficulté d'une tâche dans un jeu vidéo et la virtuosité du joueur à l'accomplir est l'apogée de l'expérience ludique. C'est le concept de *flow* (CSIKSZENTMIHALYI 2009), assemblage subtil et complexe de difficulté, d'apprentissage, de virtuosité, d'accessibilité et de fluidité dans la réussite. C'est une expérience autotélique [4] consciente et positive qui exprime le sentiment de fluidité, de continuité et de concentration, décrite par des personnes interrogées lorsqu'elles exerçaient leur activité préférée, fusse-t-elle ludique ou une tâche de leur vie quotidienne. Une difficulté trop faible, pour un joueur donné, conduira à une expérience poussive et ennuyeuse, quand à l'inverse, une difficulté trop élevée amènera chez certains une motivation supplémentaire, mais le plus souvent une grande frustration (cf. figure 4.4).

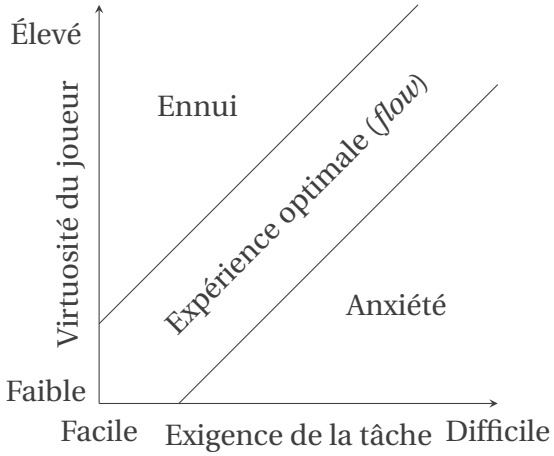


FIGURE 4.4. – La dynamique du *flow*

Le *flow* se caractérise par : un but clair, une rétroaction immédiate, l'intégration de la conscience et de l'action, la concentration sur la tâche, la perception de contrôle de la situation, l'absence de préoccupation à propos de soi, un sens altéré du temps, la motivation intrinsèque et, condition fondamentale, l'équilibre entre les exigences de la tâche et les capacités de l'individu (MONETA et CSIKSZENTMIHALYI 1996 ; KAWABATA et MALLETT 2011 ; L. WANG et D. HSIAO 2012 ; FULLAGAR, KNIGHT et SOVERN 2013). Le *flow* procure à un joueur un enchantement qu'il a le désir de revivre, lors d'un processus de « sélection psychologique » (CSIKSZENTMIHALYI et MASSIMINI 1985 ; DELLE FAVE, MASSIMINI et BASSI 2011).

Afin d'obtenir une expérience de jeu optimisée pour atteindre l'état de *flow*, des recommandations ont été données (SWEETSER et WYETH 2005) :

4. Personne qui n'entreprend une activité pour d'autre but que l'intense satisfaction qu'elle procure.

— concentration : le jeu vidéo doit amener le joueur à rester focalisé sur le jeu et les actions à mener ;
— défi : le joueur doit surmonter une suite d'obstacles ou d'énigmes pour progresser ;
— compétence : les actions de jeu doivent nécessiter une certaine virtuosité que les défis passés aident à améliorer ;
— contrôle : le jeu vidéo doit demander des décisions au joueur qui lui donne la sensation de liberté d'influer sur le déroulé des évènements ;
— objectif : le but de chaque action doit sembler évident au premier abord ;
— rétroaction : les actions, défis, évènements du jeu vidéo doivent toujours renvoyer un *feedback* au joueur ;
— immersion : l'environnement doit permettre au joueur de s'impliquer totalement dans l'expérience de jeu, sans effort conscient ;
— interaction : le jeu vidéo doit favoriser des opportunités d'interaction sociales.

L'objectif principal d'un jeu vidéo est donc de maintenir ses joueurs dans un état de *flow*. En état de *flow*, le joueur restera concentré sur les défis et les tâches à exécuter, continuera à apprendre et à développer sa virtuosité pour relever des défis plus complexes et ainsi maintenir son état de *flow*. Les développeurs devront donc adapter les défis et les organiser de façon à maintenir ce *flow*, tout en prenant en compte la variété de son public cible pour que les niveaux de difficultés leur correspondent. Il est donc d'une importance majeure de faire évoluer l'équilibrage des éléments du jeu vidéo pour l'adapter aux modifications de la compétence générale du public.

L'expérience de *flow* survient lorsque le défi et les habiletés se situent à un niveau élevé, leur ensemble d'interactions fournissant huit « canaux expérientiels » (NAKAMURA et CSIKSZENTMIHALYI 2012) formant un « paysage expérientiel » (cf. figure 4.5). Cette conception d'une expérience *flow* qu'on pourrait qualifier d'ultime est forcément moins fréquente, mais n'empêche en rien la possibilité de vivre le *flow* à des « niveaux moins élevés » (CSIKSZENTMIHALYI et BOUFFARD 2017).

4.5.4. Défi, incertitude, difficulté et virtuosité

4.5.4.1. Défi

Les défis ou épreuves, souvent appelés *challenges*, sont de différentes catégories car ils nécessitent la mise en œuvre d'actions non-triviales de type différents pour être résolus (MADER 2015).

4.5.4.2. Incertitude

L'incertitude est indissociable du défi, lui-même indissociable du jeu vidéo, et l'incertittude de résolution d'un défi, autrement dit lorsqu'un joueur n'est jamais sûr de surmonter une épreuve, signifie la présence d'un défi ayant un certain niveau de difficulté (MALONE 1980). Un déroulement connu d'avance, sans possibilité d'erreur ou de surprise, conduisant clairement à un résultat inéluctable, est incompatible avec la nature du jeu vidéo. Il faut un renouvellement constant et imprévisible de la

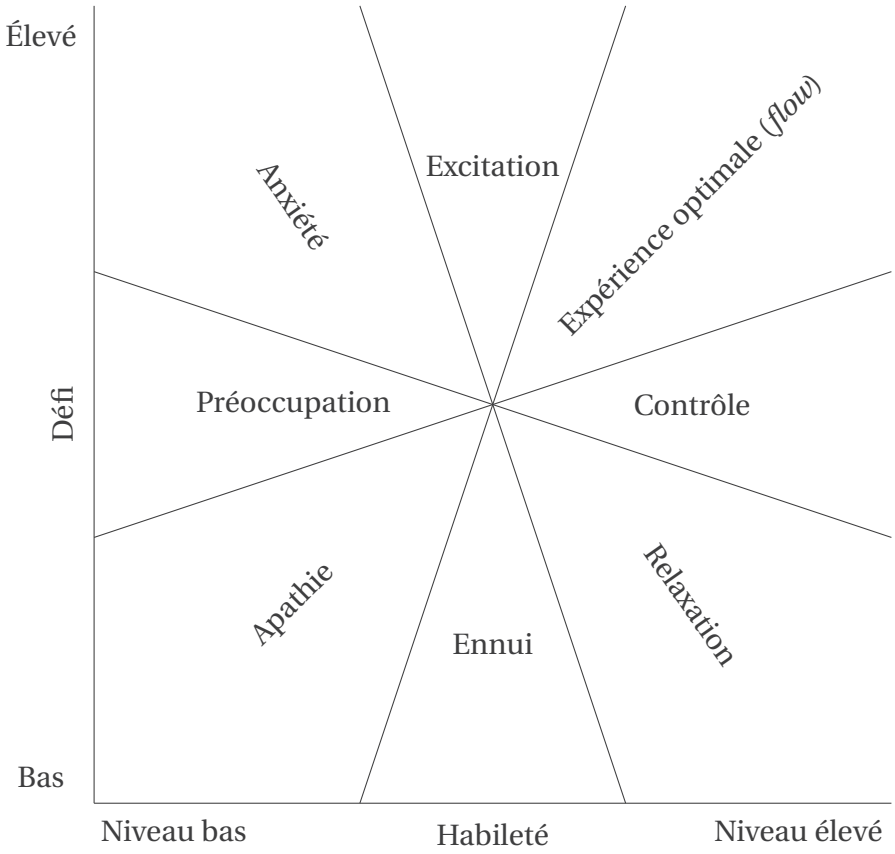


FIGURE 4.5. – Représentation du « paysage expérientiel » du *flow*

situation. Le jeu vidéo consiste dans la nécessité de trouver, d'inventer immédiatement une réponse qui est libre dans les limites des règles. Cette latitude du joueur, cette marge accordée à son action est essentielle au jeu vidéo et explique en partie le plaisir qu'il suscite (CAILLOIS 1958).

4.5.4.3. Difficulté

Nous pouvons considérer que la difficulté, ou niveau de difficulté, est une variable d'ajustement qui permet une oscillation constante du joueur entre l'échec et la réussite. La proportion d'échec attendu ou acceptable dépend du type de jeu vidéo. La difficulté est donnée, notamment, par les ressources mises à disposition du joueur pour atteindre l'objectif fixé. Ces ressources donnent des moyens qui sont rarement des plus efficaces, et poussent ainsi le joueur à utiliser des compétences, sa virtuosité, et à effectuer des action non-triviales (FULLERTON 2008 ; ADAMS 2009).

Nous pouvons considérer différentes catégories de difficulté des défis : sensorielle, logique et motrice. Pour un même genre de jeu vidéo, ce sont les mêmes types de difficultés qui sont plus ou moins élevées (LEVIEUX 2011). Un grand nombre d'auteurs considère que l'idéal est de mettre en place un système d'adaptation dynamique de la difficulté (HUNICKE 2005 ; ANDRADE, RAMALHO, SANTANA et al. 2005 ; SPRONCK, PONSEN, SPRINKHUIZEN-KUYPER et al. 2006 ; YANNAKAKIS et HALLAM 2007 ; YUN, TREVINO et HOLTKAMP 2010 ; MISSURA 2015).

4.5.4.4. Virtuosité

Il existe une relation directe entre difficulté/virtuosité et apprentissage/progression. Selon la définition classique, la virtuosité est le talent d'exécution, l'habileté technique dans la pratique, mais il ne s'agit pas uniquement de dextérité. Définir la virtuosité, ou plutôt les virtuosités, pour un jeu vidéo, c'est anticiper le niveau de maîtrise de ses différents joueurs. Mais c'est aussi donner une illusion au joueur, quel que soit son niveau. En effet, tout joueur cherche instinctivement une adéquation entre ce qu'il fait et la reconnaissance que cela engendre, totalement liée à la réussite de ses actions, et est l'essence même du plaisir de jeu (ALBINET 2015). S'il réussit à les enchaîner, le joueur entre alors en état de *flow* (cf. section 4.5.3).

4.5.5. Boucle de jeu

4.5.5.1. Interaction jeu-joueur

L'interaction jeu-joueur est un processus cyclique dans lequel deux agents actifs alternativement (et métaphoriquement) écoutent, pensent et parlent (CRAWFORD 2003). Le premier agent est le jeu vidéo, le second le joueur. Nous retrouvons peu ou prou ce cycle chez d'autres auteurs (DJAOUTI, ALVAREZ, JESSEL et al. 2008 ; PERRON 2006). La pratique d'un jeu vidéo permet à un joueur d'apprendre, pour ce même jeu vidéo, à analyser plus efficacement les situations de jeu à venir, la virtuosité requise, la maitrise des actions complexes et les connaissances stratégiques et tactiques nécessaires.

4.5.5.2. Boucle OCR

La boucle de jeu OCR (pour *objective, challenge, reward* ou objectif, défi et récompense) structure, de manière motivationnelle, les actions à accomplir par un joueur en trois étapes dont l'ordre ne peut être changé. Comme le langage possède ses phonèmes, ses morphèmes, ses mots et ses phrases, le jeu vidéo possède la boucle de jeu, et il existe plusieurs tailleurs de boucles équivalentes aux différentes unités de langage (KLIMMT 2003 ; ALBINET 2015) :

— boucle micro, constituée de situations de jeu, ou de simples entrées/sorties : cette boucle d'interaction unitaire favorise grandement la « motivation d'effectance »[5] (WHITE 1959) ;
— boucle moyenne, constituée de « missions », « d'épisodes » , qui peuvent être catégorisés comme menaçants ou non, le plus souvent en alternance plus ou moins claire pour un meilleur contrôle du rythme de jeu vidéo ;
— boucle macro, constituée de niveaux, de sessions, pouvant donner au joueur la possibilité d'appréhender des univers différents ou des approches différentes du concepteur de jeu vidéo.

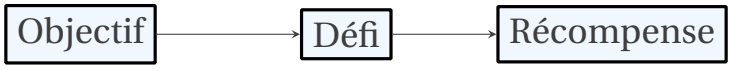


FIGURE 4.6. – Boucle OCR

4.5.6. Motivations

Nous distinguons deux types de motivation (RYAN et DECI 2000) :

— lorsqu'un individu exécute une tâche avec un objectif extérieur, par exemple ranger la table à manger pour pouvoir après l'utiliser pour un jeu de plateau, il s'agit d'une motivation extrinsèque, la réussite de cette tâche n'apporte aucun plaisir direct ;
— lorsqu'un individu exécute une tâche avec un objectif intérieur, par exemple ranger la table à manger pour le simple plaisir d'avoir du rangement, il s'agit de motivation intrinsèque, la réussite de la tâche apporte seule de la satisfaction.

Si un joueur n'est pas motivé, il ne jouera pas. Si sa motivation est poussée par un gain dans le monde réel, ou par une forme de pression sociale (jouer avec des amis), alors la motivation est extrinsèque et le jeu vidéo n'a alors aucune valeur réelle pour le joueur. Si le joueur joue pour le jeu vidéo lui-même, alors il pourra y trouver du plaisir, et peut-être même une forme d'autodétermination (ALBINET 2015).

L'autodétermination (ou SDT pour *self-determination theory*) dans le jeu vidéo repose sur les principes suivants de la théorie PENS (pour *player experience of need*

5. Besoin de compétence, besoin inné qu'a l'être humain de sentir qu'il a le pouvoir d'agir sur son environnement et de produire un impact sur ce dernier, ainsi que de se sentir efficace dans les actions qu'il entreprend pour satisfaire ce besoin.

satisfaction) (RIGBY et RYAN 2007), reprenant le défi, la féérie (*fantasy*), et la curiosité de MALONE 1980 ; MALONE 1981 :

— l'autonomie : le sentiment d'autonomie est plus fort quand le joueur n'a pas conscience des contraintes imposées par les concepteurs, et donc quand le jeu vidéo est au plus intuitif ;
— la compétence : le sentiment de compétence est plus fort quand les défis proposés nécessitent d'utiliser au mieux les capacités et connaissances apprises dans les défis passés. Cela est intimement lié avec la théorie du *flow* ;
— l'appartenance : le sentiment d'appartenance est plus fort quand les relations aux autres sont plus fortes, dans des jeux vidéo où les joueurs doivent coopérer ou s'affronter ; cela n'est pas nécessairement inhérent au jeu vidéo, mais cela peut découler d'une communication externe, que ce soit des réseaux sociaux, des centralisateurs de score ou autres ; l'appartenance passe donc par le prisme des autres joueurs et du lien qu'il se crée avec eux.

4.6. Équilibrage

L'équilibrage d'un jeu est une phase de sa conception qui concerne le réglage du *gameplay* pour créer une expérience de jeu optimale pour les joueurs. Cela implique l'ajustement de différents éléments du jeu, tels que les mécanismes de jeu, les règles et les éléments de l'interface utilisateur, afin d'assurer que le jeu soit amusant et engageant. Cette étape est particulièrement importante dans les jeux vidéo, où il est essentiel de maintenir un bon équilibre pour garantir une expérience de jeu satisfaisante pour les joueurs.

Pour être optimal, un jeu doit être juste pour tous les joueurs. Cela veut dire que personne ne doit avoir un avantage sur les autres joueurs. Pour maintenir l'équilibre, il est possible de créer un jeu symétrique en offrant à chaque joueur les mêmes possibilités et ressources. Cela permet de garantir que personne n'a un avantage injuste sur les autres. Cependant, beaucoup de jeux vidéo sont asymétriques par essence, ce qui peut amener des stratégies de jeu dominantes (cf. sous-section 4.6.2). La question de leur équilibrage se pose donc, est constitue un problème épineux au centre des préoccupations des concepteurs de jeux vidéo. L'équilibrage d'un jeu peut être notamment difficile car chaque jeu est unique et comporte de nombreux éléments différents qui doivent être ajustés pour créer une expérience de jeu équilibrée.

Exemple 4.2 - Dans les jeux de combat [6] (cf. figure 4.7), il est courant de proposer aux joueurs le choix entre plusieurs personnages, chacun avec ses propres compétences, forces et faiblesses uniques, ainsi que des attaques différentes. Lorsque certains personnages s'avèrent plus efficients que d'autres, ils sont alors massivement, voire parfois exclusivement, choisis par les joueurs. C'est une situation indésirable, rendant

6. Le jeu de combat (ou *fighting game*), également appelé *versus fighting* ou plus familièrement « jeu de baston », est un genre de jeu vidéo opposant un nombre limité de personnages (deux le plus souvent), une des expressions la plus simple et directe pour opposer deux joueurs

obsolète les autres personnages et les possibilités qu'ils offrent.

FIGURE 4.7. – Street Fighter II' : Champion Edition, exemple de jeu vidéo de combat [a]

a. Capture d'écran par Capcom Co., Ltd.

4.6.1. Domaines d'équilibrage

Nous distinguons douze domaines d'application de l'équilibrage de jeux vidéo (SCHELL 2014) :

1. l'équité : aucun joueur ne bénéficie d'un avantage sur un autre ;
2. le défi et la difficulté par rapport au succès : préserver le *flow* ;
3. le choix : chaque stratégie doit constituer une alternative viable ;
4. la place des compétences des joueurs relativement à la chance ;
5. la réflexion face à la dextérité ;
6. la coopération par rapport à la compétition ;
7. la durée de jeu de chaque situation de jeu ;
8. les récompenses : octroyées avec la bonne fréquence, et face aux défis adaptés ;
9. les « punitions » : chaque « sanction » doit être juste, compréhensive et adaptée ;
10. la part de liberté contre celle d'expérience contrôlée ;
11. la pondération entre simplicité et complexité ;
12. la proportion de niveau de détail en fonction de l'appel à l'imagination.

4.6.2. La théorie des jeux

La théorie des jeux est une branche des mathématiques qui étudie les situations dans lesquelles les joueurs ou les groupes de joueurs doivent prendre des décisions en fonction des actions potentielles des autres. Elle propose une description formelle d'interactions stratégiques des joueurs, appelés agents (ZERMELO 1912 ; BOREL 1921 ; NEUMANN et MORGENSTERN 1972). La théorie des jeux peut être appliquée à de nombreux domaines, aussi divers que variés, notamment l'équilibrage de jeux vidéo.

L'optimum de Pareto fait partie de la théorie des jeux, il est présenté au paragraphe 2.4.2.1 p. 75.

4.6.2.1. Forme normale d'un jeu

Un jeu se compose de tous les joueurs impliqués, des différentes stratégies disponibles pour chaque joueur et des conséquences ou avantages pour chaque combinaison de stratégies (OSBORNE et RUBINSTEIN 1994).

Définition 4.1 - Un jeu fini sous forme normale (ou jeu sous forme stratégique) $\langle N, A, u\rangle$ de n joueurs est défini de la manière suivante :

— l'ensemble des joueurs $N = \{1, \ldots, n\}$ est un ensemble fini de n, indexé par i ;
— l'ensemble des stratégies A_i pour un joueur i est un profil d'action si et seulement si $a = (a_1, \ldots, a_n) \in A = A_1 \times \ldots \times A_n$;
— une fonction utilitaire ou une fonction de gain $u_i : A \mapsto \mathbb{R}$ pour un joueur i est une fonction profile ou utilitaire si et seulement si $u = (u_1, \ldots, u_n)$.

4.6.2.2. Meilleure réponse

La meilleure réponse est la stratégie, ou l'ensemble de stratégies, d'un joueur qui produisent le résultat immédiat le plus favorable, étant donné les stratégies des autres joueurs.

Définition 4.2 - La meilleure réponse MR est définie de la manière suivante :

$$a_i^* \in MR(a_{-i}) \text{ si et seulement si } \forall a_i \in A_i, u_i(a_i^*, a_{-i}) \geq u_i(a_i, a_{-i})$$

4.6.2.3. Équilibre de Nash

Un équilibre de Nash est une situation où (NASH 1950) :
— chaque joueur prévoit correctement le choix des autres ;
— chaque joueur maximise son gain, compte tenu de cette prévision.

Définition 4.3 - $a = \langle a_1, \ldots, a_n\rangle$ est un équilibre de Nash si et seulement si :

$$\forall i, a_i \in MR(a_{-i})$$

4.6.2.4. Dominance stratégique

La dominance stratégique, ou stratégie dominante, survient lorsqu'une stratégie est meilleure pour un joueur qu'une autre stratégie, quelle que soit la stratégie d'un autre joueur.

Définition 4.4 - Soit s_i et s'_i deux stratégies pour un joueur i, et soit S_{-i} l'ensemble de toutes les stratégies possibles qui sont des profils d'action pour les autres joueurs :
— s_i domine strictement s'_i si $\forall s_{-i} \in S_{-i}, u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$;
— s_i domine faiblement s'_i si $\forall s_{-i} \in S_{-i}, u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$.

4.7. Conclusion

Il n'y a pas, à ce jour, de théorie rationnelle complète pour la conception de jeux vidéo et aucune formule simple pour créer des jeux vidéo de qualité (Schell 2014). D'ailleurs, la plupart des approches utilisées pour la conception de jeux vidéo n'ont pas été évaluées de manière scientifique (Neil 2012). Le principe même d'un jeu vidéo nous échappe encore, de telle sorte que nombreux sont ceux tentant de le définir et de le circonscrire (Huizinga 1951 ; Caillois 1958 ; Abt 1970 ; Crawford 1984 ; Salen et Zimmerman 2004 ; Brougère 2005).

Bien que ce chapitre ne suffise probablement pas à décrire l'ensemble de l'extraordinaire complexité qu'est la conception d'un jeu vidéo, et notamment son *gameplay* et sa boucle de jeu (Crawford 2003), nous avons pu en donner une approche formelle et aborder les aspects les plus significatifs pour notre sujet, et notamment l'équilibrage de jeu vidéo, en particulier par rapport à son audience et son type. La classification, que ce soit des modèles de joueurs ou bien des jeux vidéo eux-même, est fondamentale dans le domaine de la conception de jeux vidéo. Pourtant, en la matière, il ne semble pas exister actuellement de proposition arrêtée faisant foi ou acceptée du plus grand nombre. Nous avons fait le choix de présenter les principes qui nous semblaient les plus cohérents, les plus adoptés et les plus pertinents concernant notre approche, et en particulier l'approche atomique et ses briques *gameplay* (Alvarez 2018).

Concevoir un jeu vidéo, c'est tenter de véhiculer une expérience ludique à un ensemble de joueurs en assemblant des mécaniques et une esthétique pour créer une dynamique satisfaisante pour le public (Hunicke, Leblanc et Zubek 2004). Pour ce faire, il faut aux concepteurs appliquer un ensemble de théories décrivant la psychologie du joueur et comment il est possible d'orienter divers éléments du jeu vidéo pour renforcer la motivation des différents profils de joueurs. Parmi elles, le *fun* (Koster 2014), le *flow* (Csikszentmihalyi 2009), les motivations (Ryan et Deci 2000), les défis offerts par les jeux vidéo ainsi que leurs incertitudes, leurs difficultés et la virtuosité nécessaire pour les surmonter au mieux nous paraissent essentiels (Albinet 2015). Il est nécessaire que les décideurs adaptent ces concepts à leurs problématiques et à leur public pour réellement avoir un impact sur l'expérience d'un joueur.

L'équilibrage de jeu vidéo est un processus qui permet le réglage du *gameplay* d'un jeu vidéo afin de créer une expérience de jeu optimale pour les joueurs selon un

ensemble de domaines d'application (SCHELL 2014). Cela implique l'ajustement des différents éléments du jeu, tels que les mécanismes de jeu, les règles et les éléments de l'interface utilisateur, pour s'assurer que le jeu soit amusant et engageant. L'équilibrage d'un jeu vidéo est une étape importante dans sa conception, car cela peut avoir un impact significatif sur la façon dont les joueurs perçoivent et apprécient le jeu. Cela peut inclure l'ajustement de la difficulté, la durée de vie, la progression et d'autres éléments clefs du jeu afin de créer une expérience de jeu équilibrée et agréable pour les joueurs. Parmi les entraves pouvant se dresser à l'encontre de ce processus, nous trouvons la dominance stratégique. Elle peut être un problème dans un jeu vidéo, en le rendant peu intéressant pour les joueurs qui ne sont pas en mesure de la surmonter. Les concepteurs de jeux vidéo doivent donc souvent veiller à éviter cette situation en mettant en place des mécanismes et en ajustant régulièrement l'équilibre de leur dispositif ludique. A cette fin, les apports de la théorie des jeux peuvent s'avérer très utiles (ZERMELO 1912 ; BOREL 1921 ; NASH 1950 ; NEUMANN et MORGENSTERN 1972). Bien que ce processus soit des plus délicats, il n'existe pas à ce jour, à notre connaissance, de solution automatisée formalisée satisfaisante pour assister le concepteur de jeu vidéo dans l'équilibrage de jeux vidéo.

Deuxième partie

Contribution

5. Cubes de données hiérarchiques

Sommaire

5.1. Introduction et motivations . . . 143
5.2. Cas d'utilisation . . . 144
5.2.1. Open Match 3 . . . 144
5.2.2. Classification atomique d'Open Match 3 . . . 146
5.2.3. Relation exemple . . . 147
5.3. Dimension hiérarchique . . . 153
5.3.1. Types de hiérarchies . . . 153
5.3.1.1. Hiérarchie simple . . . 153
5.3.1.2. Hiérarchie stricte et hiérarchie non-stricte . . . 153
5.3.1.3. Hiérarchie multiple . . . 153
5.3.1.4. Hiérarchies parallèles . . . 153
5.3.2. Cadre formel d'une dimension hiérarchique . . . 154
5.3.3. Définition formelle d'une dimension hiérarchique . . . 156
5.3.3.1. Espace multidimensionnel . . . 156
5.3.3.2. Ordres de spécialisation . . . 156
5.3.3.3. Fonctions . . . 158
5.3.3.4. Opérateurs . . . 158
5.3.3.5. Caractérisation du treillis cube hiérarchique . . . 162
5.4. Cube de données hiérarchique . . . 166
5.5. Cube de données hiérarchique fermé . . . 169
5.6. Conclusion . . . 172

5.1. Introduction et motivations

Comme nous l'avons vu, et contrairement aux bases de données opérationnelles, et outre leur volume plus important, les entrepôts de données multidimensionnels (cf. 2.2.1 p. 27), que nous appellerons de manière simplifiée entrepôts de données, ont des particularités notables : les décideurs à qui ils sont dédiés sont peu nombreux et s'intéressent davantage à des tendances globales plutôt qu'aux détails des données. Les données des entrepôts sont également caractéristiques puisque plutôt soumises à une faible fréquence de mise à jour. Enfin, l'exploitation des données, bien que s'inscrivant également dans un processus d'aide à la décision (OLAP), se distingue des requêtes dans le monde relationnel par des traitements plus complexes, plus « coûteux », moins régulier mais aussi plus agrégés. Ainsi, les interrogations faites

sur un entrepôt de données (*datawarehouse* ou *datamart*) sont déterminées progressivement de manière *ad hoc*, et idéalement de façon interactive. Les calculs sur les entrepôts de données sont donc longs, et il est nécessaire de générer les résultats à l'avance.

Cependant, comme il n'est pas évident de prévoir les désidératas des décideurs, les calculs préalables doivent considérer l'ensemble de tous les résultats possibles. C'est cette idée qui nous a encouragé à intégrer dans ces réponses les dimensions hiérarchiques (HURTADO et A. O. MENDELZON 2002). Pour répondre à ce besoin, nous proposons, dans ce chapitre, une redéfinition complète du cadre et de la définition formelle de l'analyse des bases de données traditionnelles (KUIJPERS et VAISMAN 2016), du cube de données, et de son cube de données fermé par le prisme des dimensions hiérarchiques. Les hiérarchies ajoutent leur propre complexité de modélisation à celles des entrepôts de données (MALINOWSKI et ZIMÁNYI 2004), c'est pour cette raison, et les temps de calcul conséquents qu'elles induisent, que peu d'auteurs se sont intéressés à ce domaine. Pourtant, nous pensons qu'un travail de cadrage et de définition formelle des dimensions hiérarchiques est judicieux. Par ailleurs, de manière générale, nous considérons deux classes d'opérateurs OLAP. La première est dédiée à la structure des données et en permet sa manipulation afin de récupérer des informations remarquables. La seconde est dédiée à la granularité des données et propose des opérateurs qui agrègent et synthétisent les informations dans un sens (*roll up*), et qui les décomposent ou les précisent dans l'autre (*drill down*) (FAVRE, BENTAYEB, BOUSSAID et al. 2013). Les hiérarchies dimensionnelles définissent les chemins d'accès dans les données et permettent la mise en œuvre de cette seconde classe d'opérateurs OLAP. C'est également pour cette raison qu'étudier les dimensions hiérarchiques est pertinent bien que le volume de données généré, encore plus important que dans le cas du cube de données classique, doit être appréhender par une approche particulièrement réduite pour être tout à fait exploitable.

Dans ce chapitre, après avoir présenté les dimensions hiérarchiques, nous introduisons les cubes de données hiérarchiques (MARTIN NEVOT, NEDJAR et LAKHAL 2023).

5.2. Cas d'utilisation

5.2.1. Open Match 3

Pour ce chapitre, nous introduisons un nouvel exemple appliqué à un jeu vidéo, celui que nous avons étudié et expérimenté initialement lors de la recherche de cette contribution.

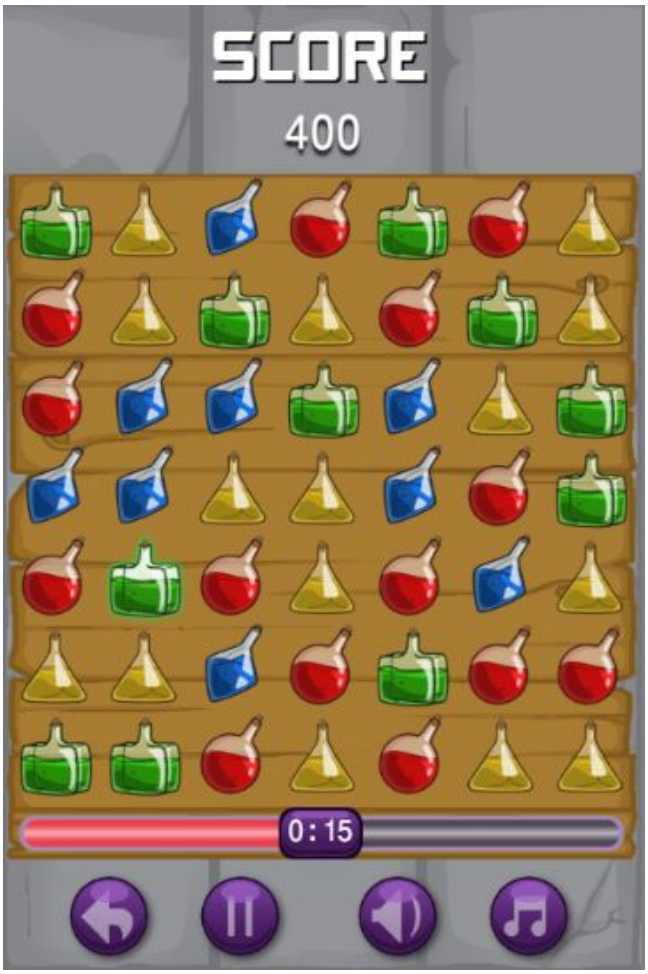


FIGURE 5.1. – Jeu vidéo Open Match 3

Nous utiliserons dans cet exemple le jeu vidéo *puzzle-game*[1] de type Match 3[2] *open source*[3] Open Match 3, jeu vidéo sur navigateur Web[4], développé par Giordano Ferdinandi, Stephen Surtees et Rachel Kehoe de la société Clockwork Chilli.

Initialement, nous avons porté notre préférence sur ce type de jeu vidéo pour son immense popularité, rendant l'exemple plus éloquent et son application pertinente. Par ailleurs, outre son côté éthique, ce choix de *l'open source* nous a permis d'avoir accès et de modifier directement le code source du jeu vidéo afin d'y loger toutes les sondes dont nous souhaitions être à l'écoute.

5.2.2. Classification atomique d'Open Match 3

Nous estimons que le cœur du *gameplay* du jeu vidéo Open Match 3 est composé des briques de *gameplay* (cf. sous-sous-section 4.4.2.4 p. 128) `Atteindre` (`At`), `Détruire` (`Dt`), `Gérer` (`Gr`), `Choisir` (`Ch`) et `Déplacer` (`Dp`) (cf. figure 5.2).

FIGURE 5.2. – Briques *gameplay* d'Open Match 3

1. Un jeu vidéo de *puzzle* ou *puzzle-game* est un type de jeu vidéo basé sur la réflexion. Son nom provient du *puzzle*, jeu consistant à mettre en ordre des pièces. Ce jeu de réflexion consiste à placer des objets d'une façon ou dans un ordre spécifique, ce qui le distingue des autres jeux de réflexion. Il peut s'agir d'un jeu dans lequel le joueur doit déplacer des éléments de manière à reproduire une forme prédéterminée ou à regrouper des couleurs spécifiques. Nombre de ces jeux sont dits des jeux de *tile-matching*, ou de « correspondance de tuiles ».

2. Un jeu match 3 est un type de jeu vidéo dans lequel le joueur doit combiner trois éléments de la même couleur ou de la même forme afin de les éliminer du plateau de jeu. Ces jeux sont souvent considérés comme des jeux de réflexion et sont généralement basés sur la combinaison de stratégie et de chance. Les jeux match 3 peuvent être joués sur diverses plateformes, y compris les ordinateurs, les téléphones mobiles et les consoles de jeux, et ils sont généralement très populaires auprès de différents groupes d'âge. Les jeux match 3 peuvent être de simples divertissements ou ils peuvent offrir des défis de plus en plus complexes au fur et à mesure que le joueur progresse dans le jeu.

3. La désignation *open source* désigne une approche de développement de logiciels (et s'étend maintenant aux œuvres de l'esprit) dont la licence respecte des critères précisément établis par l'Open Source Initiative (`https://opensource.org/`) et dans laquelle le code source est disponible librement et peut être utilisé, modifié et distribué par n'importe qui. L'objectif de l'approche *open source* est de permettre à un plus grand nombre de personnes de contribuer au développement et à l'amélioration d'un logiciel, ce qui peut entraîner une innovation accrue et une qualité améliorée. Les licences *open source* précisent les conditions dans lesquelles le code source peut être utilisé, modifié et distribué. Certaines licences exigent que les modifications apportées au code source soient publiées de manière à ce que d'autres personnes puissent en bénéficier, tandis que d'autres licences permettent à l'utilisateur de garder ses modifications privées.)

4. Un jeu vidéo sur navigateur Web est un type de jeu en ligne jouable par l'intermédiaire d'un navigateur Web ne nécessitant ni installation, ni téléchargement.

5.2.3. Relation exemple

Considérons l'entrepôt de données OM3 (cf. figure 5.3) du jeu vidéo Open Match 3 (cf. figure 5.1).

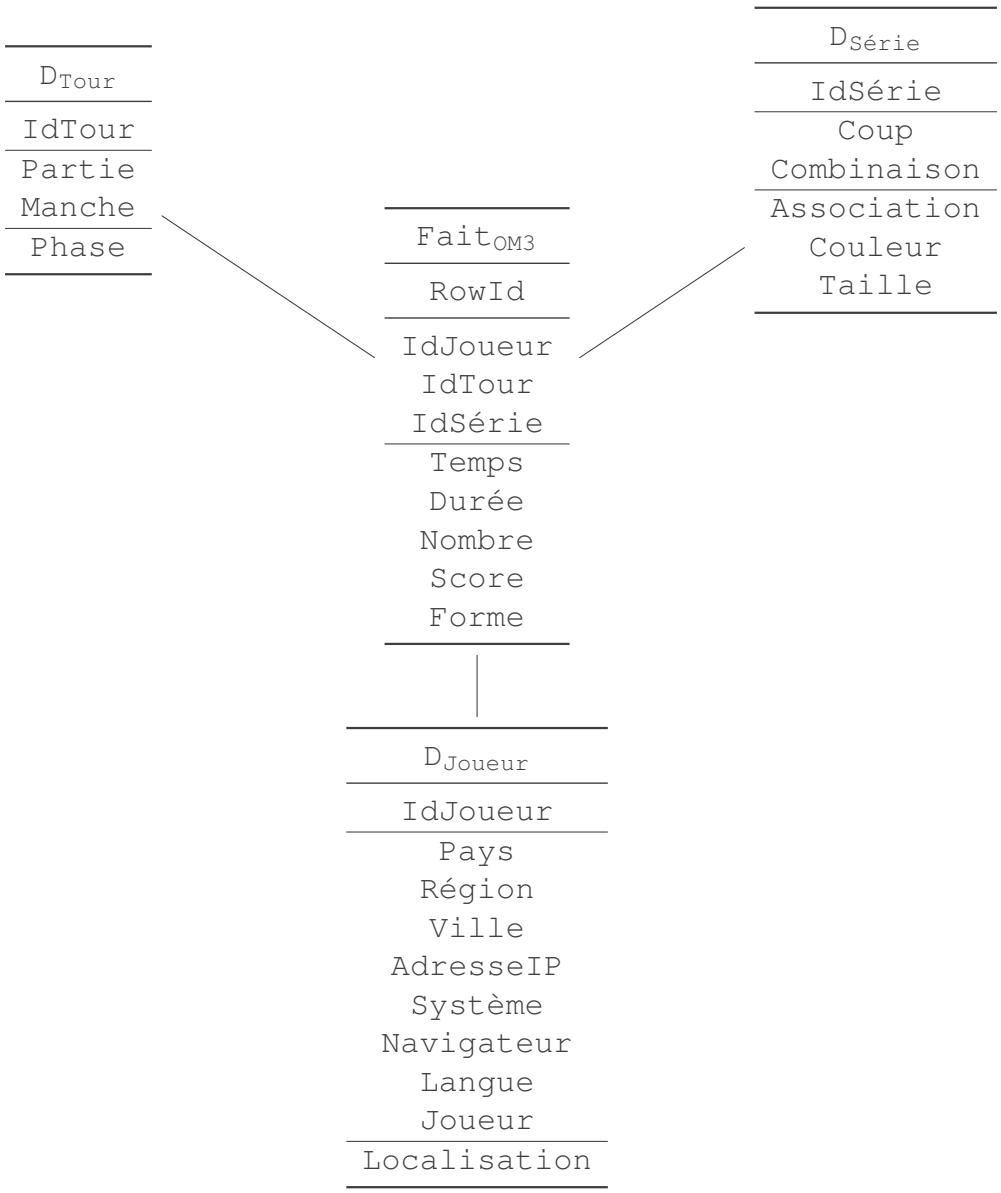


FIGURE 5.3. – Schéma en étoile de l'entrepôt OM3

Un D_{Tour} est identifié par IdTour et classé par Partie et par Manche. Ces attributs constituent une structure hiérarchique, appelée Phase (cf. tableau 5.2).

Un $D_{Série}$ est identifié par IdSérie et caractérisé par un Coup occasionnant une (ou plusieurs, alors appelé chaine) Combinaison d'une Association, caractérisée par une même Couleur (parmi les quatre) et par une Taille d'au moins trois éléments. À partir d'une Association de quatre éléments, des bonus de points et

des éléments spéciaux (explosion, rayon destructeur, etc., cf. figure 5.4) sont accordés (cf. tableau 5.4).

Un $\mathtt{D}_{\mathtt{Joueur}}$ est identifié par `IdJoueur` et est classé par sa `Localisation` (complète). Le type de localisation le plus général est un `Pays` (`P`), qui englobe une `Région` (`R`), elle-même englobant une `Ville` (`V`), précisée par une `AdresseIP` (`A`), elle-même précisée par un `Système` (`S`, *i.e.* un système d'exploitation), un `Navigateur` (`N`), puis une `Langue` (`L`) ainsi, qu'enfin, un nom de `Joueur` (`J`). Les valeurs des différents attributs de $\mathtt{D}_{\mathtt{Joueur}}$ sont codées comme suit (cf. tableau 5.3) :

Pays		Région		Ville		AdresseIP	
France	= 1	*IDF*	= 2	*Paris*	= 3	92.88.91.80	= 4
		PACA	= 9	*Marseille*	= 10	139.124.242.125	= 11

Système		Navigateur		Langue		Joueur	
Windows	= 5	*Chrome*	= 6	*fr*	= 7	J_1	= 8
Linux	= 12	*Opera*	= 13	*en*	= 14	J_2	= 15
Mac OS	= 16	*Firefox*	= 17	*es*	= 18	J_3	= 19

La table des faits $\mathtt{Fait}_{\mathtt{OM3}}$ est composée d'un `Temps` restant avant la fin de la partie (s'égrainant naturellement, mais étant aussi augmenté jusqu'à un maximum avec le `Score`, c'est donc un élément décisionnel), d'une `Durée` de réalisation, d'un `Nombre` d'éléments en correspondance, d'un `Score` (nombre de points remportés) cumulé total et d'une description de `Forme` (0.5 pour horizontal, 1 pour vertical, et entre ces deux valeurs pour mixte) pour un `IdJoueur` (`IdJ`), un `IdTour` (`IdT`) et un `IdSérie` (`IdS`). Les valeurs des différents attributs de $\mathtt{Fait}_{\mathtt{OM3}}$ sont codées comme suit (cf. tableau 5.1) :

IdJ		IdT		IdS	
J_1	= 8	P_1	= 1	A_1	= 1
J_2	= 15	P_2	= 4	A_2	= 4
J_3	= 19	P_3	= 8	A_3	= 11
				A_4	= 19
				A_5	= 21
				A_6	= 23
				A_7	= 25

Nous ne donnons de représentation d'une dimension hiérarchique que pour $\mathtt{D}_{\mathtt{Joueur}}$ (cf. figure 5.5), $\mathtt{D}_{\mathtt{Tour}}$ et $\mathtt{D}_{\mathtt{Série}}$ étant des dimensions hiérarchiques nettement moins profondes que celle-ci. L'élément maximal $ALL_{\mathtt{Joueur}}$ de cette représentation est défini par la définition 5.5 à la sous-section 5.3.2.

FIGURE 5.4. – Jeu vidéo Open Match 3, exemple d'explosion

TABLEAU 5.1. – Relation de la table des faits $\texttt{Fait}_{\text{OM3}}$

RowId	IdJ	IdT	IdS	Temps	Durée	Nombre	Score	Forme
1	J_1	P_1	A_1	6.32	2.85	3.5	700	0.5
2	J_1	P_1	A_2	18.9	1.95	3.83	2300	0.5
3	J_2	P_2	A_3	26.39	1.7	3.43	2400	0.71
4	J_2	P_2	A_4	4.1	2.07	3	300	0.5
5	J_2	P_2	A_5	7.38	3.68	3	600	0.5
6	J_3	P_3	A_6	2.14	2.15	3	300	1
7	J_3	P_3	A_7	56.04	2.25	3.17	3800	0.5

TABLEAU 5.2. – Relation de la dimension D_{Tour}

IdTour	Partie	Manche	Phase
1	1		P_1
2	1	2	P_{1-1}
3	1	3	P_{1-2}
4	4		P_2
5	4	5	P_{2-1}
6	4	6	P_{2-2}
7	4	7	P_{2-3}
8	8		P_3
9	8	9	P_{3-1}
10	8	10	P_{3-2}
11	8	11	P_{3-3}

TABLEAU 5.3. – Relation de la dimension D_{Joueur}

IdJoueur	P	R	V	A	S	N	L	J	Localisation
1	1								*France*
2	1	2							*IDF*
3	1	2	3						*Paris*
4	1	2	3	4					92.88.91.80
5	1	2	3	4	5				*Windows*
6	1	2	3	4	5	6			*Chrome*
7	1	2	3	4	5	6	7		*fr*
8	1	2	3	4	5	6	7	8	J_1
9	1	9							*PACA*
10	1	9	10						*Marseille*
11	1	9	10	11					139.124.242.125
12	1	9	10	11	12				*Linux*
13	1	9	10	11	12	13			*Opera*
14	1	9	10	11	12	13	14		*en*
15	1	9	10	11	12	13	14	15	J_2
16	1	9	10	11	16				*Mac OS*
17	1	9	10	11	16	17			*Firefox*
18	1	9	10	11	16	17	18		*es*
19	1	9	10	11	16	17	18	19	J_3

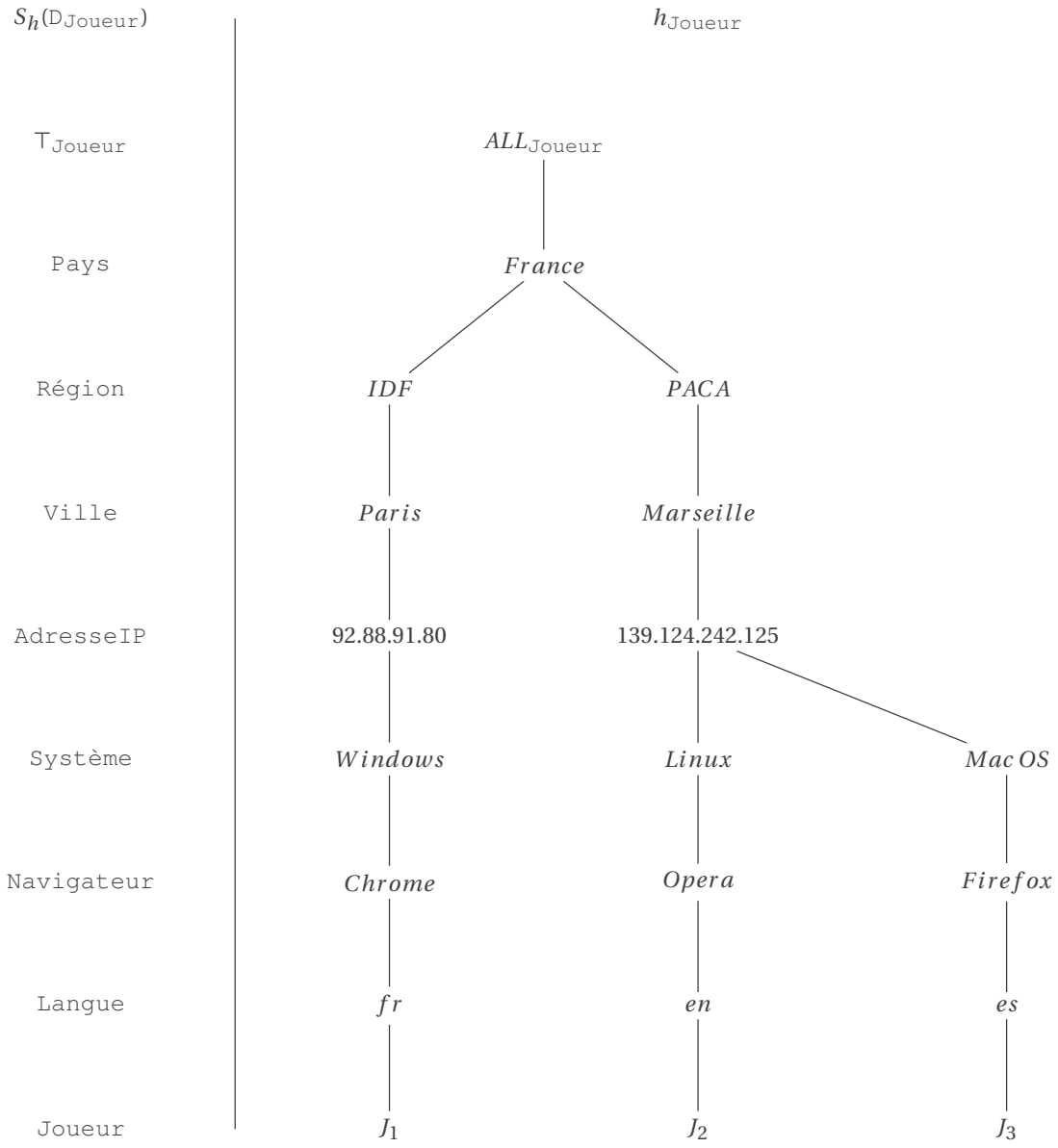


FIGURE 5.5. – Dimension de $\mathrm{D}_{\mathrm{Joueur}}$

TABLEAU 5.4. – Relation de la dimension $D_{Série}$

IdSérie	Coup	Combinaison	Association	Couleur	Taille
1	1		A_1		
2	1	2	A_{1-1}	*rouge*	3
3	1	3	A_{1-2}	*bleu*	3
4	4		A_2		
5	4	5	A_{2-1}	*vert*	4
6	4	6	A_{2-2}	*jaune*	3
7	4	7	A_{2-3}	*rouge*	3
8	4	8	A_{2-4}	*bleu*	3
9	4	9	A_{2-5}	*jaune*	3
10	4	10	A_{2-6}	*vert*	4
11	11		A_3		
12	11	12	A_{3-1}	*vert*	3
13	11	13	A_{3-2}	*red*	3
14	11	14	A_{3-3}	*red*	4
15	11	15	A_{3-4}	*vert*	3
16	11	16	A_{3-5}	*jaune*	4
17	11	17	A_{3-6}	*jaune*	3
18	11	18	A_{3-7}	*jaune*	3
19	19		A_4		
20	19	20	A_{4-1}	*vert*	3
21	21		A_5		
22	21	22	A_{5-1}	*rouge*	3
23	23		A_6		
24	23	24	A_{6-1}	*vert*	5
25	25		A_7		
26	25	26	A_{7-1}	*jaune*	3
27	25	27	A_{7-2}	*jaune*	3
28	25	28	A_{7-3}	*rouge*	3
29	25	29	A_{7-4}	*rouge*	3
30	25	30	A_{7-5}	*jaune*	3
31	25	31	A_{7-6}	*rouge*	4
32	25	32	A_{7-7}	*vert*	3
33	25	33	A_{7-8}	*jaune*	3
34	25	34	A_{7-9}	*jaune*	3
35	25	35	A_{7-10}	*vert*	3
36	25	36	A_{7-11}	*vert*	4
37	25	37	A_{7-12}	*bleu*	3

5.3. Dimension hiérarchique

5.3.1. Types de hiérarchies

Les hiérarchies dimensionnelles sont formées par l'ensemble des attributs d'une relation de dimensions ayant un rapport hiérarchique ($a \in a'$). Elles correspondent à des relations de type 1 à plusieurs et offre la possibilité d'utiliser la classe des opérateurs *roll up* et *drill down* (FAVRE, BENTAYEB, BOUSSAID et al. 2013).

Nous disposons, dans la littérature, d'un panorama des différents types de hiérarchies (INMON 1996) que nous nous proposons de présenter.

5.3.1.1. Hiérarchie simple

Les hiérarchies simples ont des niveaux qui peuvent être considérés comme des listes et leurs instances forment un arbre. Ces hiérarchies peuvent être symétriques selon que l'arbre des instances est équilibré ou asymétriques s'il ne l'est pas.

5.3.1.2. Hiérarchie stricte et hiérarchie non-stricte

Dans le cadre d'une hiérarchie stricte, la généralisation d'une valeur à un niveau donné ne conduit qu'à, au plus, une valeur.

Exemple 5.1 - La généralisation au niveau du mois de la date 14 juillet 1789 est juillet.

En pratique, il existe de nombreuses situations où ce type de hiérarchie est trop restrictif. Typiquement, un jeu vidéo peut être de plusieurs types de jeu. De telles hiérarchies sont dites non strictes.

5.3.1.3. Hiérarchie multiple

Les hiérarchies multiples permettent de modéliser les situations où les différents niveaux ne sont plus des listes mais des graphes orientés sans cycle.

Exemple 5.2 - Considérons une dimension temporelle. Il existe deux façons de généraliser le niveau journée : sur le niveau semaine ou le niveau mois, bien que ces deux niveaux ne soient pas comparables (*i.e.* l'un n'est pas une généralisation de l'autre).

5.3.1.4. Hiérarchies parallèles

Pour les dimensions ne comportant qu'un seul attribut, il ne peut exister qu'une hiérarchie par dimension. En pratique, il est fréquent qu'une même dimension soit composée de plusieurs attributs utiles pour l'analyse.

Si ces attributs possèdent une hiérarchie associée, la hiérarchie de la dimension est dite parallèle et peut être considérée comme un cas particulier des hiérarchies multiples. Une hiérarchie parallèle peut être composée d'autre types de hiérarchies. On distingue généralement deux catégories de hiérarchies parallèles :

— hiérarchie parallèle indépendante : il n'existe pas de niveau commun entre les différentes hiérarchies la composant ;
— hiérarchie parallèle dépendante : il existe des niveaux communs entre les différentes hiérarchies la composant ; la hiérarchie est dite dépendante.

5.3.2. Cadre formel d'une dimension hiérarchique

Soit r une relation de schéma $\mathscr{R}$. En plus des attributs de $\mathscr{R}$, nous considérons un attribut supplémentaire $RowId$ qui est implicite. Les valeurs de cet attribut servent d'identifiant unique à chaque tuple et sont attribuées au moment de leur insertion. On note t_x le tuple tel que $t[RowId] = x$.

Les attributs de $\mathscr{R}$ sont divisés en deux ensembles :

1. $\mathscr{D}$ l'ensemble des attributs dimensions hiérarchiques, appelés aussi catégories hiérarchiques. De plus, les attributs de $\mathscr{D}$ sont ordonnés.
2. $\mathscr{M}$ l'ensemble des attributs mesures, sur lesquels sont appliquées les fonctions agrégatives ou statistiques.

Nous appelons simplement dimension un attribut dimension hiérarchique. De même nous appelons hiérarchie une hiérarchie de valeurs.

Définition 5.1 (Dimension) - Dans r, nous appelons une dimension, un attribut dimension, ou encore une catégorie, d_i. $\forall d_i \in \mathscr{D}$, $r(d_i)$ est la projection de r sur d_i.

Une dimension d_i est caractérisée par une structure (cf. définition 5.2) et une hiérarchie (cf. définition 5.3).

Nous appelons $\mathscr{D}$ l'ensemble des dimensions d_i de r.

Exemple 5.3 - Considérons l'entrepôt de données `OM3`. Son ensemble de dimensions $\mathscr{D}$ est :

$$\mathscr{D} = \{\mathtt{D_{Joueur}}, \mathtt{D_{Tour}}, \mathtt{D_{Série}}\}$$

Définition 5.2 (Structure d'une dimension) - Pour chaque dimension $d_i \in \mathscr{D}$, nous définissons la structure d'une dimension, ou schéma, S_h comme suit : $\forall d_i \in \mathscr{D}$, $S_h(d_i)$ est la structure de la dimension d_i.

Exemple 5.4 - Considérons l'ensemble des structures des dimensions hiérarchiques :

$$\begin{aligned} S_h(\mathtt{D_{Joueur}}) &= (\top_{\mathtt{Joueur}}, \mathtt{Pays}, \mathtt{Région}, \mathtt{Ville}, \mathtt{AdresseIP}, \\ &\qquad \mathtt{Système}, \mathtt{Navigateur}, \mathtt{Langue}, \mathtt{Joueur}) \\ S_h(\mathtt{D_{Tour}}) &= (\top_{\mathtt{Tour}}, \mathtt{Partie}, \mathtt{Manche}) \\ S_h(\mathtt{D_{Série}}) &= (\top_{\mathtt{Série}}, \mathtt{Coup}, \mathtt{Combinaison}) \end{aligned}$$

Définition 5.3 (Hiérarchie) - Pour chaque dimension $d_i \in \mathscr{D}$, nous définissons h_i la hiérarchie associée à cette dimension, que nous nommerons simplement hiérarchie.

Définition 5.4 (Niveau d'une hiérarchie) - Pour chaque hiérarchie h_i, nous définissons $h_i(e)$, le niveau, ou échelon, ou encore étage, e de cette hiérarchie.

Définition 5.5 (Élément maximal d'une hiérarchie) - Une hiérarchie h_i associée à la dimension $d_i \in \mathscr{D}$ contient un élément maximal noté $\top_i$ défini par la valeur ALL_i. Cette valeur reprend l'idée de la valeur *ALL* de GRAY, CHAUDHURI, BOSWORTH et al. 1997, qui représente la généralisation de toutes les valeurs du domaine d'une dimension.

Exemple 5.5 - En considérant la hiérarchie $h_{\texttt{Joueur}}$ de la dimension $\texttt{D}_{\texttt{Joueur}}$ (cf. figure 5.5), la valeur de son élément maximal est $ALL_{\texttt{Joueur}}$.

Définition 5.6 (Profondeur d'une hiérarchie) - Pour chaque hiérarchie h_i, nous définissons sa $Prof$ comme suit : $\forall h_i \in d_i, d_i \in \mathscr{D}, Prof(h_i)$ est la profondeur de la hiérarchie h_i, soit son nombre de niveaux.

Exemple 5.6 - En considérant la hiérarchie $h_{\texttt{Joueur}}$ de la dimension $\texttt{D}_{\texttt{Joueur}}$ (cf. figure 5.5), $Prof(h_{\texttt{Joueur}}) = 9$.

Définition 5.7 (Domaine d'une dimension) - Nous définissons le domaine d'une dimension Dom comme suit : $\forall d_i \in \mathscr{D}, Dom(d_i)$ est le domaine de la dimension d_i.

Définition 5.8 (Domaine d'un niveau d'une dimension) - Nous définissons le domaine d'un niveau d'une dimension, ou empreinte dimensionnelle, Dom comme suit :

— $\forall d_i \in \mathscr{D}, \forall e \in S_h(d_i), Dom(d_i, e)$ est le domaine de la dimension d_i pour le niveau e, *i.e.* $Dom(d_i, e)$ correspond à l'ensemble des valeurs possibles pour le niveau e de la hiérarchie h_i ;

— $\forall d_i \in \mathscr{D}, Dom(h_i) = \bigcup_{e \in S_h(d_i)} Dom(h_i, e)$.

Définition 5.9 (Table de dimension) - Dans le modèle de données « en étoile » utilisé dans les entrepôts de données (*datawarehouse* ou *datamart*), chaque hiérarchie peut être représentée par une table, de dimension $\texttt{D}_{d_i}$ correspondant à la dimension éponyme (les deux concepts sont confondus), de schéma relationnel $S_h(d_i)$.

Définition 5.10 (Tuple dimensionnel) - Pour chaque $x \in Dom(d_i)$, nous définissons le tuple dimensionnel t_x correspondant à la valeur x dans la table $\texttt{D}_{d_i}$ est constitué de la liste des ancêtres de x dans h_i.

De même, pour chaque $x \in Dom(d_i, e)$, nous définissons le tuple dimensionnel t_x correspondant à la valeur x pour le niveau e de la hiérarchie h_i dans la table $\texttt{D}_{d_i}$, également constitué de la liste des ancêtres de x dans h_i.

Nous appelons simplement tuple un tuple dimensionnel.

Exemple 5.7 - Pour $x = J_2$, le tuple t_x correspondant dans la relation $\texttt{D}_{\texttt{Joueur}}$ est $t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, J_2)$.

Pour $x = 139.124.242.125$, le tuple t_x correspondant dans la relation $\texttt{D}_{\texttt{Joueur}}$ est $t_x = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$.

5.3.3. Définition formelle d'une dimension hiérarchique

5.3.3.1. Espace multidimensionnel

Définition 5.11 (Espace multidimensionnel) **-** L'espace multidimensionnel de r est noté et redéfini avec les dimensions hiérarchiques comme suit :

$$Space(r) = \{\times_{d_i \in \mathscr{D}} Dom(h_i) \cup \{ALL_i, \ldots, ALL_j\})\} \cup \{(\emptyset, \ldots, \emptyset)\}$$

Où $\times$ symbolise le produit cartésien, et $(\emptyset, \ldots, \emptyset)$ la combinaison de valeurs vides. Toute combinaison de $Space(r)$ est un tuple et représente un motif multidimensionnel.

La valeur de chaque élément maximal ALL_i d'une hiérarchie h_i est naturellement contenue dans la hiérarchie, dans le domaine $Dom(h_i)$ associé, et donc également dans $Space(r)$.

Exemple 5.8 - L'espace multidimensionnel de l'entrepôt de données `OM3`, et sa table des faits (cf. tableau 5.1), est illustré par le tableau 5.5. Par souci de clarté et de concision, tous les tuples de l'espace multidimensionnel ne sont pas représentés (il y en aurait plusieurs milliers), et les valeurs spéciales sont abrégées : $ALL_{\texttt{Joueur}}$ en $ALL_{\texttt{J}}$, $ALL_{\texttt{Tour}}$ en $ALL_{\texttt{T}}$ et $ALL_{\texttt{Série}}$ en $ALL_{\texttt{S}}$.

Les tuples d'identifiants de valeurs 1, …, 13 et 31 sont des tuples possibles pour r (car toutes leurs valeurs sont réelles), même si le tuple d'identifiant de valeur 31 n'est pas présent explicitement dans la relation r. Dans ce tuple, le symbole $\emptyset$ signifie « valeur vide ». Les autres tuples (identifiants 14, …, 30) ne peuvent pas être des tuples de r car ils comportent au moins une valeur ALL_i. Ils véhiculent donc une information à un niveau de détail plus agrégé que les précédents.

5.3.3.2. Ordres de spécialisation

L'espace multidimensionnel de r est structuré par la relation de spécialisation entre tuples. Cet ordre est originellement introduit par Mitchell 1982 ; Mitchell 1997 dans le cadre de l'apprentissage de concepts.

L'ensemble ordonné $CL(r) = \langle Space(r), \preceq_s \rangle$ est un treillis complet appelé treillis cube.

Définition 5.12 (Ordre de spécialisation intradimensionnelle) **-** Soit $x, y \in Dom(h_i)$:

$$x \preceq_{d_i} y \Leftrightarrow x \text{ est un ancêtre de } y \text{ dans la hiérarchie } h_i.$$

Définition 5.13 (Ordre de spécialisation multidimensionnelle) **-** Soit $t, t' \subseteq Space(r)$, nous définissons la relation d'ordre $\preceq_s$ comme suit :

$$t \preceq_s t' \Leftrightarrow \forall d_i \in \mathscr{D}, t[d_i] \preceq_{d_i} t'[d_i]$$

TABLEAU 5.5. – Espace multidimensionnel de l'entrepôt de données OM3

RowId	IdJ	IdT	IdS
1	$France$	P_1	A_1
2	$France$	P_1	A_{1-1}
3	$France$	P_1	A_{1-2}
4	$France$	P_1	A_2
...	...	...	...
5	$France$	P_{1-1}	A_1
...	...	...	...
6	IDF	P_1	A_1
...	...	...	...
7	es	P_{3-3}	A_{7-12}
8	J_1	P_1	A_1
...	...	...	...
9	J_1	P_1	A_2
...	...	...	...
10	J_3	P_3	A_7
...	...	...	...
11	$France$	P_1	ALL_S
...	...	...	...
12	$PACA$	P_{3-3}	ALL_S
13	$Marseille$	P_1	ALL_S
...	...	...	...
14	J_1	P_1	ALL_S
...	...	...	...
15	$France$	ALL_T	A_1
16	$France$	ALL_T	A_{1-1}
...	...	...	...
17	fr	ALL_T	A_{7-12}
18	J_1	ALL_T	A_1
...	...	...	...
19	ALL_J	P_1	A_1
20	ALL_J	P_1	A_{1-1}
...	...	...	...
21	$France$	ALL_T	ALL_S
22	IDF	ALL_T	ALL_S
23	$Paris$	ALL_T	ALL_S
...	...	...	...
24	ALL_J	P_1	ALL_S
...	...	...	...
25	ALL_J	ALL_T	A_7
...	...	...	...
26	ALL_J	ALL_T	ALL_S
27	$\emptyset$	$\emptyset$	$\emptyset$

5.3.3.3. Fonctions

Définition 5.14 (Fonction $Attribute$ dimensionnelle) **-** Pour un tuple t_x de d_i, la fonction $Attribute_{d_i}$ renvoie l'ensemble des attributs dont la valeur est différente de de $NULL$.

Soit t_x un tuple de d_i :

$$Attribute_{d_i}(t_x) = \{A \in t_x \mid t_x[A] \neq NULL\}$$

Exemple 5.9 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

Si $t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, J_2)$
Alors $Attribute_{\texttt{D}_{\texttt{Joueur}}}(t_x) =$
$\{France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, J_2\}$
Si $t_y = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$
Alors $Attribute_{\texttt{D}_{\texttt{Joueur}}}(t_y) = \{France, PACA, Marseille, 139.124.242.125\}$

5.3.3.4. Opérateurs

Définition 5.15 (Opérateurs min/max dimensionnels) **-** Nous définition les opérateurs min et max comme suit :

$$min_{\preceq_{d_i}}(d_i) = \{x \in Dom(h_i, e) \mid \nexists e' \in S_h(d_i) : e' < e\}$$
$$max_{\preceq_{d_i}}(d_i) = \{x \in Dom(h_i, e) \mid \nexists e' \in S_h(d_i) : e' > e\}$$

Donc, $min_{\preceq_{d_i}}(d_i)$ représente l'ensemble des valeurs possibles du niveau le plus haut (général) de la structure de la dimension hiérarchique. De manière semblable, $max_{\preceq_{d_i}}(d_i)$ représente l'ensemble des valeurs possibles du niveau le plus bas (spécifique) de la structure de la dimension hiérarchique.

Exemple 5.10 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

$$min_{\preceq_{\texttt{D}_{\texttt{Joueur}}}}(\texttt{D}_{\texttt{Joueur}}) = \{France\}$$
$$max_{\preceq_{\texttt{D}_{\texttt{Joueur}}}}(\texttt{D}_{\texttt{Joueur}}) = \{J_1, J_2, J_3\}$$

Les opérateurs min/max peuvent être surchargés afin d'être appliqués à un ensemble de tuples T d'une dimension d_i, notés respectivement $min_{\preceq_s}(T)$ et $max_{\preceq_s}(T)$, en étant implémentés comme suit :

$$min_{\preceq_{d_i}}(T) = \forall x \in Dom(h_i, e), t_x \mid \nexists e' \in S_h(d_i) : e' < e$$
$$max_{\preceq_{d_i}}(T) = \forall x \in Dom(h_i, e), \{t_x \mid \nexists e' \in S_h(d_i) : e' > e\}$$

Exemple 5.11 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$, si $T = \{t_x, t_y\}$ avec :

$$t_x = (France, PACA, Marseille, 139.124.242.125, Linux, Opera, en, J_2)$$
$$t_y = (France, PACA, Marseille, 139.124.242.125, Mac\ OS, Firefox, es, J_3)$$

Alors, nous avons :

$$min_{\preceq_{\texttt{D}_{\texttt{Joueur}}}}(T) = (France, NULL, NULL, NULL, NULL, NULL, NULL, NULL)$$
$$max_{\preceq_{\texttt{D}_{\texttt{Joueur}}}}(T) = \{(NULL, NULL, NULL, NULL, NULL, NULL, NULL, J_2),$$
$$(NULL, NULL, NULL, NULL, NULL, NULL, NULL, J_3)\}$$

Définition 5.16 (Opérateurs min/max généralisé sur le treillis cube) **-** Nous généralisons l'opérateur min sur le treillis cube de la façon suivante :

$$\begin{cases} \forall T \subseteq CL(r), min_{\preceq_s}(T) = \{t \in T \mid \nexists t' \in T : t' \preceq_s t\} \\ \forall d_i \in \mathscr{D}, min_{\preceq_s}(T[d_i]) = t[d_i] \in T[d_i] \mid \nexists t'[d_i] \in T[d_i] : t'[d_i] \preceq_s t[d_i] \end{cases}$$

De même, nous généralisons l'opérateur max sur le treillis cube de la façon suivante :

$$\begin{cases} \forall T \subseteq CL(r), max_{\preceq_s}(T) = \{t \in T \mid \nexists t' \in T : t \preceq_s t'\} \\ \forall d_i \in \mathscr{D}, max_{\preceq_s}(T[d_i]) = \{t[d_i] \in T[d_i] \mid \nexists t'[d_i] \in T[d_i] : t[d_i] \preceq_s t'[d_i]\} \end{cases}$$

Définition 5.17 (Opérateur Somme dimensionnelle) **-** Soit $d_i \in \mathscr{D}$, nous définissons la Somme dimensionnelle $+_{d_i}$ de d_i comme suit : $\forall x, y \in Dom(h_i), x +_{d_i} y$ est le plus proche (petit) ancêtre commun à x et y dans h_i. Autrement dit :

$$\forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g)$$
$$x +_{d_i} y = z \in Dom(h_i, e) \mid z \preceq_{d_i} x \text{ et } z \preceq_{d_i} y \mid \nexists e' \in S_h(d_i) : e' > e$$

Exemple 5.12 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$, si $x = Paris$ et $y = Marseille$ alors $x +_{\texttt{D}_{\texttt{Joueur}}} y = France$.

L'opérateur de Somme dimensionnelle peut être surchargé afin d'être appliqué à tous les tuples t_x et t_y d'une dimension d_i, noté $t_x +_{d_i} t_y$, en étant implémenté comme suit :

$$\forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),$$
$$\exists z \in Dom(h_i, e), e < f \text{ et } e < g, t_z = t_x +_{d_i} t_y = t_{x +_{d_i} y}$$

Exemple 5.13 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

$$\begin{aligned}
&\text{Si } x = Paris,\\
&\quad t_x = (France, IDF, Paris, NULL, NULL, NULL, NULL, NULL)\\
&\text{et } y = Marseille,\\
&\quad t_y = (France, PACA, Marseille, NULL, NULL, NULL, NULL, NULL)\\
&\text{Alors } z = France,\\
&\quad t_z = t_x +_{d_i} t_y\\
&\quad\ = (France, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
\end{aligned}$$

Définition 5.18 (Opérateur Somme généralisé sur le treillis cube) **-** Nous généralisons l'opérateur Somme sur le treillis cube de la façon suivante :

$$\begin{cases} \forall u, v \in CL(r), \{z = u + v\} \\ \forall d_i \in \mathscr{D}, z[d_i] = u[d_i] +_{di} v[d_i] \end{cases}$$

z est la somme des tuples u et v.

Définition 5.19 (Opérateur Produit dimensionnel) **-** Soit $d_i \in \mathscr{D}$, nous définissons le Produit dimensionnel $\bullet_{d_i}$ de d_i comme suit : $\forall x, y \in Dom(h_i), x \bullet_{d_i} y$ est l'ensemble des plus proches descendants communs à x et y dans h_i. Autrement dit :

$$\begin{aligned}
&\forall f, g < Prof(h_i), \forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),\\
&x \bullet_{d_i} y = \{z \in Dom(h_i, e) \mid x \preceq_{d_i} z \text{ et } y \preceq_{d_i} z \mid \nexists e' \in S_h(d_i) : e' < e\}
\end{aligned}$$

Si x et y n'ont pas de descendants communs dans h_i, alors :

$$x \bullet_{d_i} y = \{\emptyset\}$$

Exemple 5.14 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

$$\begin{aligned}
\text{Si } x = Paris \text{ et } y = 92.88.91.80, x \bullet_{\texttt{D}_{\texttt{Joueur}}} y = \{Windows\}\\
\text{Si } x = Marseille \text{ et } y = 139.124.242.125, x \bullet_{\texttt{D}_{\texttt{Joueur}}} y = \{Linux, Mac\ OS\}\\
\text{Si } x = Paris \text{ et } y = Marseille, x \bullet_{\texttt{D}_{\texttt{Joueur}}} y = \{\emptyset\}
\end{aligned}$$

L'opérateur de Produit dimensionnel peut être surchargé afin d'être appliqué à tous tuples t_x et t_y d'une dimension d_i, noté $t_x \bullet_{d_i} t_y$, en étant implémenté comme suit :

$$\begin{aligned}
&\forall f, g < Prof(h_i), \forall x \in Dom(h_i, f), \forall y \in Dom(h_i, g),\\
&t_z = t_x \bullet_{d_i} t_y = \begin{cases} \{t_{x \bullet_{d_i} y}\} \text{ si } \exists z \in Dom(h_i, e), f < e \text{ et } g < e \\ \{(\emptyset, \ldots, \emptyset)\} \text{ sinon.} \end{cases}
\end{aligned}$$

Exemple 5.15 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

Si $x = Marseille$,
$t_x = (France, PACA, Marseille, NULL, NULL, NULL, NULL, NULL)$
Et $y = 139.124.242.125$,
$t_y = (France, PACA, Marseille, 139.124.242.125, NULL, NULL, NULL, NULL)$
Alors $z = \{Linux, Mac\ OS\}$,
$t_z = t_x \bullet_{d_i} t_y$
$= \{(France, PACA, Marseille, 139.124.242.125, Linux, NULL, NULL, NULL)$
$(France, PACA, Marseille, 139.124.242.125, Mac\ OS, NULL, NULL, NULL)\}$

Définition 5.20 (Opérateur Produit généralisé sur le treillis cube) **-** Nous généralisons l'opérateur Produit sur le treillis cube de la façon suivante :

$$\begin{cases} \forall u, v \in CL(r), \{z = u \bullet v\} \\ \forall d_i \in \mathscr{D}, \{z[d_i] = u[d_i] \bullet_{d_i} v[d_i]\} \end{cases}$$

z est le produit des tuples u et v.

Définition 5.21 (Opérateur Semi-produit dimensionnel) **-** Soit $d_i \in \mathscr{D}$, nous définissons le Semi-produit dimensionnel $\odot_{d_i}$ de d_i comme suit : $\forall x, y \in Dom(h_i), x \odot_{d_i} y$ est l'ensemble des plus proches descendants de x et de y dans h_i. Autrement dit :

$$\forall f < Prof(h_i), \forall x, y \in Dom(h_i, f),$$
$$x \odot_{d_i} y = \{z \in Dom(h_i, e) \mid x \preceq_{d_i} z \text{ et } y \preceq_{d_i} z \mid \nexists e' \in S_h(d_i) : e' < e\}$$

Si x et y n'ont pas le même niveau, ou s'ils n'ont pas de descendants, dans h_i, alors :

$$x \odot_{d_i} y = \{\emptyset\}$$

Exemple 5.16 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

$$\text{Si } x = Linux \text{ et } y = Max\ OS, x \odot_{\texttt{D}_{\texttt{Joueur}}} y = \{Opera, Firefox\}$$
$$\text{Si } x = Marseille \text{ et } y = Max\ OS, x \odot_{\texttt{D}_{\texttt{Joueur}}} y = \{\emptyset\}$$

L'opérateur de Semi-produit dimensionnel peut être surchargé afin d'être appliqué à tous tuples t_x et t_y d'une dimension d_i, noté $t_x \odot_{d_i} t_y$, en étant implémenté comme suit :

$$\forall f < Prof(h_i), \forall x, y \in Dom(h_i, f), \forall z \in Dom(h_i, e),$$
$$t_z = t_x \odot_{d_i} t_y = \{t_{x \odot_{d_i} y} \mid e > f \mid \nexists e' \in S_h(d_i) : e' < e\}$$

$$\forall x, y \in Dom(h_i, f),$$
$$t_z = t_x \odot_{d_i} t_y = \begin{cases} \{t_{x \odot_{d_i} y}\} \text{ si } \exists z \in Dom(h_i, e), f < e \\ \{(\emptyset, \ldots, \emptyset)\} \text{ sinon.} \end{cases}$$

Exemple 5.17 - Pour la dimension $\texttt{D}_{\texttt{Joueur}}$:

Si $x = Linux$,
$t_x = (France, PACA, Marseille, 139.124.242.125, Linux, NULL, NULL, NULL)$
Et $y = Mac\ OS$,
$t_y = (France, PACA, Marseille, 139.124.242.125, Mac\ OS, NULL, NULL, NULL)$
Alors $z = \{Opera, Firefox\}$,
$t_z = t_x \odot_{d_i} t_y$
$= \{(France, PACA, Marseille, 139.124.242.125, Linux, Opera, NULL, NULL)$
$(France, PACA, Marseille, 139.124.242.125, Mac\ OS, Firefox, NULL, NULL)\}$

Définition 5.22 (Opérateur Semi-produit généralisé sur le treillis cube) **-** Nous généralisons l'opérateur Semi-produit sur le treillis cube de la façon suivante :

$$\begin{cases} \forall u, v \in CL(r), \{z = u \odot v\} \\ \forall d_i \in \mathscr{D}, \{z[d_i] = u[d_i] \odot_{d_i} v[d_i]\} \end{cases}$$

z est le produit des tuples u et v.

5.3.3.5. Caractérisation du treillis cube hiérarchique

En dotant l'espace multidimensionnel $Space(r)$ de r des ordres de spécialisation et en utilisant les opérateurs, notamment Somme et Produit, nous proposons un structure algébrique appelée treillis cube de données hiérarchique, ou treillis cube hiérarchique, ou encore plus simplement treillis cube, qui fixe un cadre théorique et général pour la fouille de bases de données multidimensionnelles. Il est aisément transposable du treillis cube de données standard. Les différents lemmes et les propositions suivantes donnent les propriétés fondamentales du treillis cube, qui sont reprises dans le théorème 5.6.

Lemme 5.1 - L'ensemble ordonné $CL(r) = \langle Space(r), \preceq_s \rangle$ est un treillis complet appelé treillis cube pour lequel :
— $\forall T \subseteq CL(r), \bigwedge T = +_{t \in T}\ t$ où $\bigwedge$ symbolise *l'infimum.*
— $\forall T \subseteq CL(r), \bigvee T = \bullet_{t \in T}\ t$ où $\bigvee$ symbolise le *supremum.*

Lemme 5.2 - Le treillis $CL(r) = \langle Space(r), \preceq_s \rangle$ est un treillis co-atomique et atomique.

Proposition 5.3 (Treillis hiérarchique des parties des attributs binaires) **-** Soit $\mathscr{L}(r)$ le treillis hiérarchique des parties des attributs binaires de la relation binaire, *i.e.* le treillis $\langle \mathscr{P}(\bigcup_{d_i \in \mathscr{D}} d_i.a, \forall a \in Dom(d_i)), \subseteq \rangle$. Alors il existe un plongement d'ordre Φ :
$CL(r) \rightarrow \mathscr{L}(r)$

$$t \mapsto \begin{cases} \bigcup_{d_i \in \mathscr{D}} d_i.a, \forall a \in Dom(d_i) \text{ si } t = (\emptyset, \ldots, \emptyset) \\ \{d_i.t[d_i] \mid \forall d_i \in Attribute(t)\} \text{ sinon.} \end{cases}$$

Le rang d'un tuple t, noté $rank(t)$, est la longueur du plus petit chemin (nombre d'arcs minimal) dans le treillis cube le reliant au tuple $(ALL_i, \ldots, ALL_j)$. Nous avons donc : $rank(t) = |\Phi(t)|$ si $t \neq (\emptyset, \ldots, \emptyset), |\mathscr{D}| + 1$ sinon.

Proposition 5.4 (Co-atome / atome hiérarchique) **-** Les co-atomes hiérarchiques (respectivement les atomes hiérarchiques) sont les tuples maximaux, c'est à dire les tuples les plus spécifiques (respectivement les tuples minimaux) du treillis privé de son majorant (respectivement minorant) universel. Les co-atomes hiérarchiques (respectivement les atomes hiérarchiques) du treillis cube hiérarchique d'une relation r sont notés $\mathscr{CA}t(CL(r))$ (respectivement $\mathscr{A}t(CL(r))$).

Lemme 5.5 - Le treillis cube $CL(r)$ est gradué. Si $|\mathscr{D}| \leq 2$ alors $CL(r)$ n'est pas distributif.

Le lemme 5.5 montre que le treillis cube est un treillis gradué. En conséquence, nous pouvons appliquer des algorithmes par niveau sur cet espace de recherche.

Théorème 5.6 (Treillis cube de données hiérarchique) **-** Soit r un entrepôt de données composés des dimensions hiérarchiques et des mesures $(\mathscr{D} \cup \mathscr{M})$. L'ensemble ordonné $CL(r) = \langle Space(r), \preceq_s \rangle$ est un treillis hiérarchique complet, atomique, co-atomique et gradué appelé treillis cube de données hiérarchique dans lequel :

— $\forall T \subseteq CL(r), \bigwedge T = +_{t \in T} t$

— $\forall T \subseteq CL(r), \bigvee T = \bullet_{t \in T} t$.

Exemple 5.18 - La figure 5.6 représente le treillis cube de notre relation de la table des faits exemple (cf. tableau 5.1). Dans ce diagramme, les arêtes représentent les liens de généralisation ou spécialisation entre tuples. Les valeurs des attributs codés le sont comme suit :

Localisation		Phase		Association	
$France$	$= F$	$P_x(x \in \mathbb{N}^*)$	$= x$	$A_x(x \in \mathbb{N}^*)$	$= x$
IDF	$= I$	$P_{x-y}(x, y \in \mathbb{N}^*)$	$= x - y$	$A_{x-y}(x, y \in \mathbb{N}^*)$	$= x - y$
$PACA$	$= P$	$ALL_{\texttt{Tour}}$	$= *$	$ALL_{\texttt{Série}}$	$= *$
$Marseille$	$= M$				
$ALL_{\texttt{Joueur}}$	$= *$				

En faisant abstraction de $(\emptyset, \emptyset, \emptyset)$, les co-atomes hiérarchiques sont des tuples véhiculant l'information au niveau le plus détaillé, *i.e.* celui des valeurs réelles des dimensions. En d'autres termes, les co-atomes hiérarchiques sont les tuples potentiels

d'une relation. Ainsi, nous avons :

$$\mathscr{C}\mathscr{A}t(CL(\texttt{OM3})) = \left\{ \begin{array}{l} (France, P_1, A_1), \\ (France, P_1, A_{1-1}), \\ (France, P_1, A_{1-2}), \\ (France, P_1, A_2), \\ \ldots \\ (France, P_{1-1}, A_1), \\ \ldots \\ (IDF, P_1, A_1), \\ \ldots \\ (es, P_{3-3}, A_{7-12}), \\ (J_1, P_1, A_1), \\ \ldots \\ (J_1, P_1, A_2), \\ \ldots \\ (J_2, P_3, A_7), \\ \ldots \end{array} \right\}$$

Les atomes hiérarchiques du treillis offrent l'information la plus synthétique possible, à l'exception du tuple $(ALL_{\texttt{Joueur}}, ALL_{\texttt{Tour}}, ALL_{\texttt{Série}})$. Étant donné que nous ne considérons ici que trois dimensions, tous les atomes hiérarchiques comportent deux valeurs synthétiques ALL_i différentes. Nous avons donc :

$$\mathscr{A}t(CL(r)) = \left\{ \begin{array}{l} (France, ALL_{\texttt{Tour}}, ALL_{\texttt{Série}}), \\ (IDF, ALL_{\texttt{Tour}}, ALL_{\texttt{Série}}), \\ (Paris, ALL_{\texttt{Tour}}, ALL_{\texttt{Série}}), \\ \ldots \\ (ALL_{\texttt{Joueur}}, P_1, ALL_{\texttt{Série}}), \\ \ldots \\ (ALL_{\texttt{Joueur}}, ALL_{\texttt{Tour}}, A_7), \\ \ldots \end{array} \right\}$$

Propriété 5.1 (Taille d'un treillis cube hiérarchique) **-** La hauteur (nombre de niveaux) du treillis cube hiérarchique est $|\mathscr{D}|+1$. Le nombre d'éléments pour un niveau $i (i \in 1..|\mathscr{D}|)$ est :

$$\sum_{\substack{d_i \subseteq \mathscr{D} \\ |d_i|=i}} \left(\prod_{A \in d_i} |Dom(h_i)| \right) \leq \binom{|\mathscr{D}|}{i} \max_{A \in \mathscr{D}} (|Dom(h_i)|)^i.$$

Le nombre total d'éléments dans le treillis cube hiérarchique est :

$$\sum_{i=1..|\mathscr{D}|} \left(\sum_{\substack{d_i \subseteq \mathscr{D} \\ |d_i|=i}} \left(\prod_{A \in d_i} |Dom(h_i)| \right) \right) + 2 = \left(\prod_{A \in \mathscr{D}} (|Dom(h_i))| + 1) \right) + 1$$

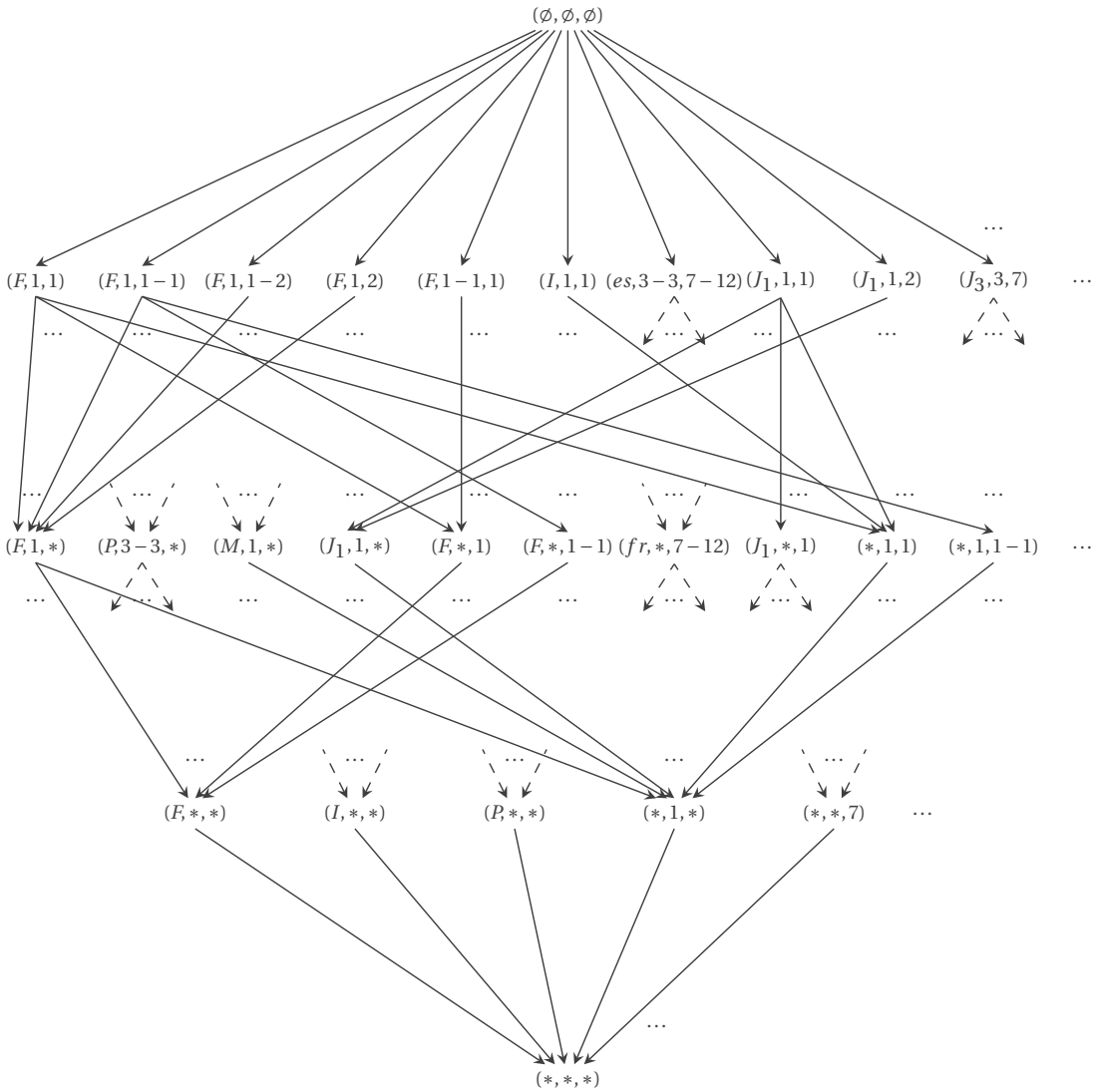


FIGURE 5.6. – Diagramme de Hasse du treillis cube hiérarchique de OM3

La propriété précédente donne une analyse du nombre d'éléments contenus dans un niveau du treillis cube hiérarchique ainsi que le nombre total d'éléments. Cette propriété est particulièrement importante car elle permet de caractériser la complexité des algorithmes utilisant le treillis cube hiérarchique comme espace de recherche.

5.4. Cube de données hiérarchique

Le concept de cube de données en monde relationnel (GRAY, CHAUDHURI, BOSWORTH et al. 1997) est immédiatement transposable en cube de données hiérarchique en monde entrepôt de données, avec l'intégration de dimensions hiérarchiques (cf. cf. 2.2.4 p. 34).

Ainsi, nous appelons le résultat d'une requête de partitionnement, selon un ensemble $X \subseteq \mathscr{D}$ de dimensions hiérarchiques, cuboïde, noté $cuboide_X(r)$. De même, le cube de données constitué de l'ensemble de tous les cuboïdes est noté $datacube(r)$. Les cuboïdes peuvent être ordonnés par rapport à leur niveau de détail par la relation d'ordre partiel $\preceq$.

Exemple 5.19 - Le cuboïde selon les dimensions hiérarchiques $\mathtt{D}_{\mathtt{Joueur}},\mathtt{D}_{\mathtt{Tour}}$ est moins agrégé (plus détaillé) que le cuboïde selon $\mathtt{D}_{\mathtt{Joueur}}$ ou celui selon $\mathtt{D}_{\mathtt{Tour}}$. En d'autres termes :

$$Cuboide_{\mathtt{D}_{\mathtt{Joueur}}}(r) \preceq Cuboide_{\mathtt{D}_{\mathtt{Joueur}},\mathtt{D}_{\mathtt{Tour}}}(r)$$
$$Cuboide_{\mathtt{D}_{\mathtt{Tour}}}(r) \preceq Cuboide_{\mathtt{D}_{\mathtt{Joueur}},\mathtt{D}_{\mathtt{Tour}}}(r)$$

De manière standard, l'ensemble des cuboïdes muni de cet ordre forme un treillis. Les cuboïdes de ce treillis sont regroupés par niveau en fonction de leur nombre de dimensions hiérarchiques. Ces niveaux sont numérotés en partant du bas du treillis (cuboïde ne portant sur aucune dimension d_i) et en remontant vers le sommet (cuboïde suivant tous les critères possibles appelé « cuboïde de base »). Soit deux cuboïdes $cuboide_{\mathscr{U}}(r)$ selon le sous-ensemble de dimensions hiérarchiques $\mathscr{U}$ et $cuboide_{\mathscr{V}}(r)$ selon $\mathscr{V}$, si $\mathscr{U} \subset \mathscr{V}$ alors $cuboide_{\mathscr{U}}(r) \preceq cuboide_{\mathscr{V}}(r)$, on dit que les cuboides ont un lien de parenté, $cuboide_{\mathscr{V}}(r)$ est nommé « l'ancêtre » du $cuboide_{\mathscr{U}}(r)$ et $cuboide_{\mathscr{U}}(r)$ « le descendant » de $cuboide_{\mathscr{V}}(r)$.

La figure 5.7 donne la représentation du treillis des cuboïdes de l'entrepôt de données `OM3` suivant les dimensions hiérarchiques $\mathtt{D}_{\mathtt{Joueur}}$ (`IdJ`), $\mathtt{D}_{\mathtt{Tour}}$ (`IdT`) et $\mathtt{D}_{\mathtt{Série}}$ (`IdS`).

En reprenant l'exemple de l'entrepôt de données `OM3`, et sa table des faits (cf. tableau 5.1), nous pouvons exprimer en SQL le cube de données hiérarchique nécessaire à une analyse selon les mesures `Temps` et `Score` par rapport aux dimensions hiérarchiques `IdJ`, `IdT` et `IdS` avec la requête suivante :

```
01 | SELECT IdJ, IdT, IdS, SUM(Temps), MAX(Score)
02 | FROM OM3
03 | GROUP BY CUBE IdJ, IdT, IdS
```

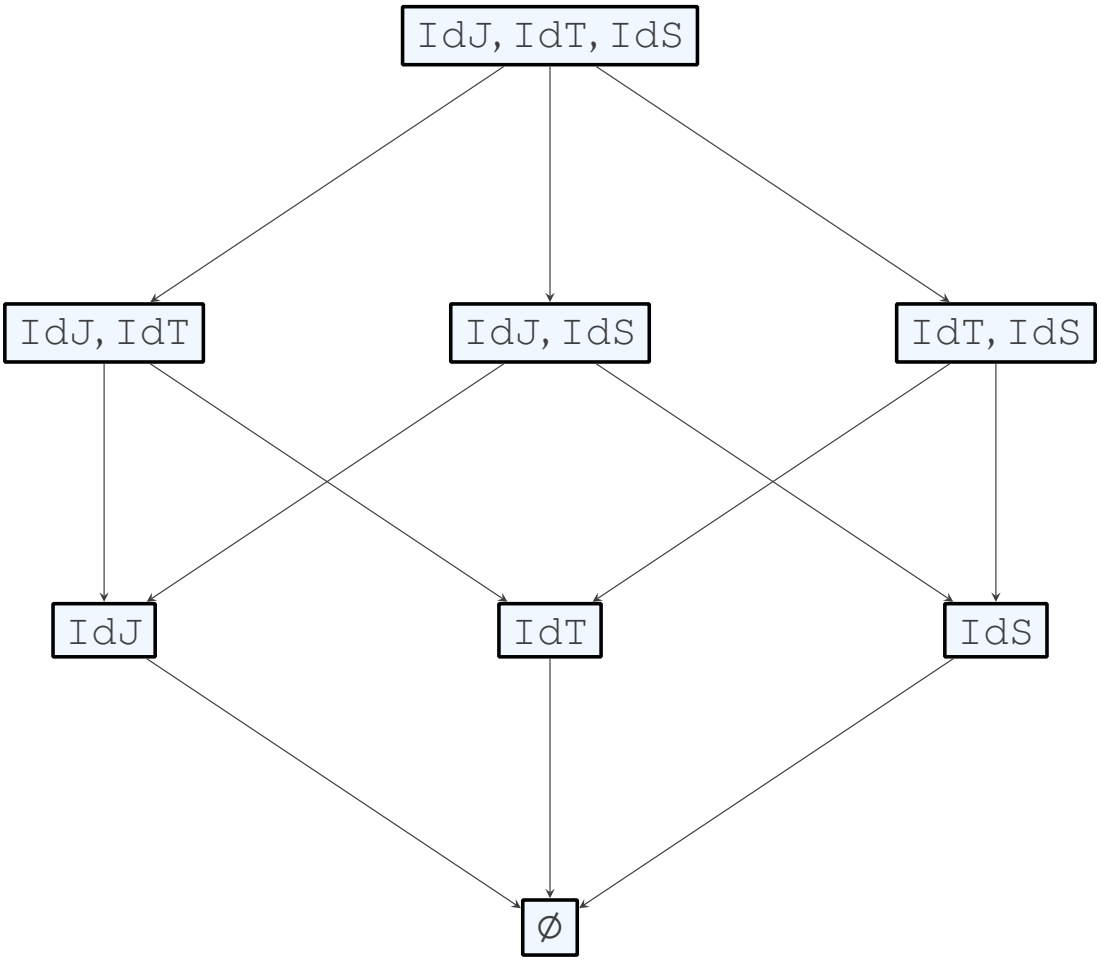


FIGURE 5.7. – Cuboïdes de OM3 suivant D_{Joueur} (IdJ), D_{Tour} (IdT) et $D_{Série}$ (IdS)

Cette requête va calculer les $2^3 = 8$ partitionnements (où $\emptyset$ représente le partitionnement selon aucune dimension d_i, *i.e.* agréger toute l'entrepôt de données en un seul tuple) :

— 'IdJ,IdT,IdS';
— 'IdJ,IdT';
— 'IdJ,IdS';
— 'IdT,IdS';
— 'IdJ';
— 'IdT';
— 'IdS';
— '$\emptyset$'.

De même que pour le cube de donné traditionnel, la manière naïve de calculer ce type de requête est de la réécrire comme une collection de huit requêtes agrégatives et de les exécuter séparément. Cependant, chacune de ses sous-requêtes a un schéma qui lui est propre. Pour rendre tous ces cuboïdes uni-compatibles, nous utilisons la valeur ALL_i (cf. définition 5.5) : cette valeur a une sémantique particulière, c'est une généralisation de toutes les valeurs du domaine d'un attribut d'une dimension d_i. Ainsi, les cuboïdes partagent tous le même schéma ($\mathscr{D} \cup \mathscr{M}$) et peuvent être regroupés dans une seule et même relation : le cube de données hiérarchique. Chaque tuple multidimensionnel est constitué d'un ensemble de valeurs pour ses dimensions hiérarchiques et de valeurs numériques pour leurs mesures. La valeur de la mesure est calculée en agrégeant l'ensemble des tuples de la relation originelle partageant les mêmes valeurs des dimensions hiérarchiques sélectionnées.

Exemple 5.20 - Le tuple multidimensionnel $t = (8, 1, ALL)$ est obtenu en agrégeant par `IdJ` et `IdT` les tuples $t_1 = (8, 1, 1)$ et $t_2 = (8, 1, 4)$ de l'entrepôt de données exemple (cf. tableau 5.1). Les mesures de t sont obtenues en appliquant les fonctions mesure sur l'ensemble des tuples nécessaires à l'agrégation.

Dans notre cas, nous avons :

— $f_{\texttt{Temps}}(t, r) = t_1(\texttt{Temps}) + t_2(\texttt{Temps}) = SUM(6.32, 18.9) = 25.22$
— ...
— $f_{\texttt{Score}}(t, r) = t_1(\texttt{Score}) + t_2(\texttt{Score}) = MAX(700, 2300) = 2300$
— ...

Avec l'entrepôt de données exemple (cf. tableau 5.1), la requête pour calculer naïvement le cube de données hiérarchique suivant les dimensions hiérarchiques $\texttt{D}_{\texttt{Joueur}}$ (`IdJ`), $\texttt{D}_{\texttt{Tour}}$ (`IdT`) et $\texttt{D}_{\texttt{Série}}$ (`IdS`) et les mesures `Temps`, `Durée`, `Nombre`, `Score` et `Forme` est la suivante :

```
01 | SELECT IdJ, IdT, IdS, SUM(Temps), MAX(Score)
02 | FROM OM3
03 | GROUP BY IdJ, IdT, IdS
04 | UNION
05 | SELECT IdJ, IdT, ALL, SUM(Temps), MAX(Score)
06 | FROM OM3
07 | GROUP BY IdJ, IdT
```

```
08 | UNION
09 | SELECT IdJ, ALL, IdS, SUM(Temps), MAX(Score)
10 | FROM OM3
11 | GROUP BY IdJ, IdS
12 | UNION
13 | SELECT ALL, IdT, IdS, SUM(Temps), MAX(Score)
14 | FROM OM3
15 | GROUP BY IdT, IdS
16 | UNION
17 | SELECT IdJ, ALL, ALL, SUM(Temps), MAX(Score)
18 | FROM OM3
19 | GROUP BY IdJ
20 | UNION
21 | SELECT ALL, IdT, ALL, SUM(Temps), MAX(Score)
22 | FROM OM3
23 | GROUP BY IdT
24 | UNION
25 | SELECT ALL, ALL, IdS, SUM(Temps), MAX(Score)
26 | FROM OM3
27 | GROUP BY IdS
28 | UNION
29 | SELECT ALL, ALL, ALL, SUM(Temps), MAX(Score)
30 | FROM OM3
```

Le résultat de cette requête est donné par le tableau 5.6. Par souci de clarté, les différents cuboïdes sont séparés par une ligne horizontale, et les valeurs spéciales sont abrégées : $ALL_{\texttt{Joueur}}$ en $ALL_{\texttt{J}}$, $ALL_{\texttt{Tour}}$ en $ALL_{\texttt{T}}$ et $ALL_{\texttt{Série}}$ en $ALL_{\texttt{S}}$.

5.5. Cube de données hiérarchique fermé

L'explosion combinatoire des résultats lors des calculs des cubes de données est un phénomène bien connu (HAN, KAMBER et PEI 2011), qui est amplifié avec le calcul des cubes de données hiérarchiques. L'approche du cube fermé hiérarchique que nous proposons a pour objectif la représentation du cube de données hiérarchique sans perte d'information mais avec une diminution conséquente de l'espace de stockage nécessaire. Pour ce faire, nous éliminons les redondances éventuelles en ne conservant qu'un représentant pour un ensemble de tuples issus des mêmes données de la relation d'origine. Il est aisément transposable du cube de données fermé standard. Les différentes définitions et le corollaire suivants donnent les propriétés fondamentales du cube de données hiérarchique fermé, qui sont reprises dans le théorème 5.8.

Définition 5.23 (Opérateur de fermeture cubique hiérarchique) - L'opérateur de fermeture cubique hiérarchique associe à tout tuple t du treillis cube hiérarchique un unique tuple appelé la fermeture hiérarchique de t, ou simplement fermeture de t. Il est obtenu en considérant l'ensemble des tuples plus spécifiques que t et en déterminant le tuple le plus général de cet ensemble grâce à l'opérateur Somme. Cet

TABLEAU 5.6. – Représentation en relation du cube de données hiérarchique de OM3

IdJ	IdT	IdS	Temps	Durée	Nombre	Score	Forme
J_1	P_1	A_1	6.32	2.85	3.5	700	0.5
J_1	P_1	A_2	18.9	1.95	3.83	2300	0.5
J_2	P_2	A_3	26.39	1.7	3.43	2400	0.71
J_2	P_2	A_4	4.1	2.07	3	300	0.5
J_2	P_2	A_5	7.38	3.68	3	600	0.5
J_3	P_3	A_6	2.14	2.15	3	300	1
J_3	P_3	A_7	56.04	2.25	3.17	3800	0.5
J_1	P_1	ALL_S	25.22	2.04	3.8	2300	0.5
J_2	P_2	ALL_S	37.87	1.79	3.4	2400	0.7
J_3	P_3	ALL_S	58.18	2.25	3.17	3800	0.5
J_1	ALL_T	A_1	6.32	2.85	3.5	700	0.5
J_1	ALL_T	A_2	18.9	1.95	3.83	2300	0.5
J_2	ALL_T	A_3	26.39	1.7	3.43	2400	0.71
J_2	ALL_T	A_4	4.1	2.07	3	300	0.5
J_2	ALL_T	A_5	7.38	3.68	3	600	0.5
J_3	ALL_T	A_6	2.14	2.15	3	300	1
J_3	ALL_T	A_7	56.04	2.25	3.17	3800	0.5
ALL_J	P_1	A_1	6.32	2.85	3.5	700	0.5
ALL_J	P_1	A_2	18.9	1.95	3.83	2300	0.5
ALL_J	P_2	A_3	26.39	1.7	3.43	2400	0.71
ALL_J	P_2	A_4	4.1	2.07	3	300	0.5
ALL_J	P_2	A_5	7.38	3.68	3	600	0.5
ALL_J	P_3	A_6	2.14	2.15	3	300	1
ALL_J	P_3	A_7	56.04	2.25	3.17	3800	0.5
J_1	ALL_T	ALL_S	25.22	2.04	3.8	2300	0.5
J_2	ALL_T	ALL_S	37.87	1.79	3.4	2400	0.7
J_3	ALL_T	ALL_S	58.18	2.25	3.17	3800	0.5
ALL_J	P_1	ALL_S	25.22	2.04	3.8	2300	0.5
ALL_J	P_2	ALL_S	37.87	1.79	3.4	2400	0.7
ALL_J	P_3	ALL_S	58.18	2.25	3.17	3800	0.5
ALL_J	ALL_T	A_1	6.32	2.85	3.5	700	0.5
ALL_J	ALL_T	A_2	18.9	1.95	3.83	2300	0.5
ALL_J	ALL_T	A_3	26.39	1.7	3.43	2400	0.71
ALL_J	ALL_T	A_4	4.1	2.07	3	300	0.5
ALL_J	ALL_T	A_5	7.38	3.68	3	600	0.5
ALL_J	ALL_T	A_6	2.14	2.15	3	300	1
ALL_J	ALL_T	A_7	56.04	2.25	3.17	3800	0.5
ALL_J	ALL_T	ALL_S	121.27	2.11	3.32	3800	0.55

opérateur est noté $\mathbb{C}$ et défini comme suit :

$$\begin{array}{rcl} \mathbb{C} & : & CL(r) \rightarrow CL(r) \\ t & \mapsto & \begin{cases} +t' \mid t \preceq_s t' \text{ et } t' \in r \\ (\emptyset, \ldots, \emptyset) \text{ sinon.} \end{cases} \end{array}$$

Exemple 5.21 - En considérant l'espace multidimensionnel de l'entrepôt de données OM3 (cf. tableau 5.5) et sa table des faits (cf. tableau 5.1), nous avons $\mathbb{C}((J_1, ALL_T, ALL_S)) = (J_1, P_1, A_1) + (J_1, P_1, A_2) = (J_1, P_1, ALL_S)$ et $\mathbb{C}((ALL_J, ALL_T, A_7)) = (J_3, P_3, A_7)$.

Corollaire 5.7 - $\mathbb{C}$ est un opérateur de fermeture de $CL(r)$ sur r. $\mathbb{C}$ satisfait les propriétés suivantes :

— $t \preceq_g t' \Rightarrow \mathbb{C}(t, r) \preceq_g \mathbb{C}(t', r)$ (isotonie) ;
— $t \preceq_g \mathbb{C}(t, r)$ (extensivité) ;
— $\mathbb{C}(t, r) = \mathbb{C}(\mathbb{C}(t, r), r)$ (idempotence).

Définition 5.24 (Système de fermeture cubique hiérarchique) **-** Soit $\mathbb{C}(r) = \{ t \in CL(r) \mid \mathbb{C}(t, r) = t \}$. $\mathbb{C}(r)$ est un système de fermeture sur r et l'opérateur de fermeture associé est $\mathbb{C}$. Tout tuple appartenant à $\mathbb{C}(r)$ est un tuple fermé hiérarchique ou un fermé cubique hiérarchique.

Exemple 5.22 - En considérant l'espace multidimensionnel de l'entrepôt de données OM3 (cf. tableau 5.5) et sa table des faits (cf. tableau 5.1, nous avons :

$$\mathbb{C}(r) = \left\{ \begin{array}{l} (J_1, P_1, A_1), \\ (J_1, P_1, A_2), \\ (J_2, P_2, A_3), \\ (J_2, P_2, A_4), \\ (J_2, P_2, A_5), \\ (J_3, P_3, A_6), \\ (J_3, P_3, A_7), \\ (J_1, P_1, ALL_S), \\ (J_2, P_2, ALL_S), \\ (J_3, P_3, ALL_S), \\ (\emptyset, \ldots, \emptyset) \end{array} \right\}$$

Théorème 5.8 - L'ensemble partiellement ordonné $CCL(r) = \langle \mathbb{C}(r), \preceq_g \rangle$ est un treillis complet et co-atomique appelé treillis cube fermé tel que :

— $\forall T \subseteq CCL(r), \bigwedge T = +_{t \in T} t$
— $\forall T \subseteq CCL(r), \bigvee T = \mathbb{C}(\bullet_{t \in T} t, r)$

Tous les tuples ayant la même fermeture généralisent les mêmes tuples de la relation originelle. Comme ils généralisent les mêmes tuples originaux, ils partagent la même valeur agrégée de la mesure (propriété du `GROUP BY`). Pour tout tuple du treillis cube, il suffit de calculer sa fermeture pour retrouver sa mesure. Ainsi le cube fermé hiérarchique est une couverture du cube de données hiérarchique.

Exemple 5.23 - La figure 5.8 représente le treillis cube hiérarchique fermé de OM3.

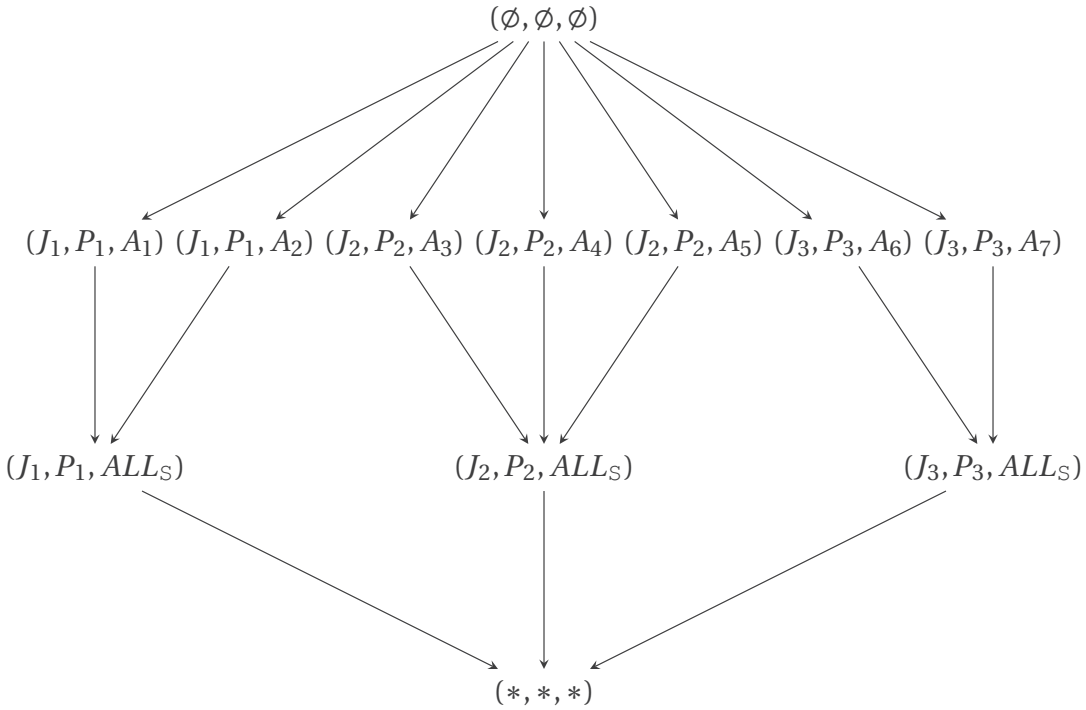


FIGURE 5.8. – Diagramme de Hasse du treillis cube hiérarchique fermé de OM3

5.6. Conclusion

Dans le cadre des entrepôts de données multidimensionnels (*datawarehouse* ou *datamart*), après avoir présenté un cas d'application sur un jeu vidéo, avoir donné une définition aux dimensions hiérarchiques et avoir présenté les différents types de hiérarchies (INMON 1996), nous avons défini un cadre théorique et une définition formelle aux attributs dimensions hiérarchiques ainsi qu'à leur structure et hiérarchie associées.

En s'inspirant de la théorie classique, et en la transposant de manière relativement directe, nous avons défini, dans un contexte hiérarchique, l'espace multidimensionnel, des ordres de spécialisation (MITCHELL 1982 ; MITCHELL 1997), des fonctions et des opérateurs, en particulier l'opérateur Somme et l'opérateur Produit, et chacun, définis au niveau dimensionnel, surchargé et généralisé sur le treillis cube.

Nous avons ensuite caractérisé le treillis cube de données hiérarchique et donné un calcul de sa taille, montrant la complexité des algorithmes utilisant le treillis cube hiérarchique comme espace de recherche.

Nous avons ensuite proposé le cube de données hiérarchique, reprenant l'intérêt du cube de données classique (GRAY, CHAUDHURI, BOSWORTH et al. 1997), concept majeur pour la gestion des entrepôt de données.

Comme pour le cube de données classique, le cube de données hiérarchique fermé est sa représentation la plus concise. Il exploite les redondances généralement largement incluses dans un cube de données hiérarchique. En les supprimant, il permet de réduire conséquemment la taille nécessaire sans perdre de données utiles. Lorsque les données sont très corrélées, la représentation par cube hiérarchique fermé montre tout son intérêt car la réduction apportée est tout à fait significative, et permet pourtant de retrouver encore plus d'information qu'avec un cube de données classique. Ces fortes corrélations apparaissent dans les jeux de données réelles et s'accroissent lorsque les données des dimensions sont très éparpillées sur des domaines très vastes. De telles caractéristiques sont typiques des entrepôts de données (ROSS et SRIVASTAVA 1997 ; BEYER et RAMAKRISHNAN 1999), et en particulier dans le domaine des jeux vidéo où les attributs dimensions hiérarchiques sont bien souvent légion.

6. Skycube émergent

Sommaire

6.1. Introduction et motivations . . . 174
6.2. Cas d'utilisation . . . 175
6.2.1. Pokémon Showdown! . . . 175
6.2.1.1. Système de combat de Pokémon Showdown! . . . 175
6.2.1.2. Ladder de Pokémon Showdown! . . . 178
6.2.1.3. Pokédex . . . 178
6.2.1.4. Individual value (IV) . . . 178
6.2.1.5. Tier stratégique . . . 178
6.2.2. Kaggle . . . 181
6.2.3. Classification atomique de Pokémon Showdown! . . . 181
6.2.4. Relation exemple . . . 182
6.3. Treillis des Concepts Accords et Skylines . . . 188
6.3.1. Treillis des partitions . . . 188
6.3.2. Ensembles en accord . . . 190
6.3.3. Concepts Accords d'une relation . . . 191
6.3.4. Treillis des Concepts Skylines et Skycube . . . 195
6.3.4.1. Treillis des concepts Skylines . . . 195
6.3.4.2. Matérialisation partielle du Skycube . . . 198
6.3.5. Évaluations expérimentales . . . 201
6.3.5.1. Réductions de la taille . . . 201
6.3.5.2. Temps de réponse des requêtes . . . 201
6.3.6. Comparaison avec les travaux antérieurs . . . 204
6.4. Calcul du Skycube émergent . . . 205
6.5. Skycube fermé émergent . . . 209
6.6. Conclusion . . . 213

6.1. Introduction et motivations

Dans ce chapitre, nous introduisons le Skycube émergent et le Skycube fermé émergent. L'optique est de combiner l'approche multicritère permettant une analyse de préférences avec l'approche OLAP dédiée à la découverte de renversement de tendances. Le Skycube émergent permet d'extraire des données optimales selon plusieurs caractéristiques et selon leur évolution significative par rapport à une propriété.

A l'origine, nous souhaitions réutiliser le concept de cube de données émergent (cf. section 3 p. 86) en combinant notre formalisation de l'algorithme de cube de données fermé émergent C-IDEA dans MARTIN NEVOT, NEDJAR, LAKHAL et CICCHETTI 2019 (cf. section 3.5.2 p. 116) au concept de Skycube (cf. section 2.4.3 p. 81). Nous avons conservé notre idée initiale mais nous l'avons étendue à l'usage de l'ensemble de la plateforme algorithmique IDEA tout en intégrant une représentation réduite du Skycube basée sur le treillis des Concepts Skylines.

Après avoir introduit le cas d'utilisation que nous avons expérimenté lors de la recherche de cette contribution, nous présentons les treillis des partitions, puis des Concepts Accords, et enfin Skylines afin de proposer une matérialisation partielle sans perte d'information du Skycube. Puis, nous transposerons le calcul du cube de données émergent, de sa bordure L et du cube de données fermé émergent respectivement au Skycube émergent, à sa bordure L et au Skycube fermé émergent.

6.2. Cas d'utilisation

6.2.1. Pokémon Showdown!

Pour ce chapitre, nous introduisons un nouvel exemple appliqué à un jeu vidéo, celui que nous avons étudié et expérimenté initialement lors de la recherche de cette contribution.

Pokémon Showdown! est un jeu vidéo sur navigateur Web et PC dans une version téléchargeable *open source*. C'est un simulateur de combat de Pokémon [1] populaire (des millions d'utilisateurs mensuellement, avec jusqu'à plus de 20000 internautes en ligne simultanément) permettant de jouer à des combats de Pokémon en ligne entièrement animés.

Il est utilisé par la Smogon University [2] depuis 2012, fut l'un des premiers simulateurs à implémenter les changements apportés par Pokémon Versions Noire 2 et Blanche 2 et inclut des formats spéciaux qui proposent des approches originales des combats Pokémon modifiant les mécaniques du jeu vidéo.

6.2.1.1. Système de combat de Pokémon Showdown!

Pokémon Showdown! propose à ses utilisateurs un système de rencontres aléatoires classées dans un format et une génération de Pokémon souhaités. Le site Web met à disposition, en outre, un tchat [3] lors des combats.

1. Pokémon, possiblement car il s'agit d'un terme issu du japonais, qui ne prend pas de marque du pluriel sur les substantifs, est, de manière conventionnelle, en principe écrit sans marque différenciante au pluriel.
2. Smogon University est un site Web, et un forum d'utilisateurs, de référence à propos des combats Pokémon compétitifs en proposant des guides stratégiques au sujet des combats.
3. Aussi appelé clavardage, un tchat est une conversation écrite virtuelle et en temps réel, par écran interposé, par Internet. Par extension, un site Web proposant ce mode de communication est aussi appelé tchat.

FIGURE 6.1. – Jeu vidéo Pokémon Showdown ! : un exemple de combat [a]

a. Exemple de combat dans le jeu vidéo Pokémon Showdown !

Il est possible de s'affronter avec des équipes générées aléatoirement, ou en créant des équipes personnelles qui sont validés automatiquement en fonction du format recherché.

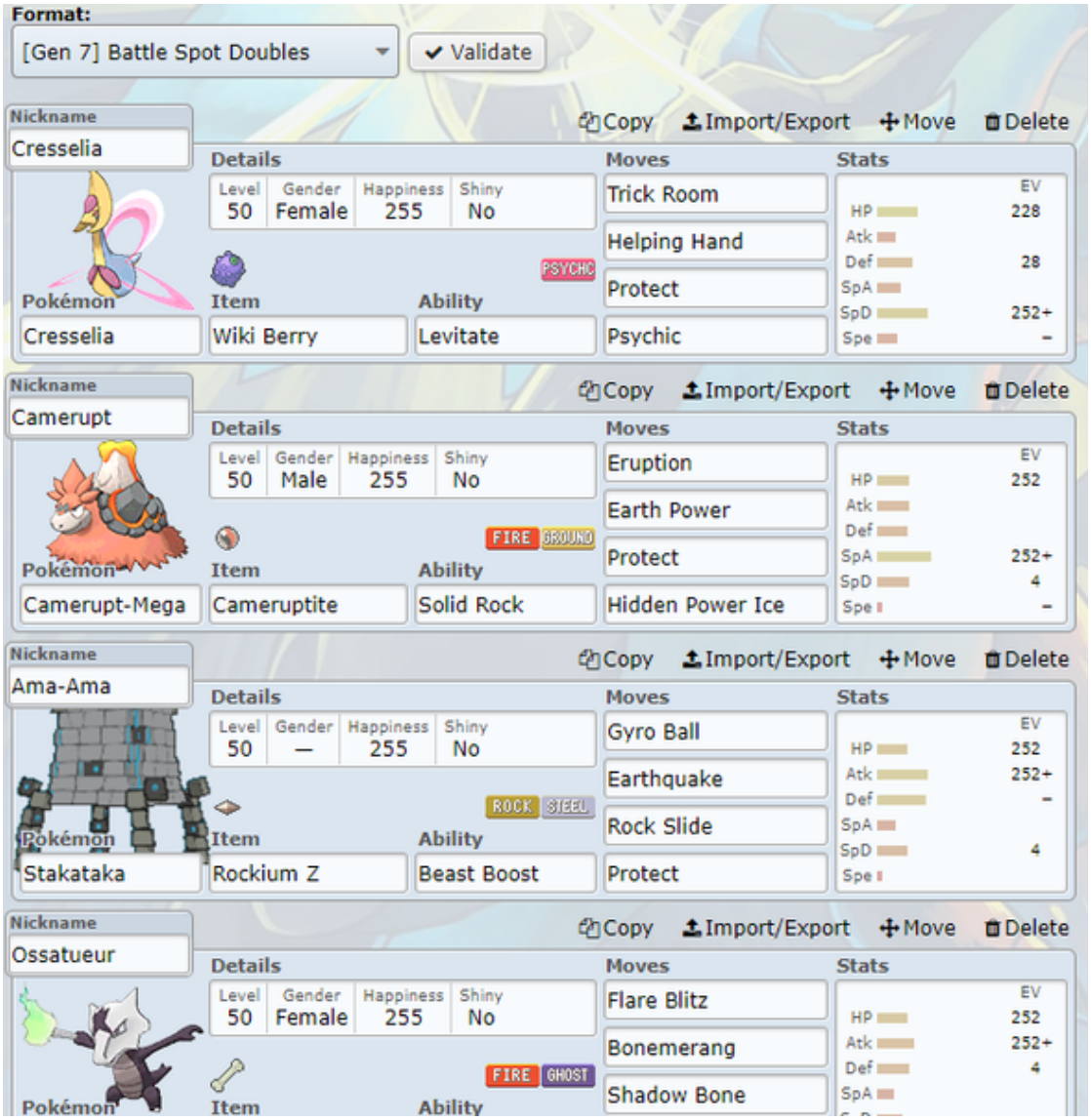


FIGURE 6.2. – Jeu vidéo Pokémon Showdown! : un exemple d'équipe [a]

a. Exemple d'équipe dans le jeu vidéo Pokémon Showdown!

Il est possible de revoir ses combats ou de voir les combats des autres joueurs depuis une interface permettant de consulter les combats récents ou de faire une recherche avancée par critères.

6.2.1.2. Ladder de Pokémon Showdown!

Pokémon Showdown! possède un système de rang basé sur des points pour chacun des *tiers* stratégiques (cf. sous-section 6.2.1.5), appelé *ladder*, ou *leaderboard*. Le classement des joueurs est effectué grâce aux systèmes Elo (cf. sous-sous-section 8.5.2.5 p. 255), GXE (pour *glicko x-Act estimate*) et Clicko-1. Le tableau de classement des meilleurs joueurs pour chaque *ladder* est disponible directement depuis Pokémon Showdown!.

6.2.1.3. Pokédex

Sorte de véritable encyclopédie des Pokémon, un Pokédex est un outil de recherche des caractéristiques des Pokémon.

Pokémon Showdown! propose son propre Pokédex.

6.2.1.4. Individual value (IV)

Un IV, ou *individual value*, est une valeur numérique entière positive définie pour chaque caractéristique d'un Pokémon. La plupart du temps, ces données ne sont pas explicitement communiquées dans les jeux Pokémon sous ce vocable, c'est plutôt « potentiel » qui est cité à la place d'IV, IV étant un terme en usage par la communauté. Plus l'IV d'une caractéristique est important, plus la caractéristique correspondante l'est également.

Soit C, une caractéristique d'un Pokémon, N la valeur de la nature de ce Pokémon (1.1, 0.9 ou 1), L le niveau du Pokémon, B la valeur de base de la caractéristique et EV [4] les points d'effort du Pokémon :

— voici la formule du calcul des IV pour les caractéristiques autres que HP [5] :

$$IV = (\frac{C}{N} - 5) \times \frac{100}{L} - \frac{EV}{4} - 2 \times B$$

— voici la formule de calcul de l'IV de la caractéristique HP :

$$IV = \frac{100 \times (C - L - 10)}{L} - \frac{EV}{4} - 2 \times B$$

6.2.1.5. Tier stratégique

Un *tier* [6] stratégique est une catégorisation non officielle des jeux vidéo, basée sur la dominance stratégique (cf. sous-sous-section 4.6.2.4 p. 140), en fonction des taux

4. Un EV, ou *effort value*, ou encore *effort point*, ou point d'effort en français, est un « point » qu'un Pokémon peut obtenir, et qui permet l'augmentation « bonus » d'une de ses caractéristiques.

5. Dans Pokémon Showdown!, les autres jeux Pokémon et la plupart des jeux, HP (pour *hit points* ou PV en français, pour points de vie) est utilisé comme un indicateur de la santé ou la « vie » d'un personnage ou encore la robustesse d'un objet.

6. En anglais, *tier* peut être traduit par gradin, étage ou étager.

FIGURE 6.3. – Jeu vidéo Pokémon Showdown! : utilisation du Pokédex [a]

a. Utilisation du Pokédex dans le jeu vidéo Pokémon Showdown!

de performance ou de victoire des personnages, des équipes de personnages ou des stratégies mises en œuvre.

Bien qu'officieux et issus d'une lacune d'équilibrage d'un jeu vidéo (cf. sous-sous-section 4.6 p. 137), les *tiers* stratégiques sont fondamentaux pour les joueurs et visent, paradoxalement, à améliorer les déséquilibres en regroupant les personnages, les équipes de personnages ou les stratégies considérées comme « équitables » par la communauté de joueur ou les équipes de développement d'un jeu vidéo.

Tier stratégique de Pokémon Showdown! Dans Pokémon Showdown!, un *tier* stratégique représente donc une classification hiérarchique destinée à trier les Pokémon selon leur efficacité, soit le plus souvent, leur IV (cf. sous-section 6.2.1.4).

En plus des *tiers* stratégiques spécifiques à Pokémon Showdown!, la The Pokémon Company, entreprise gérant la licence Pokémon dans son intégralité, a établi son propre *tier* stratégique, le *video game championship* (VGC), qui est utilisé pour les tournois qualificatifs en vue des Championnats du monde Pokémon annuels et d'éventuels autres tournois *e-sport* (cf. sous-section 8.5.2 p. 246).

De même, le *tier* stratégique Limbo est souvent utilisé. Il s'agit d'un *tier* stratégique de transition, mis en place lors de l'arrivée d'une nouvelle génération de jeux vidéo, avec de nouveaux Pokémon qu'il est nécessaire de jouer longuement afin de pouvoir les catégoriser empiriquement. Ce *tier* stratégique n'a pas pour vocation d'être utilisé comme format en compétition.

Voici les principaux *tiers* stratégiques de Pokémon Showdown! :

— *Anything goes* : *tier* stratégique où « tout est permis », permettant tous les Pokémon ;
— *Uber* (U) : *tier* stratégique constitué de Pokémon dont la puissance ou les capacités constituent un déséquilibre de jeu jugés trop important, les bannissant le plus souvent des compétitions organisées ;
— *Over Used* (OU) : *tier* stratégique signifiant « sur-utilisé » en français, réunissant, comme son nom l'indique, les Pokémon les plus couramment employés dans les compétitions ;
— *Under Used* (UU) : *tier* stratégique signifiant « sous-utilisé » en français, réunissant les Pokémon jugés trop faibles face à ceux de la catégorie *Over Used* (OU) mais qui offrent cependant un défi (cf. sous-sous-section 4.5.4.1 p. 133) diversifié recherché par les joueurs ;
— *Rarely Used* (RU) : ou *Little Used* (LU), *tier* stratégique signifiant « rarement utilisé » en français ;
— *Never Used* (NU) : *tier* stratégique signifiant « jamais utilisé » en français ;
— PU : *tier* stratégique dont le nom n'est pas une abréviation et qui est considéré comme le *Never Used* (NU) du *Never Used* (NU) ;
— *Not Fully Evolved* (NFE) : *tier* stratégique signifiant « n'ayant pas complètement évolué » en français, réunissant les Pokémon n'étant pas de niveau maximum ;
— *Little Cup* (LC) : *tier* stratégique signifiant « petite coupe » en français, réunissant les Pokémon de niveau 5, et de premier stade d'évolution (mais évoluant), uniquement.

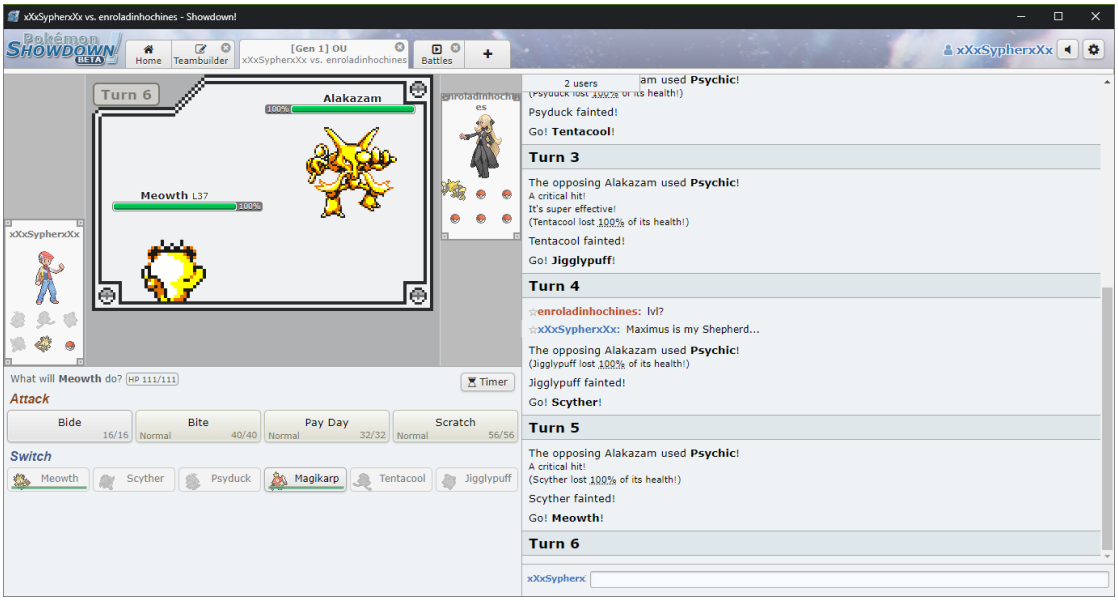


FIGURE 6.4. – Jeu vidéo Pokémon Showdown! : l'application de bureau [a]

a. Exemple d'équipe dans le jeu vidéo Pokémon Showdown!

6.2.2. Kaggle

Kaggle est un site Web, fondé en 2010 puis acquis par Google en 2017, proposant des compétitions d'apprentissage automatique [7] en science des données et mettant gracieusement à disposition des chercheurs des jeux de données pour réaliser des projets de fouilles de données ou d'apprentissage automatique.

Afin de compléter les informations nécessaires à notre cas d'utilisation, et en particulier pour le calcul de la `Rareté` (de la séquence d'apparition de l'équipe), qui constitue l'un des critères de notre cas d'utilisation, nous nous sommes appuyés sur la ressource Pokemon All Status Data (Gen1 to 9) mise à disposition sur Kaggle par Takamasa Kato, chercheur en science des données.

6.2.3. Classification atomique de Pokémon Showdown!

Nous estimons que le cœur du *gameplay* du jeu vidéo Pokémon Showdown! est composé des briques de *gameplay* (cf. sous-sous-section 4.4.2.4 p. 128) `Atteindre` (`At`), `Éviter` (`Ev`), `Détruire` (`Dt`), `Gérer` (`Gr`), `Choisir` (`Ch`) et `Déplacer` (`Dp`) (cf. figure 6.5).

7. L'apprentissage automatique (de l'anglais *machine learning*) constitue un champ d'étude de l'intelligence artificielle qui utilise des approches mathématiques et statistiques permettant « l'apprentissage » des systèmes informatiques (autrement dit l'amélioration de leurs performances algorithmique à répondre à des besoins sans avoir précisément été développé pour cela), le plus souvent à partir de grands ensembles de données.

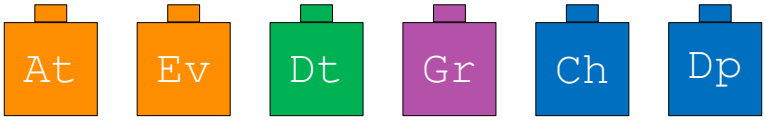


FIGURE 6.5. – Briques *gameplay* de Pokémon Showdown!

6.2.4. Relation exemple

Exemple 6.1 - La relation exemple `Pokémon` (cf. tableau 6.1) conserve, de notre expérimentation, les attributs `Tier` représentant le tier stratégique (cf paragraphe 6.2.1.5) de la partie[8], `Joueur`, la séquence de Pokémon jouée par le joueur, et `Adversaire`, la séquence de l'adversaire. La propriété `Rang` permet de catégoriser les joueurs en deux classes, les novices (N) et les experts (E), selon le système de classement Elo (cf. sous-sous-section 8.5.2.5 p. 255) du *ladder* de Pokémon Showdown!(cf. sous-sous-section 6.2.1.2). Les critères de choix afin de déterminer le « meilleur ordre d'apparition de Pokémon dans un combat » sont la `Rareté` des Pokémon de la séquence jouée par le joueur, la `Durée` (en nombre de tours) du combat et le taux d'`Échec` de l'ordre d'apparition des Pokémon du joueur par rapport à celui de l'adversaire. Tous ces critères sont, de préférence, à minimiser. L'attribut `Rareté` est invariable d'une catégorie de joueur à une autre. Lors de l'agrégation des données, nous confondons les combats miroirs, qui sont peu pertinent de part leur caractéristiques de données symétriques, et pour l'exemple, nous nous limitons à trois Pokémon par séquence, qu'elle soit du joueur ou de l'adversaire, et afin de simplifier leur usage dans leurs représentations, nous les indiquons par des listes de numéros de Pokémon, et leur affectons une lettre pour les désigner. Ainsi, la liste $121,113,006(A)$ correspond à la séquence des trois Pokémon Staross, Leveinard et Dracaufeu. Pour la lisibilité, nous arrondissons par un multiple de cinq les durées des combats et les taux d'échecs.

FIGURE 6.6. – Dracaufeu, Grodoudou, Alakazam, Flagadoss, Noadkoko, Leveinard, Staross, Ronflex[a]

a. De gauche à droite : Dracaufeu (006, UU), Grodoudou (040, UU), Alakazam (n°065, OU), Flagadoss (n°080, OU), Noadkoko (n°103, OU), Leveinard (n°113, OU), Staross (n°121, OU), Ronflex (n°143, OU) ; illustrations de Pokémon Versions Rouge Feu et Vert Feuille sur Poképédia (`https://www.pokepedia.fr/`).

8. Comme il est possible d'avoir des parties composées de Pokémon ayant des tiers stratégiques différents, nous considérons que le tier stratégique d'une partie est le tier stratégique le plus faible de l'ensemble des tiers stratégiques de tous les Pokémon joués par le joueur. Par exemple, une partie composée de la liste de Pokémon du joueur $121,113,006(A)$, pour Staross (OU), Leveinard (OU), Dracaufeu (UU) est de tier stratégique UU.

TABLEAU 6.1. – La relation exemple Pokémon

RowId	Tier [a]	Joueur [b]	Adversaire [c]	...	Rang [d]	Rareté [e]	Durée [f]	Échec [g]
1	*UU*	121,113,006(*A*)	065,113,143(*D*)	...	*N*	5	25	30
2	*OU*	065,103,065(*B*)	065,040,065(*E*)	...	*N*	4	65	50
3	*OU*	065,103,065(*B*)	121,113,121(*F*)	...	*N*	4	35	40
4	*OU*	065,103,065(*B*)	121,113,006(*A*)	...	*N*	4	85	40
5	*OU*	121,113,080(*C*)	121,113,006(*A*)	...	*N*	1	95	60
6	*OU*	121,113,080(*C*)	065,103,065(*B*)	...	*N*	1	35	50
7	*OU*	065,113,143(*D*)	065,103,065(*B*)	...	*N*	9	85	60
8	*OU*	065,113,143(*D*)	121,113,080(*C*)	...	*N*	9	85	70
9	*UU*	065,040,065(*E*)	065,113,143(*D*)	...	*N*	7	25	50
10	*UU*	065,040,065(*E*)	065,040,065(*E*)	...	*N*	7	65	30
11	*UU*	121,113,006(*A*)	065,113,143(*D*)	...	*E*	5	20	30
12	*OU*	065,103,065(*B*)	065,040,065(*E*)	...	*E*	4	60	45
13	*OU*	065,103,065(*B*)	121,113,121(*F*)	...	*E*	4	30	30
14	*OU*	065,103,065(*B*)	121,113,006(*A*)	...	*E*	4	80	50
15	*OU*	121,113,080(*C*)	121,113,006(*A*)	...	*E*	1	90	70
16	*OU*	121,113,080(*C*)	065,103,065(*B*)	...	*E*	1	30	30
17	*OU*	065,113,143(*D*)	065,103,065(*B*)	...	*E*	9	80	50
18	*OU*	065,113,143(*D*)	121,113,080(*C*)	...	*E*	9	90	70
19	*UU*	065,040,065(*E*)	065,113,143(*D*)	...	*E*	7	20	30
20	*UU*	065,040,065(*E*)	065,040,065(*E*)	...	*E*	7	60	45

a. Tier stratégique.

b. Avec les numéros officiels des Pokémon (et leur tier stratégique) :
— n°006 : Dracaufeu (UU) ;
— n°040 : Grodoudou (UU) ;
— n°065 : Alakazam (OU) ;
— n°080 : Flagadoss (OU) ;
— n°103 : Noadkoko (OU) ;
— n°113 : Leveinard (OU) ;
— n°121 : Staross (OU) ;
— n°143 : Ronflex (OU).

c. *idem supra.*

d. Catégorie de joueur, avec seulement deux partitions considérées : novice (N) et expert (E).

e. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de Rareté r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

f. En nombre de tours de combat au total.

g. En pourcentage.

TABLEAU 6.2. – La relation exemple $\texttt{Pokémon}_1$, des joueurs novices

RowId	Tier [a]	Joueur	Adversaire	...	Rareté [b]	Durée [c]	Échec [d]
1	*UU*	*A*	*D*	...	5	25	30
2	*OU*	*B*	*E*	...	4	65	50
3	*OU*	*B*	*F*	...	4	35	40
4	*OU*	*B*	*A*	...	4	85	40
5	*OU*	*C*	*A*	...	1	95	60
6	*OU*	*C*	*B*	...	1	35	50
7	*OU*	*D*	*B*	...	9	85	60
8	*OU*	*D*	*C*	...	9	85	70
9	*UU*	*E*	*D*	...	7	25	50
10	*UU*	*E*	*E*	...	7	65	30

a. Tier stratégique.

b. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de Rareté r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

c. En nombre de tours de combat au total.

d. En pourcentage.

TABLEAU 6.3. – La relation exemple $\texttt{Pokémon}_2$, des joueurs experts

RowId	Tier [a]	Joueur	Adversaire	...	Rareté [b]	Durée [c]	Échec [d]
1	*UU*	*A*	*D*	...	5	20	30
2	*OU*	*B*	*E*	...	4	60	45
3	*OU*	*B*	*F*	...	4	30	30
4	*OU*	*B*	*A*	...	4	80	50
5	*OU*	*C*	*A*	...	1	90	70
6	*OU*	*C*	*B*	...	1	30	30
7	*OU*	*D*	*B*	...	9	80	50
8	*OU*	*D*	*C*	...	9	90	70
9	*UU*	*E*	*D*	...	7	20	30
10	*UU*	*E*	*E*	...	7	60	45

a. Tier stratégique.

b. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de Rareté r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

c. En nombre de tours de combat au total.

d. En pourcentage.

Exemple 6.2 - Considérons que nous recherchons les renversements de tendances se produisant entre les stratégies des joueurs de Rang novice et ceux de Rang expert de la relation Pokémon (cf. tableau 6.1). La relation $\texttt{Pokémon}_1$ (cf. tableau 6.2) correspond aux joueurs novices et $\texttt{Pokémon}_2$ (cf. tableau 6.3) sélectionne les tuples de Pokémon pour lesquels Rang $=' E'$. La relation fusionnée dédoublonnée $\texttt{Pokémon}^{+}_{Fus}$ (cf. tableau 6.4) est donné à titre indicatif pour l'instant (avec R pour Rareté, $\texttt{D}_1$ pour $\texttt{Durée}_1$, $\texttt{É}_1$ pour $\texttt{Échec}_1$, $\texttt{D}_2$ pour $\texttt{Durée}_2$ et $\texttt{É}_2$) pour $\texttt{Échec}_2$.

TABLEAU 6.4. – Relation fusionnée dédoublonnée $\texttt{Pokémon}^{+}_{Fus}$

RowId	Tier [a]	Joueur	Adversaire	...	R [b]	$\texttt{D}_1$ [c]	$\texttt{É}_1$ [d]	$\texttt{D}_2$ [e]	$\texttt{É}_2$ [f]
1	*UU*	*A*	*D*	...	5	25	30	20	30
2	*OU*	*B*	*E*	...	4	65	50	60	45
3	*OU*	*B*	*F*	...	4	35	40	30	30
4	*OU*	*B*	*A*	...	4	85	40	80	50
5	*OU*	*C*	*A*	...	1	95	60	90	70
6	*OU*	*C*	*B*	...	1	35	50	30	30
7	*OU*	*D*	*B*	...	9	85	60	80	50
8	*OU*	*D*	*C*	...	9	85	70	90	70
9	*UU*	*E*	*D*	...	7	25	50	20	30
10	*UU*	*E*	*E*	...	7	65	30	60	45

a. Tier stratégique.
b. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de R r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.
c. En nombre de tours de combat au total.
d. En pourcentage.
e. *idem* $\texttt{D}_1$.
f. *idem* $\texttt{É}_1$.

Exemple 6.3 - La relation fusionnée dédoublonnée de la relation exemple est donnée dans le tableau 6.4 (avec R pour Rareté, $\texttt{D}_1$ et $\texttt{É}_1$ qui correspondent respectivement aux attributs Durée et Échec des joueurs novices et $\texttt{D}_2$ et $\texttt{É}_2$ qui correspondent respectivement aux attributs Durée et Échec des joueurs experts).

Nous rappelons que le Skycube peut être représenté par un treillis comme celui employé pour le cube de données des joueurs novices (cf. figure 6.7) ou celui des experts (cf. figure 6.8). Les cuboïdes du Skycube, appelés Skycuboïdes, sont regroupés, en fonction de leur nombre de critères, par niveau, du bas du treillis correspondant aux Skycuboïdes n'ayant qu'un seul critère (si l'on omet $\emptyset$) au haut de ce dernier présentant le Skycuboïde avec l'ensemble des critères considérés.

Exemple 6.4 - La figure 6.7 présente le Skycube associé à la relation exemple $\texttt{Pokémon}_1$ (cf. tableau 6.2) pour les joueur de Rang novice, et la figure 6.8 le Skycube associé à

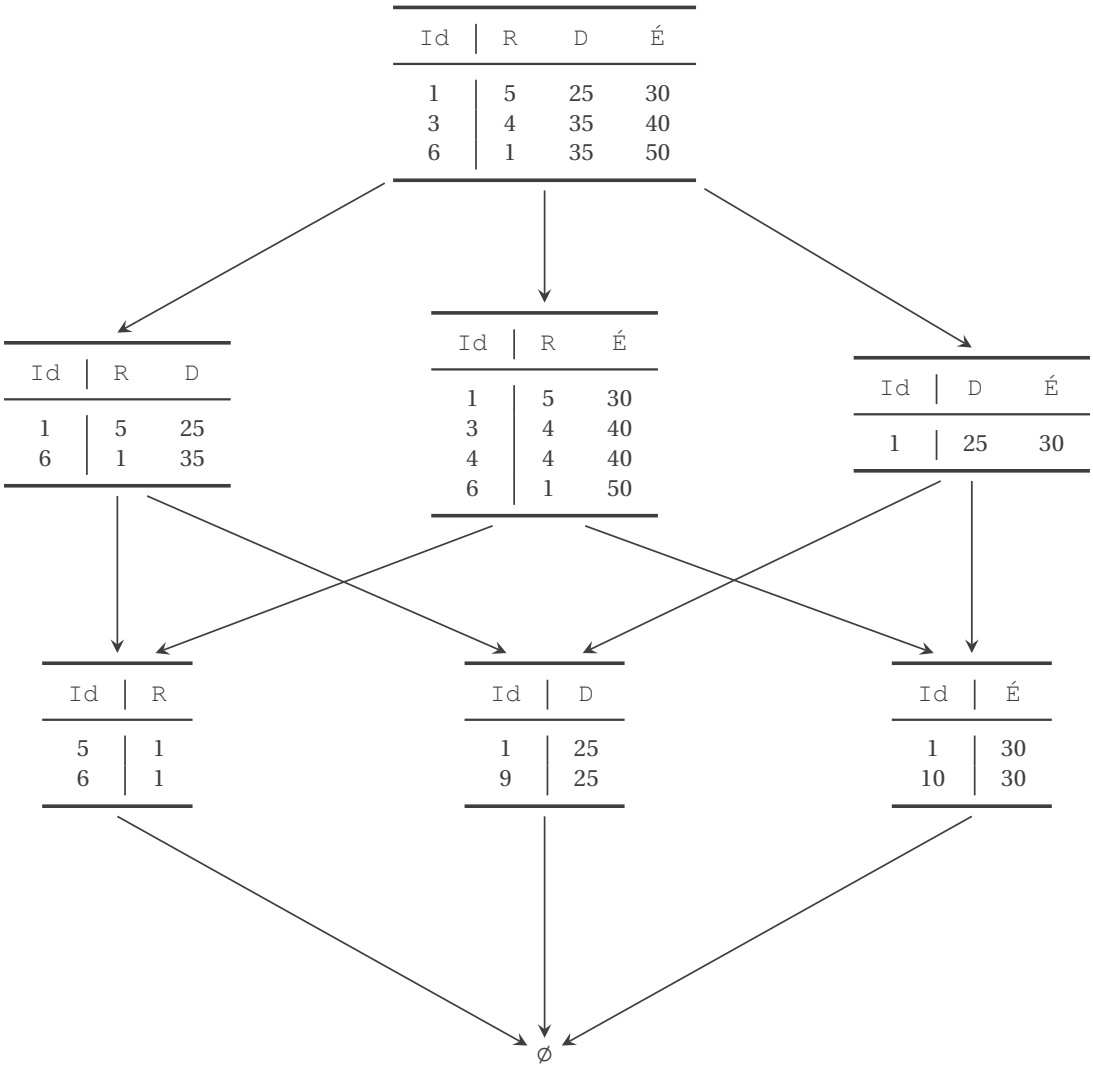


FIGURE 6.7. – Représentation du treillis du Skycube de la relation $\texttt{Pokémon}_1$

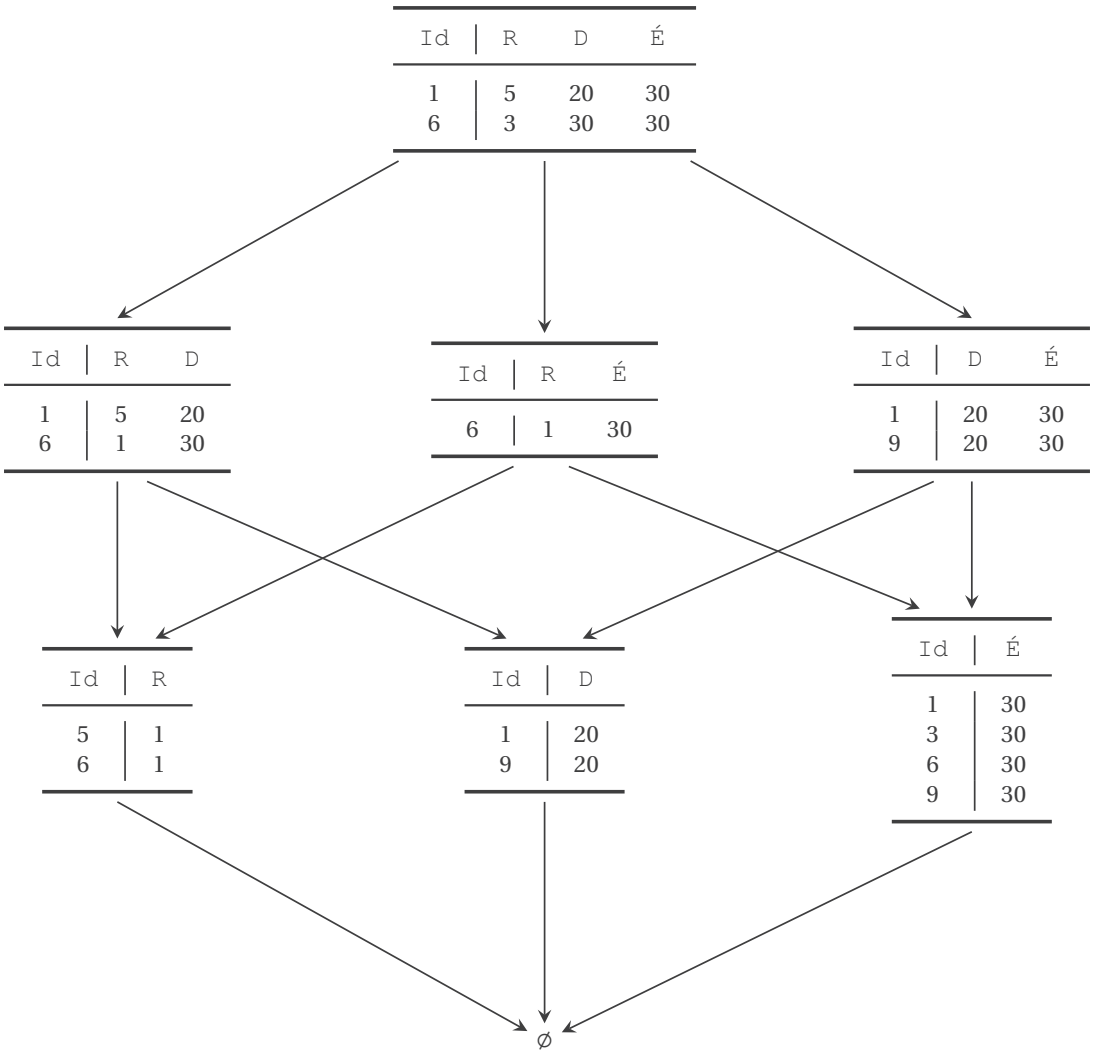


FIGURE 6.8. – Représentation du treillis du Skycube de la relation $\texttt{Pokémon}_2$

la relation exemple $\texttt{Pokémon}_2$ (cf. tableau 6.3) des joueurs experts. Les critères sont symbolisés par leur initiale (R pour Rareté, D pour Durée et É pour Échec).

6.3. Treillis des Concepts Accords et Skylines

Comme dans le cas du cube de données (cf. section 2.2 p. 27), le Skycube (cf. sous-section 2.4.3) peut contenir des informations superflues. C'est cette problématique qui a motivé la proposition de représentations réduites du Skycube (Pei, Yuan, Lin et al. 2006 ; Z. Liu, L. Li, M. Zhang et al. 2023). Notre contribution s'intéresse à la même problématique de réduction en combinant nous aussi l'analyse de concepts formels (Ganter et Wille 1999) et le Skyline (cf. sous-section 2.4.2 p. 75). Pour éviter le coût important de reconstruction des cuboïdes Skylines induit par le regroupement orienté valeur de Pei, Yuan, Lin et al. 2006, notre méthode de réduction choisit une approche de regroupement orientée critère en se basant sur les ensembles en accord.

Dans cette section, notre objectif est de définir un cadre de travail formel combinant le concept d'ensemble en accord et celui de treillis des concepts. Nous proposons une nouvelle structure, le treillis des concepts Accords d'une relation, sur laquelle s'appuie notre matérialisation partielle du Skycube. Après un rappel des notions de treillis des partitions, d'ensemble en accord, des classes d'équivalence associées et du treillis des concepts Accords (Nedjar, Pesci, Lakhal et al. 2011a ; Nedjar, Pesci, Lakhal et al. 2011b ; Lakhal, Nedjar et Cicchetti 2017), nous caractérisons le treillis des concepts Skylines pour la matérialisation partielle du Skycube émergent.

6.3.1. Treillis des partitions

Les concepts rappelés dans cette sous-section ont été utilisés pour la résolution de problèmes de bases de données (Spyratos 1987).

Définition 6.1 (Partition d'un ensemble) - Soit E un ensemble, une partition $\pi(E)$ [9] d'un ensemble E est une famille de parties de cet ensemble telle que chaque élément de E appartient exactement à une seule de ces familles (ou classes). Autrement dit, $\pi(E)$ est une famille d'ensembles deux à deux disjoints ($\forall X, Y \in \pi(E)$ on a $X \cap Y = \emptyset$) dont l'union est égale à E ($\bigcup_{X \in \pi(E)} = E$).

Définition 6.2 (Relation d'ordre entre partitions) - Soit $\pi(E)$, $\pi'(E)$ deux partitions d'un même ensemble E, $\pi(E)$ est dite plus fine que $\pi'(E)$ si et seulement si toute classe de $\pi(E)$ est obtenue par subdivision de classes de $\pi'(E)$ [10]. La relation de finesse entre deux partitions est une relation d'ordre partiel notée $\sqsubseteq$. Elle est définie comme suit :

$$\pi(E) \sqsubseteq \pi'(E) \Leftrightarrow \pi(E) \text{ est plus fine que } \pi'(E) \text{ [11]} \Leftrightarrow (\forall X \in \pi(E), \exists X' \in \pi'(E), X \subseteq X')$$

9. S'il n'y a pas d'ambiguïté sur l'ensemble E, nous notons la partition π.
10. De façon équivalente, $\pi(E)$ est plus fine que $\pi'(E)$ si et seulement si toute classe de $\pi'(E)$ est l'union de classes de $\pi(E)$.
11. Réciproquement $\pi'(E)$ est dite plus grossière que $\pi(E)$.

Définition 6.3 (Produit de partitions) - Soit $\pi(E)$ et $\pi'(E)$ deux partitions d'un même ensemble E. Le produit des partitions $\pi(E)$ et $\pi'(E)$, noté $\pi(E) \bullet \pi'(E)$, est obtenu comme suit :

$$\pi(E) \bullet \pi'(E) = \{Z = X \cap Y \mid Z \neq \emptyset, X \in \pi(E) \text{ et } Y \in \pi'(E)\}$$

Définition 6.4 (Somme de partitions) - Soit la fonction outil R suivante :

$$R(e,F) = \bigcup_{\substack{X \in F \\ e \in X}} X$$

Avec e un élément d'un ensemble E, F une famille de parties de E. $R(e,F)$ correspond à l'union des ensemble de F contenant l'élément e.

Soit $\pi(E)$ et $\pi'(E)$ deux partitions d'un même ensemble E. La somme des partitions $\pi(E)$ et $\pi'(E)$, notée $\pi(E) + \pi'(E)$, est obtenue par fermeture transitive de l'opération qui fait correspondre à un élément de E l'ensemble des éléments de ses classes dans $\pi(E)$ et $\pi'(E)$ (BIRKHOFF 1970). La suite S est définie ci-dessous pour formaliser ce calcul :

$$\begin{cases} S_0 = \max_{\subseteq}(\pi(E) \cup \pi'(E)) \\ S_n = \max_{\subseteq}(\{R(e, S_{n-1}) \mid e \in E\}) \end{cases}$$

Ainsi l'opérateur somme peut se définir de la manière suivante :

$$\pi(E) + \pi'(E) = S_k \text{ avec } k \text{ tel que } S_k = S_{k-1}$$

Théorème 6.1 (Treillis des partitions) - Soit $\Pi(E)$ l'ensemble des partitions possibles d'un ensemble E. L'ensemble ordonné $\langle \Pi(E), \sqsubseteq \rangle$ forme un treillis complet nommé treillis des partitions de E. $\forall P \subseteq \Pi(E)$, son *infimum* ou borne inférieure ($\bigwedge$) et son *supremum* ou borne supérieure ($\bigvee$) sont donnés ci-après :

$$\bigwedge P = \bullet_{\pi \in P} \pi$$
$$\bigvee P = +_{\pi \in P} \pi$$

Exemple 6.5 - Le diagramme de Hasse du treillis des partitions de l'ensemble $E = \{1,2,3,4\}$ est donné dans la figure 6.9. Les partitions les plus grossières sont en bas et les plus fines en haut.

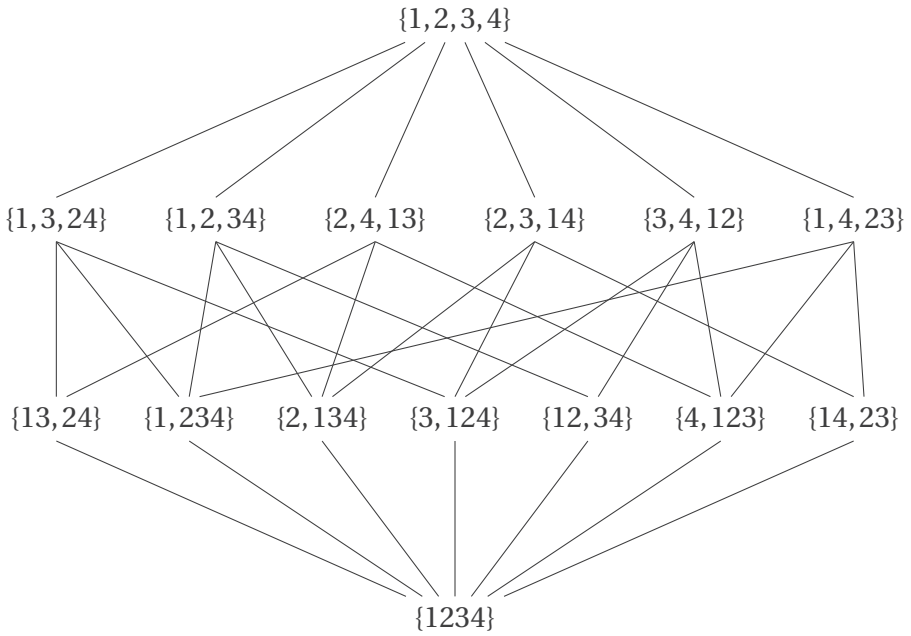


FIGURE 6.9. – Diagramme de Hasse du treillis des partitions de $E = \{1,2,3,4\}$

6.3.2. Ensembles en accord

Le concept d'ensemble en accord et son système de fermeture associé, introduit par BEERI, DOWD, FAGIN et al. 1984 pour caractériser la relation d'Armstrong, a été utilisé avec succès pour l'extraction de dépendances fonctionnelles exactes et approximatives (LOPES, PETIT et LAKHAL 2002). Deux tuples sont en accord sur un ensemble d'attributs X s'ils partagent la même valeur pour X.

Nous commençons par présenter, avec un autre formalisme, la définition d'ensemble en accord déjà vu précédemment (cf. sous-section 3.6 p. 107), puis nous donnons d'autres définitions qui en découlent.

Définition 6.5 (Ensemble en accord) - Soit t, t' deux tuples de r et $X \subseteq \mathscr{C}$ un ensemble d'attributs (critères dans notre contexte). t, t' sont en accord sur X si et seulement si $t_i[X] = t_j[X]$. L'ensemble des attributs en accord de t et t', $Acc(t,t')$ est défini comme suit :

$$Acc(t,t') = \{C \in \mathscr{C} \mid t[C] = t'[C]\}$$

Cette définition peut être généralisée pour un ensemble de tuples $T \subseteq r$ composé d'au moins deux éléments :

$$Acc(T) = \{C \in \mathscr{C} \mid t[C] = t'[C], \forall\, t,t' \in T\}$$

Définition 6.6 (Ensemble en accord d'une relation) - L'ensemble des attributs en accord $Accords$ d'une relation r est défini comme suit :

$$Accords(r) = \{Acc(t_i,t_j) \mid t_i,t_j \in r \text{ et } i \neq j\}$$

Cet ensemble peut être redéfini de manière équivalente :

$$Accords(r) = \{Acc(T) \mid \forall T \subseteq r \text{ et } |T| \geq 2\}$$

Exemple 6.6 - Avec la relation $\texttt{Pokémon}_1$ (cf. tableau 6.2), $Acc(t_7, t_8) = RD$ car ces deux tuples partagent la même valeur sur les attributs `Rareté` ainsi que `Durée`, et ont des valeurs différentes pour `Échec`. De même, $Acc(\{t_2, t_6, t_9\}) = \text{É}$ car ces trois tuples ont la même valeur uniquement pour le critère `Échec`. L'ensemble des ensembles d'attributs en accord de la relation $\texttt{Pokémon}_1$ est : $Accords(\texttt{Pokémon}_1) = \{\emptyset, R, D, \text{É}, RD, R\text{É}\}$.

Définition 6.7 (Classe d'équivalence d'un tuple) **-** Soit r une relation et $C \subseteq \mathscr{C}$ un ensemble de critères. La classe d'un tuple t selon C, notée $[t]_C$, est définie comme l'ensemble des identifiants i (`RowId`) de tous les tuples $t_i \in r$ en accord avec t selon C (*i.e.* l'ensemble des identifiants des tuples t_i partageant avec t les mêmes valeurs pour C). Nous avons donc :

$$[t]_C = \{i \in Tid(r) \mid t_i[C] = t[C]\}$$

Exemple 6.7 - Avec la relation $\texttt{Pokémon}_1$, $[t_2]_R = \{2, 3, 4\}$ car seuls les tuples t_2, t_3 et t_4 partagent la même valeur sur le critère `Rareté`.

6.3.3. Concepts Accords d'une relation

Notre objectif est de définir un treillis des concepts particulier se basant sur les ensembles en accord et les partitions (Spyratos 1987). Pour atteindre ce but, nous caractérisons une instance de la connexion de Galois entre d'une part le treillis des parties de l'ensemble des critères et d'autre part le treillis des partitions de l'ensemble des identifiants de tuples. Cette connexion nous permet de dériver des opérateurs de fermeture duals, de définir un concept Accord et de caractériser le treillis des concepts Accords.

Définition 6.8 - Soit $\texttt{RowId} : r \rightarrow \mathbb{N}$ une application qui associe à chaque tuple un unique entier naturel et $Tid(r) = \{Rowid(t) \mid t \in r\}$. Soit f, g deux applications entre les ensembles ordonnés $\langle \Pi(Tid(r)), \sqsubseteq \rangle$ et $\langle \mathcal{P}(\mathscr{C}), \subseteq \rangle$ qui sont définies comme suit :

$$\begin{array}{rrcl} f: & \langle \Pi(Tid(r)), \sqsubseteq \rangle & \longrightarrow & \langle \mathcal{P}(\mathscr{C}), \subseteq \rangle \\ & \pi & \longmapsto & \bigcap\limits_{[t] \in \pi} Acc(\{t_i \mid i \in [t]\}) \\ g: & \langle \mathcal{P}(\mathscr{C}), \subseteq \rangle & \longrightarrow & \langle \Pi(Tid(r)), \sqsubseteq \rangle \\ & C & \longmapsto & \{[t]_C \mid t \in r\} \end{array}$$

Pour un ensemble de critères C, l'ensemble des classes d'équivalence selon C forme une partition de $Tid(r)$. C'est la fonction g qui associe à C cette partition des identifiants. Cette dernière est notée π_C et définie comme suit : $\pi_C = g(C)$. L'ensemble de toutes les partitions π_C possibles est noté $\Pi_{\mathcal{P}(\mathscr{C})}$. La fonction f fait l'association

inverse de g.

Exemple 6.8 - Avec la relation $\texttt{Pokémon}_2$, en prenant les ensembles de critères R, D et DÉ, on a $g(R) = \{1,234,56,78,9[10]\}$ (sont dans la même classe d'équivalence t_1 seul, t_2 et t_3 et t_4, t_5 et t_6, t_7 et t_8 ainsi qu'enfin t_9 et $t_{[}10]$), $g(D) = \{19,2[10],36,47,58\}$ et $g(D\text{É}) = \{19,2[10],36,47,58\}$. Avec $\{19,2[10],36,47,58\}$ et $\{1,2,34,56,78,9[10]\}$ comme partitions, on a $f(\{19,2[10],36,47,58\}) = D\text{É}$ et $f(\{1,2,34,56,78,9[10]\}) = R$.

Proposition 6.2 - Le couple d'applications $gc = (f,g)$ est une connexion de Galois entre le treillis des parties des critères $\mathscr{C}$ et le treillis des partitions de $Tid(r)$.

Définition 6.9 (Opérateurs de fermeture) **-** Le couple $gc = (f,g)$ étant un cas particulier de la connexion de Galois, les compositions $f \circ g$ et $g \circ f$ des deux applications sont des opérateurs de fermeture (GANTER et WILLE 1999) définis ainsi :

$$
\begin{array}{rccl}
h: & \mathcal{P}(\mathscr{C}) & \longrightarrow & \mathcal{P}(\mathscr{C}) \\
 & C & \longmapsto & f(g(C)) = \bigcap\limits_{\substack{C' \in Accords(r) \\ C \subseteq C'}} C' \\
h': & \Pi(Tid(r)) & \longrightarrow & \Pi(Tid(r)) \\
 & \pi & \longmapsto & g(f(\pi)) = \bullet_{\substack{\pi' \in \Pi_{\mathcal{P}(\mathscr{C})} \\ \pi \sqsubseteq \pi'}} \pi'
\end{array}
$$

Corollaire 6.3 - h et h' satisfont les propriétés suivantes :

1. $C \subseteq C' \Rightarrow h(C) \subseteq h(C')$ et $\pi \sqsubseteq \pi' \Rightarrow h'(\pi) \sqsubseteq h'(\pi')$ (isotonie)
2. $C \subseteq h(C)$ et $\pi \sqsubseteq h'(\pi)$ (extensivité)
3. $h(C) = h(h(C))$ et $h'(\pi) = h'(h'(\pi))$ (idempotence)

Exemple 6.9 - Avec la relation $\texttt{Pokémon}_2$, en prenant les ensembles de critères D et DÉ, d'après l'exemple précédent nous avons :

— $h(D) = f(g(D)) = f(\{19,2[10],36,47,58\}) = D\text{É}$

— $h(D\text{É}) = f(g(D\text{É})) = f(\{19,2[10],36,47,58\}) = D\text{É}$

Avec les partitions $\{19,2[10],36,47,58\}$ et $\{1,2,34,56,78,9[10]\}$, nous avons :

— $h'(\{19,2[10],36,47,58\}) = g(f(\{19,2[10],36,47,58\})) = g(D\text{É}) = \{19,2[10],36,47,58\}$

— $h'(\{1,2,34,56,78,9[10]\}) = g(f(\{1,2,34,56,78,9[10]\})) = g(R) = \{1,234,56,78,9[10]\}$

Définition 6.10 (Concepts Accords) **-** Un concept Accord d'une relation r est un couple (C,π) associant un ensemble de critères à une partition des identifiants : $C \in \mathcal{P}(\mathscr{C})$ et $\pi \in \Pi(Tid(r))$. Les éléments de ce couple doivent être liés par les conditions suivantes : $C = f(\pi)$, $\pi = g(C) = \pi_C$.

Soit $c_a = (C_{c_a}, \pi_{c_a})$ un concept Accord de r, nous appelons π_{c_a} l'extension de c_a (notée $ext(c_a)$) et C_{c_a} son intension (notée $int(c_a)$). L'ensemble de tous les concepts Accords d'une relation r est noté $\texttt{CONCEPTSACCORDS}(r)$.

Théorème 6.4 (Treillis des concepts Accords) **-** Soit $\texttt{CONCEPTSACCORDS}(r)$ l'ensemble des concepts Accords d'une relation r. $\langle \texttt{CONCEPTSACCORDS}(r), \leq \rangle$ est un

ensemble ordonné formant un treillis complet nommé treillis des concepts Accords [12]. Nous avons, $\forall P \subseteq \texttt{CONCEPTSACCORDS}(r)$, *l'infimum* ou borne inférieure ($\bigwedge$) et le *supremum* ou borne supérieure ($\bigvee$) sont donnés ci-après :

$$\bigwedge P = (\bigcap_{c_a \in P} int(c_a),\ h'(+_{c_a \in P} ext(c_a)))$$
$$\bigvee P = (h(\bigcup_{c_a \in P} int(c_a)),\ \bullet_{c_a \in P} ext(c_a))$$

Démonstration. Puisque le couple $gc = (f, g)$ est une connexion de Galois, le treillis des concepts Accords est un treillis des concepts d'après le théorème fondamental de Ganter et Wille 1999. □

Exemple 6.10 - La figure 6.10 donne le diagramme de Hasse du treillis des concepts Accords de la relation $\texttt{Pokémon}_1$. Si nous prenons $c_a = (RD, \{1,2,3,4,5,6,78,9,[10]\})$ et $c_b = (R\text{É}, \{1,2,34,5,6,7,8,9,[10]\})$, nous avons :

$$\begin{aligned} c_a \wedge c_b &= (RD \cap R\text{É}, h'(\{19,2[10],36,47,58\} + \{1,234,56,78,9[10]\})) \\ &= (R, h'(\{1,234,56,78,9[10]\})) = (R, \{1,234,56,78,9[10]\}) \\ c_a \vee c_b &= (h(RD \cup R\text{É}), \{19,2[10],36,47,58\} \bullet \{1,234,56,78,9[10]\}) \\ &= (h(RD\text{É}), \{1,2,3,4,5,6,7,8,9,[10]\}) = (RD\text{É}, \{1,2,3,4,5,6,7,8,9,[10]\}) \end{aligned}$$

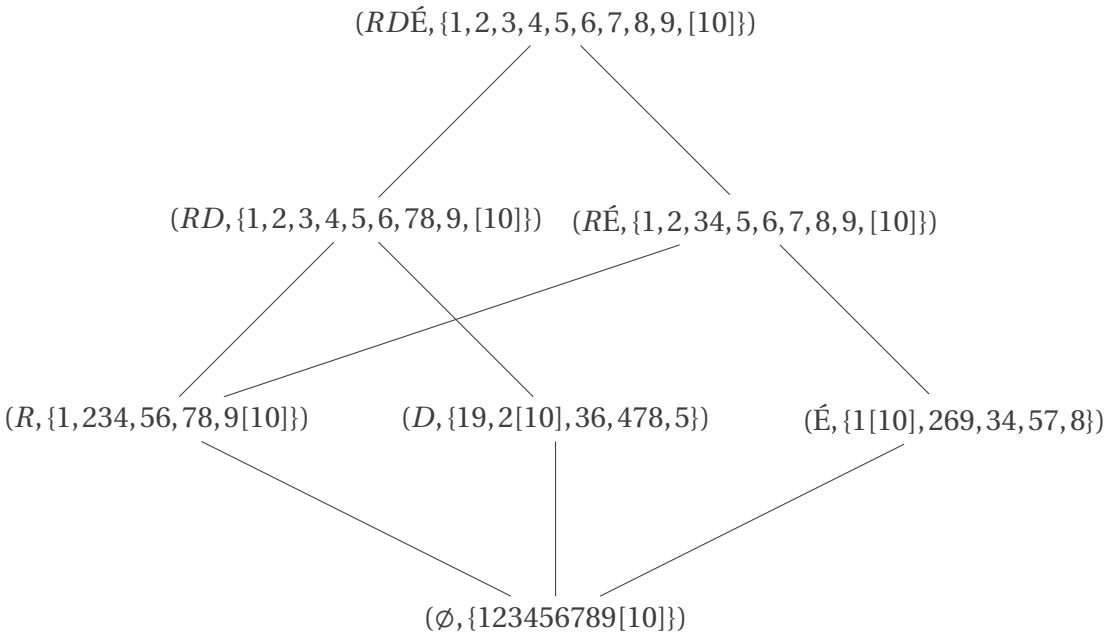


FIGURE 6.10. – Diagramme de Hasse du treillis des concepts Accords des novices

12. Soit (C_1,π_1), $(C_2,\pi_2) \in \texttt{CONCEPTSACCORDS}(r)$, $(C_1,\pi_1) \leq (C_2,\pi_2) \Leftrightarrow C_1 \subseteq C_2$ (ou de manière équivalente $\pi_2 \sqsubseteq \pi_1$).

Exemple 6.11 - La figure 6.11 donne le diagramme de Hasse du treillis des concepts Accords de la relation $\texttt{Pokémon}_2$. Le couple $(D\text{É}, \{19, 2[10], 36, 47, 58\})$ est un concept Accord car d'après les exemples 6.8 et 6.9 nous avons $g(D\text{É}) = \{19, 2[10], 36, 47, 58\}$ et $f(\{19, 2[10], 36, 47, 58\}) = D\text{É}$. À l'inverse le couple $(D, \{19, 2[10], 36, 47, 58\})$ ne constitue pas un concept Accord car $f(\{1, 2, 35, 4\}) \neq D$. Soit $c_a = (D\text{É}, \{19, 2[10], 36, 47, 58\})$ et $c_b = (R, \{1, 234, 56, 78, 9[10]\})$ deux concepts Accords, nous avons :

$$\begin{aligned} c_a \wedge c_b &= (D\text{É} \cap R, h'(\{19, 2[10], 36, 47, 58\} + \{1, 234, 56, 78, 9[10]\})) \\ &= (\emptyset, h'(\{123456789[10]\})) = (\emptyset, \{123456789[10]\}) \\ c_a \vee c_b &= (h(D\text{É} \cup R), \{19, 2[10], 36, 47, 58\} \bullet \{1, 234, 56, 78, 9[10]\}) \\ &= (h(RD\text{É}), \{1, 2, 3, 4, 5, 6, 7, 8, 9, [10]\}) = (RD\text{É}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, [10]\}) \end{aligned}$$

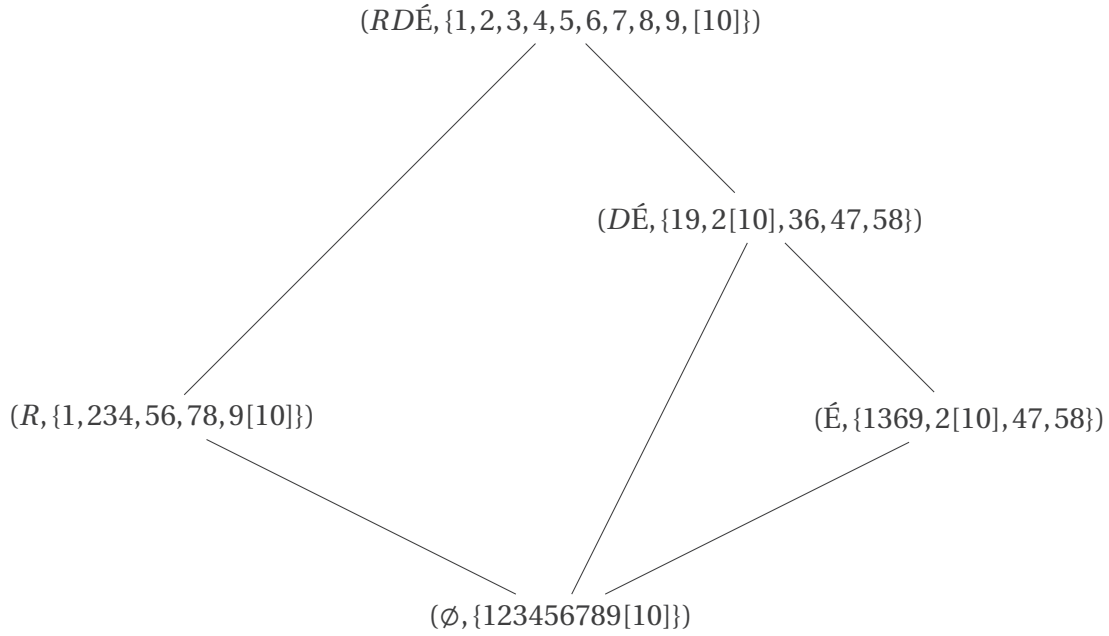


FIGURE 6.11. – Diagramme de Hasse du treillis des concepts Accords des experts

Proposition 6.5 - Pour tout ensemble de critères $C \subseteq \mathscr{C}$, la partition associée π_C est identique à la partition de sa fermeture :

$$\forall C \subseteq \mathscr{C}, \pi_C = \pi_{h(C)}$$

La proposition précédente signifie que la fermeture d'un ensemble de critères C peut être vue comme le plus grand sur-ensemble de C ayant la même partition.

Démonstration. Par définition $\forall C \subseteq \mathscr{C}$, $\pi_C = g(C)$ et $h(C) = f(g(C))$. Nous avons donc $\pi_{h(C)} = g(f(g(C))$. Or, le couple $gc = (f, g)$ étant une connexion de Galois, on

a $g \circ f \circ g = g$ (GANTER et WILLE 1999). Ainsi, on a donc bien $\pi_{h(C)} = g(f(g(C))) = g(C) = \pi_C$. □

Exemple 6.12 - Avec la relation $\texttt{Pokémon}_2$, en prenant comme ensemble de critères D, d'après les exemples 6.8 et 6.9, nous avons : $\pi_D = g(D) = \{19,2[10],36,47,58\}$ et $\pi_{h(D)} = g(h(D)) = g(RD\text{É}) = \{1,2,3,4,5,6,7,8,9,[10]\}$.

6.3.4. Treillis des Concepts Skylines et Skycube

Le treillis des concepts Skylines est un treillis des concepts Accords contraints. Nous présentons une propriété fondamentale nous permettant de ne matérialiser que partiellement le Skycube en éliminant certains cuboïdes. Puis, nous indiquons ensuite comment de tels cuboïdes peuvent être reconstruits facilement.

6.3.4.1. Treillis des concepts Skylines

Soit π_C une partition de r sur C. Par définition, les tuples d'une même classe d'équivalence $[t]_C \in \pi_C$ sont indistinguables sur C (ils partagent tous la même projection sur C). Si t est non dominé sur C, il en sera de même pour tous les autres tuples de sa classe (et réciproquement). Il suffit donc de tester la dominance d'un seul tuple d'une classe pour savoir si l'ensemble des tuples de la classe appartient ou non au Skycuboïde selon C. Ainsi, pour optimiser le calcul d'un Skycuboïde selon C à partir de sa partition π_C, on conserve uniquement un seul représentant de chaque classe d'équivalence. Cet ensemble est noté $reps(\pi_C)$. On diminue alors la taille de l'entrée en éliminant les tuples qui auraient conduit à un grand nombre de comparaisons inutiles. Pour prendre en compte les particularités du calcul de l'opérateur Skyline sur une partition, nous introduisons l'opérateur suivant.

Définition 6.11 (Opérateur Π-S) **-** Soit C un ensemble de critères et π une partition de r. On définit un nouvel opérateur Π-S de la manière suivante :

$$\begin{aligned}\Pi\text{-}S_C(\pi_C) &= \{[t_i] \in \pi_C \mid \forall\, t_j \in r \text{ on a } t_j \not\succ_C t_i\} \\ &= \{[t_i] \in \pi_C \mid t_i \in \Pi\text{-}S_C(r)\}\end{aligned}$$

Définition 6.12 (Concepts Skylines) **-** Soit $c_a = (C,\pi) \in \text{CONCEPTSACCORDS}(r)$ un concept Accord d'une relation r. Le concept Skyline associé c_s est défini de la manière suivante :

$$c_s = (C, \Pi\text{-}S_C(\pi))$$

Il y a exactement autant de concepts Skylines que de concepts Accords. Nous notons $\text{CONCEPTSSKYLINES}(r)$ l'ensemble des concepts Skylines associés aux concepts Accords de r.

Les concepts Skylines sont des concepts Accords où les partitions ont été contraintes. Ainsi, ce type de concept n'est plus forcément ordonné par la relation $\sqsubseteq$ entre partitions. La relation d'ordre $\leq$ entre concepts Skylines s'exprime donc de la manière sui-

vante : $\forall (C_1, \Pi\text{-}S_{C_1}(\pi_1)), (C_2, \Pi\text{-}S_{C_2}(\pi_2)) \in \texttt{CONCEPTSSKYLINES}(r)$, alors $(C_1, \Pi\text{-}S_{C_1}(\pi_1)) \leq (C_2, \Pi\text{-}S_{C_2}(\pi_2)) \Leftrightarrow C_1 \subseteq C_2$.

Définition 6.13 (Treillis des concepts Skylines) - $\langle \texttt{CONCEPTSSKYLINES}(r), \leq \rangle$ est un ensemble ordonnée qui forme un treillis complet nommé treillis des concepts Skylines. Il est isomorphe au treillis des concepts Accords.

Exemple 6.13 - La figure 6.12 donne le diagramme de Hasse du treillis des concepts Skylines de la relation $\texttt{Pokémon}_1$. La représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_1$, permettant de constater le gain obtenu avec le Skycuboïde estompé, est donné par la figure 6.13.

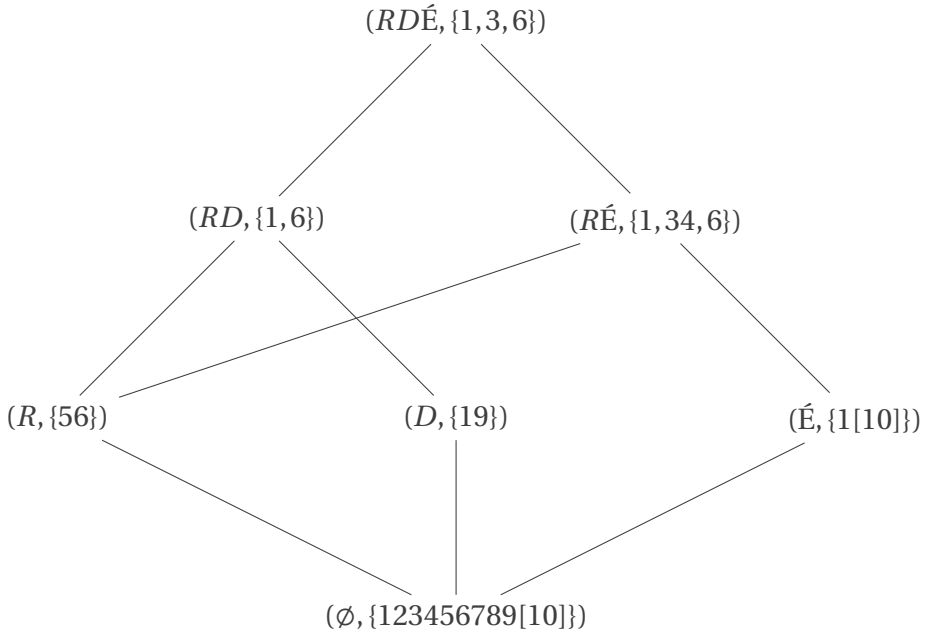


FIGURE 6.12. – Diagramme de Hasse du treillis des concepts Skylines des novices

Exemple 6.14 - La figure 6.14 donne le diagramme de Hasse du treillis des concepts Skylines de la relation $\texttt{Pokémon}_2$. D'après l'exemple 6.11, nous savons que le couple $c_a = (D\text{É}, \{19, 2[10], 36, 47, 58\})$ est un concept Accord. Le concept Skyline c_s associé est $c_s = (D\text{É}, \Pi\text{-}S_{D\text{É}}(\{19, 2[10], 36, 47, 58\})) = (D\text{É}, \{19\})$. Les autres identifiants sont éliminés de l'extension par l'opérateur Π-S car les tuples correspondants sont dominés. La représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_2$, permettant de constater le gain obtenu avec les Skycuboïdes estompés, est donné par la figure 6.15.

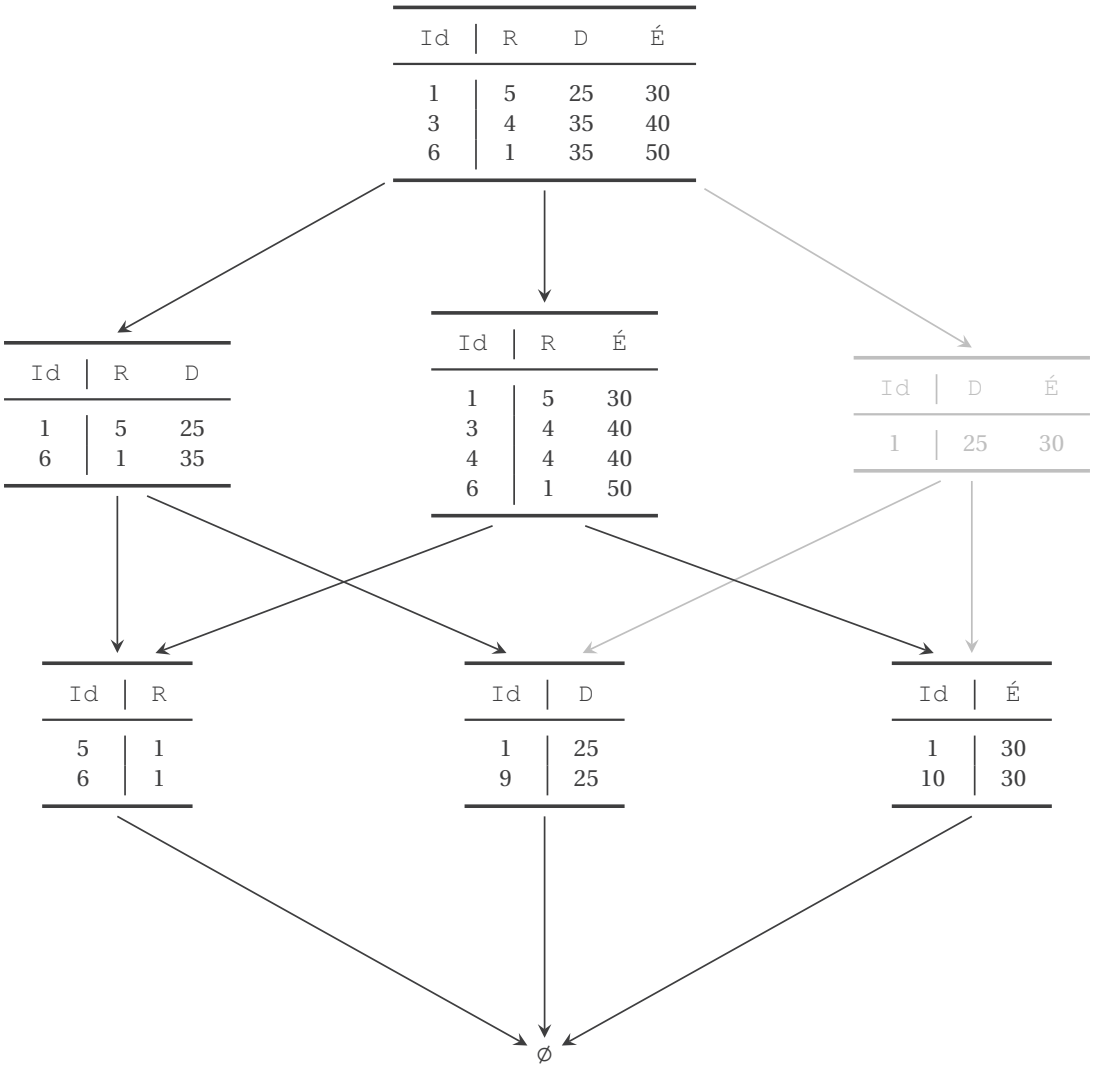


FIGURE 6.13. – Représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_1$

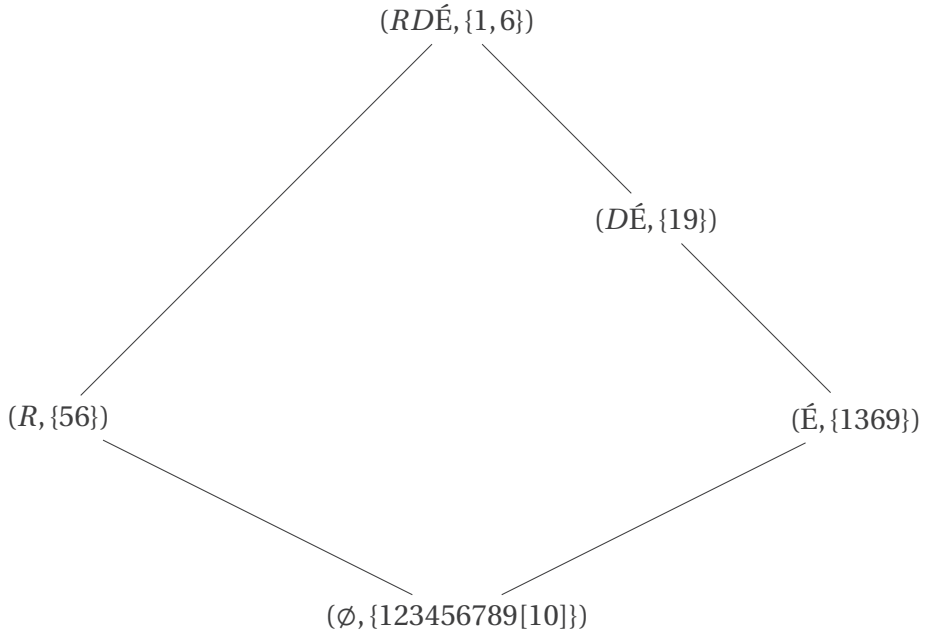


FIGURE 6.14. – Diagramme de Hasse du treillis des concepts Skylines des experts

6.3.4.2. Matérialisation partielle du Skycube

Cette sous-sous-section propose le treillis des concepts Skylines comme une matérialisation partielle des Skycubes. L'idée sous-jacente, afin d'obtenir une représentation réduite, est d'éliminer les cuboïdes les plus efficacement reconstructibles.

Définition 6.14 (Condition de non accord) - Soit r une relation et C un ensemble de critères. La condition de non accord, ou $CNA_C(r)$ est vérifiée quand :

$$\nexists t_i,\ t_j \in r \text{ tels que } t_i[C] = t_j[C] \text{ avec } i \neq j,\ C \subseteq \mathscr{C}$$

Lorsque $CNA_C(r)$ est vérifiée, $CNA_X(r)$ l'est aussi avec X un sur-ensemble de C.

Cette condition de non accord est une version « limitée » de la condition de valeurs distinctes (YUAN, LIN, Q. LIU et al. 2005) puisqu'elle s'applique sur les projections et non pas sur les valeurs individuelles de chacun des critères.

Définition 6.15 (Dominance sous condition de non accord) - Soit $C \subseteq \mathscr{C}$ un ensemble de critères et r une relation. La relation de dominance $t \succ_C t$ sous l'hypothèse $CNA_C(r)$ s'exprime plus simplement :

$$\text{Soit } C \text{ tel que } CNA_C(r), \forall t_i, t_j \in r \text{ on a } t_j \succ_C t_i \Leftrightarrow \forall c \in C,\ t_j[c] \leq t_i[c] \text{ avec } i \neq j$$

Lemme 6.6 - Soit r une relation. Pour tout ensemble de critères $C \subseteq \mathscr{C}$ vérifiant l'hypothèse de non accord $CNA_C(r)$, on a $S_C(r) \subseteq CNA_{C \cup c_0}(r)$ avec $c_0 \in \mathscr{C} \setminus C$.

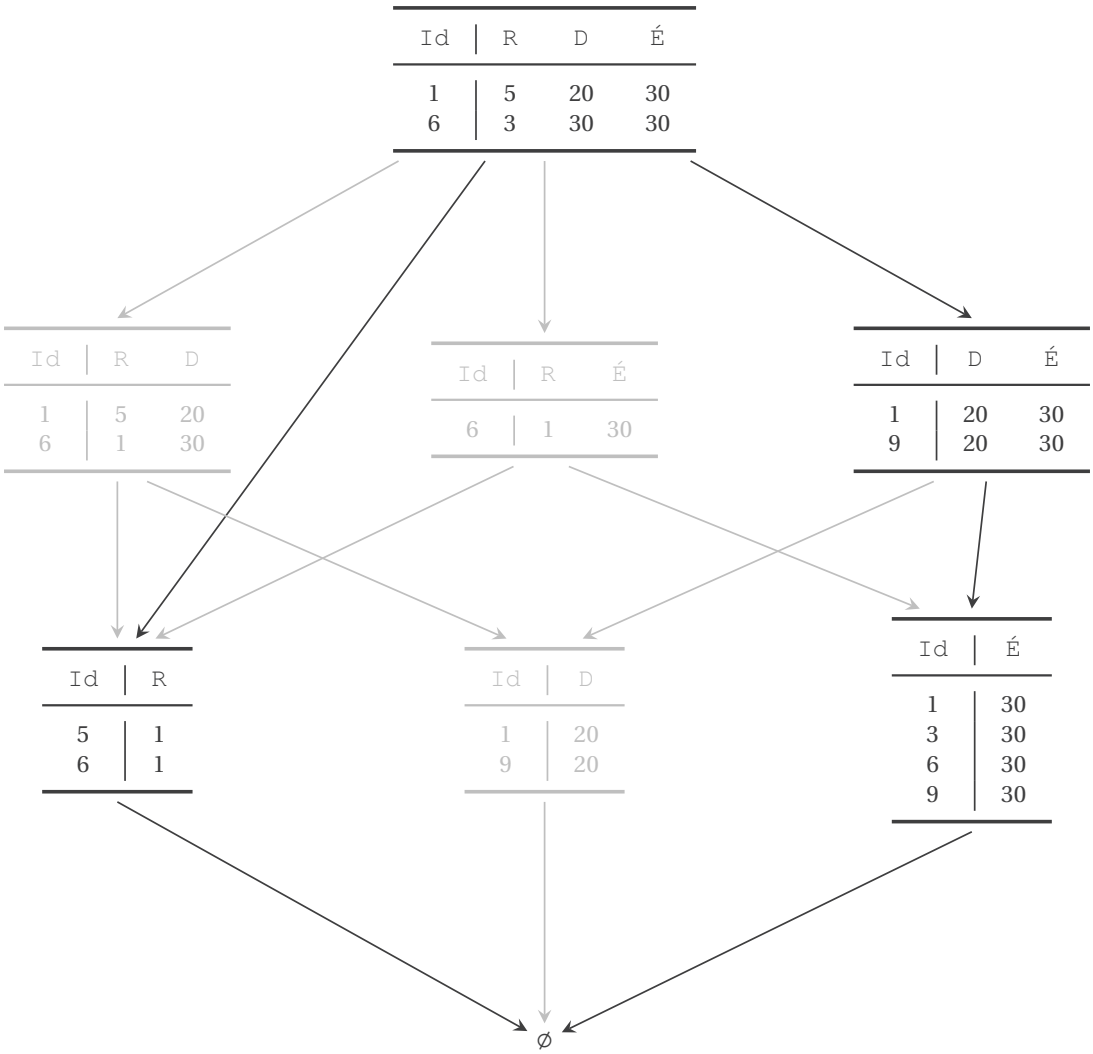


FIGURE 6.15. – Représentation réduite du treillis du Skycube de la relation $\texttt{Pokémon}_2$

Démonstration. Sous l'hypothèse $CNA_C(r)$, on a : $t_i \in S_C(r) \Rightarrow \forall\, t_j \in r$ avec $i \neq j$, on a $t_j \not\succ_C t_i \Rightarrow \forall\, t_j \in r$ avec $i \neq j$, $\exists c \in C, t_j[c] > t_i[c]$ (cf. définition 6.15) $\Rightarrow \forall\, t_j \in r$ avec $i \neq j, \exists c \in C \cup c_0, t_j[c] > t_i[c] \Rightarrow \forall\, t_j \in r$ avec $i \neq j$, on a $t_j \not\succ_{C\cup c_0} t_i \Rightarrow t_i \in S_{C\cup c_0}(r)$. Sous l'hypothèse $CNA_C(r)$, nous avons donc $S_C(r) \subseteq S_{C\cup c_0}(r)$. □

Le contre exemple suivant montre que la propriété réciproque n'est pas vérifiée.

Exemple 6.15 - Soit $r = \{t_1, t_2\}$ une relation (avec $t_1 = (0,1)$ et $t_2 = (1,0)$) et $\mathscr{C} = \{A, B\}$ l'ensemble de ses critères. La condition $CNA_A(r)$ est bien vérifiée. On a $t_2 \notin S_A(r)$ car $t_1 \succ_A t_2$ alors que $t_2 \in S_{A\cup B}(r)$.

Corollaire 6.7 - Soit C tel que $CNA_C(r)$ est vérifiée. Par définition, pour tout X ($C \subseteq X$) vérifiant la condition $CNA_X(r)$, on a $S_C(r) \subseteq S_X(r)$.

Cette propriété est satisfaite pour tout sur-ensemble de C jusqu'à l'ensemble de tous les critères $\mathscr{C}$.

Théorème 6.8 (Théorème fondamental) **-** Soit r une relation, C un ensemble de critères et $h(C)$ sa fermeture. Alors :

$$\forall C \subseteq \mathscr{C}, S_C(r) \subseteq S_{h(C)}(r)$$

Démonstration. Par définition, $t_i \in S_C(r) \Leftrightarrow \exists [t] \in \Pi\text{-}S_C(\pi_C), i \in [t]$. On cherche donc à montrer que $\Pi\text{-}S_C(\pi_C) \subseteq \Pi\text{-}S_C(\pi_{h(C)})$. On sait que $\forall X, C \subseteq X \subseteq h(C), \pi_X = \pi_C$. Soit $E = \{t_i \in r \mid i \in reps(\pi_C)\}$ l'ensemble des tuples représentants de cette partition. On ne peut se baser sur E pour le calcul des Skylines sur X que dans l'intervalle $C \subseteq X \subseteq h(C)$, autrement les classes d'équivalences n'étant plus égales il est impossible de garantir une reconstruction correcte du Skyline total à partir des représentants des classes d'équivalences. Aussi, la condition $CNA_X(E)$ est vérifiée car chaque tuple représentant est distinguable d'un autre donc d'après le lemme 6.6 on a $S_C(E) \subseteq S_{h(C)}(E)$ d'où $S_C(r) \subseteq S_{h(C)}(r)$. □

Exemple 6.16 - Dans la relation $\texttt{Pokémon}_2$, nous avons $S_{RD}(\texttt{Pokémon}_2) = \{t_1, t_6\}$. De plus, nous avons $h(RD) = RD\text{É}$ et $S_{RD\text{É}}(\texttt{Pokémon}_2) = \{t_1, t_6\}$. $S_{RD}(\texttt{Pokémon}_2) \subseteq S_{h(RD)}(\texttt{Pokémon}_2)$ est donc bien vérifiée. Aussi, on a $S_R(\texttt{Pokémon}_2) = \{t_5, t_6\}$ avec $h(R) = R$. On note donc la non inclusion $S_R(\texttt{Pokémon}_2) \not\subseteq S_{RD}(\texttt{Pokémon}_2)$.

Le théorème précédent montre une inclusion entre certains cuboïdes. Plus exactement, pour tout ensemble de critères, il y a une chaîne d'inclusions allant de ses générateurs minimaux jusqu'à sa fermeture. Cela implique que l'on peut calculer un cuboïde à partir d'un autre qui le contient. Ainsi au lieu d'utiliser la relation complète, on ne considère qu'un sous-ensemble restreint de celle-ci. Les concepts Skylines représentent les plus grands cuboïdes (selon l'inclusion) permettant d'en calculer d'autres. Ainsi en ne conservant que les concepts Skylines (*i.e.* non matérialisation des cuboïdes non fermés), on peut reconstruire rapidement les cuboïdes manquants simplement en trouvant le fermé à partir duquel ils peuvent être recalculés.

De plus grâce à la notion de classe d'équivalence des concepts Skylines, on évite un grand nombre de comparaisons inutiles. Les éléments indistinguables n'étant considérés que par groupes, la complexité du calcul ne dépend plus du nombre de tuples mais du nombre de groupes (classes d'équivalence).

Le treillis des concepts Skylines constitue donc une matérialisation partielle du Skycube, où l'information non matérialisée est efficacement calculable.

6.3.5. Évaluations expérimentales

Dans cette sous-section, nous présentons la validations de l'approche théorique de la matérialisation partielle du Skycube par le biais d'évaluations expérimentales sur des bases de données universitaires synthétiques générées de dix dimensions et d'un million de tuples [13]. Ces dernières sont menées selon un double objectif : d'une part, mesurer la réduction de la taille et, d'autre part, évaluer le temps de réponse des requêtes Skylines.

6.3.5.1. Réductions de la taille

Dans cette expérimentation, c'est à la fois la taille du Skycube et celle du treillis des concepts Skylines qui sont mesurés afin de les comparer. La figure 6.16 montre les résultats obtenus lorsque la cardinalité des critères varie de 100 à 10000. Plus la cardinalité augmente, plus la représentation est intéressante quant à l'espace de stockage qu'elle nécessite pour être matérialisée, alors que le Skycube devient de plus en plus volumineux.

6.3.5.2. Temps de réponse des requêtes

Dans cette expérimentation, nous voulons comparer le temps de réponse d'une requête Skyline utilisant l'algorithme Sort-Filter-Skyline (SFS) représenté par l'algorithme 11 (CHOMICKI, GODFREY, GRYZ et al. 2005), lorsqu'il n'y pas de matérialisation partielle du Skycube d'une part, et avec le treillis de concept Skyline d'autre part.

L'algorithme Sort-Filter-Skyline (SFS) nécessite une fonction monotone ou anti-monotone (par exemple SUM) appliquée à l'ensemble de tuples. En utilisant cette fonction, un tri topologique est effectué sur SUM(t) de manière croissante. Soit t et t' deux tuples d'une relation r avec seulement des valeurs positives. Supposons que tous les critères doivent être minimisés. Si SUM(t') > SUM(t) alors t' ne domine pas t.

Avec la matérialisation partielle du Skycube, deux cas sont à considérer. Dans le cas le plus simple, le résultat de la requête est inclus dans la matérialisation partielle du Skycube et, par conséquent, le coût de la requête est également minimal. Dans le second, le résultat de la requête doit être calculé à partir du Skycuboïde fermé associé en utilisant le théorème 6.8. La qualité de l'évaluation est améliorée et permet d'obtenir des résultats significatifs car chaque point de la courbe de la figure 6.17 représente

13. Le générateur de données synthétiques est celui mis à disposition gracieusement par l'Université de l'Illinois à Urbana-Champaign.

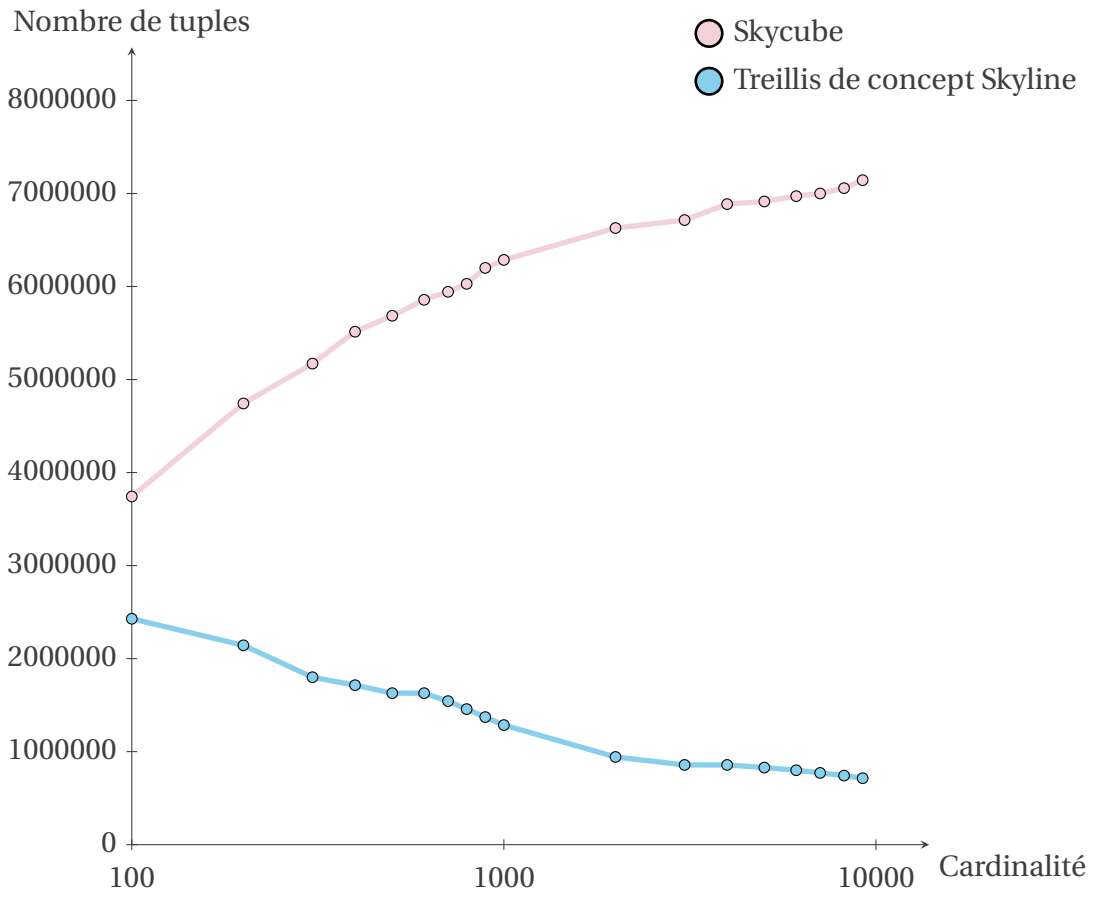


FIGURE 6.16. – Réduction de la taille

Algorithme 11 Algorithme Sort-Filter-Skyline (SFS)

Entrée :
La relation r triée de manière croissante selon SUM(t).
Le premier tuple t de r.
L'ensemble de critères $\mathscr{C}$.
Sortie :
L'ensemble de tuples du Skyline $S_{\mathscr{C}}(r)$.

```
01: soit S_C(r) := t;
02: pour tout t' ∈ S_C(r)\t faire
03:   si ∄t'' ∈ S_C(r), t'' ≺ t' alors
04:     S_C(r) := S_C(r) ∪ t';
05:   fin si
06:   retourner S_C(r)
07: fin pour
```

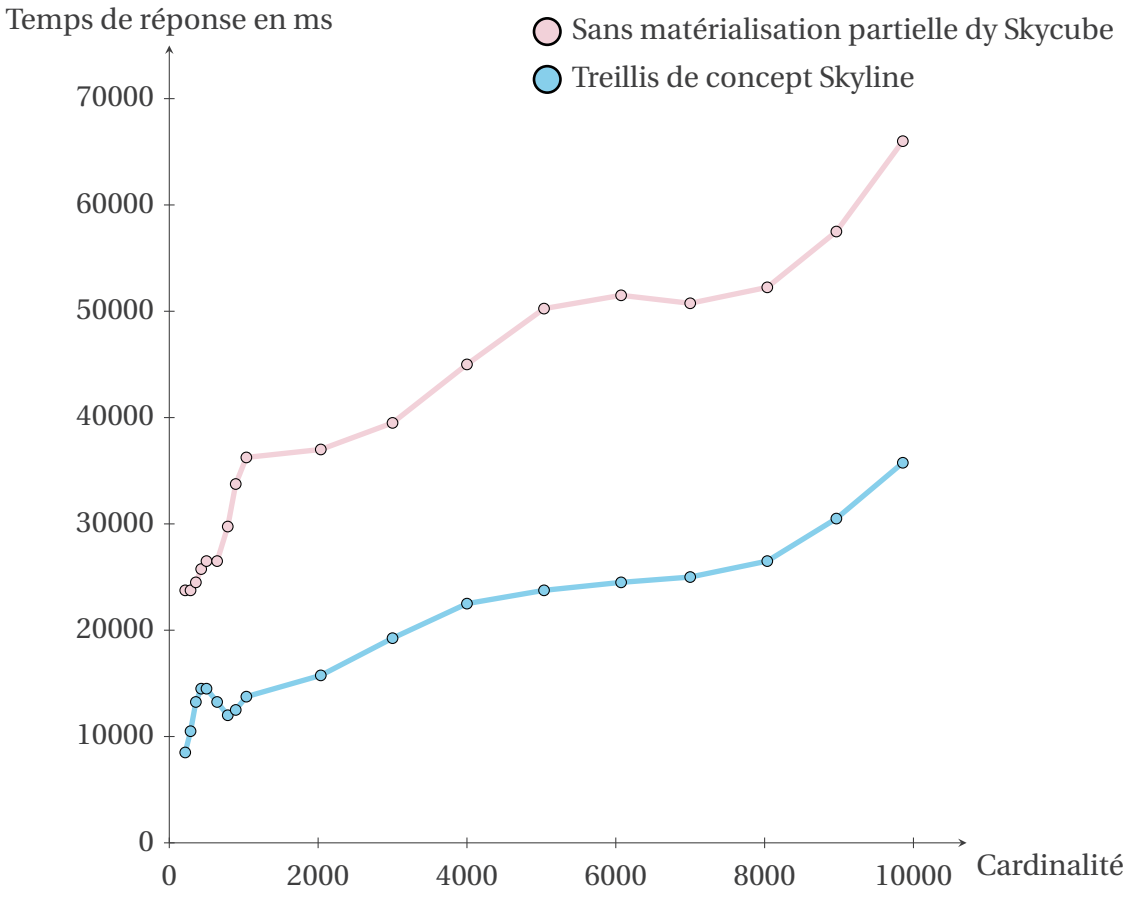


FIGURE 6.17. – Temps de réponse des requêtes

le temps d'exécution moyen de centaines de requêtes alors que la cardinalité des critères varie de 100 à 10000. En outre, pour simuler les requêtes des utilisateurs, le nombre de critères pour chacune d'entre elles est déterminé grâce à une distribution binomiale centrée sur le nombre de critères possibles divisé par deux, puis la combinaison de critères est choisi de manière aléatoire. De cette façon, les requêtes traitées prennent souvent en compte plusieurs critères, sans qu'il n'y en est trop pour autant, de sorte qu'elles aient plus de sens que si elles étaient choisies au hasard. Les résultats montrent que le treillis des concepts Skylines est bien efficace car il peut répondre rapidement à n'importe quelle requête Skyline et que son facteur de gain est constant.

6.3.6. Comparaison avec les travaux antérieurs

L'opérateur Skyline (BÖRZSÖNYI, KOSSMANN et STOCKER 2001) s'intéresse à trouver les éléments les plus intéressants dans un contexte base de données. Il tire ses origines du *maximal vector problem* (BARNDORFF-NIELSEN et SOBEL 1966; BENTLEY, KUNG, SCHKOLNICK et al. 1978). Cependant les particularités du contexte font que des algorithmes spécifiques ont été développés, sans index ou avec des index, classiques ou spatiaux, coûteux à maintenir (cf. la sous-sous-section 2.4.2.6).

Également, des efforts ont été déployés afin de paralléliser le calcul de Skyline (BØGH, CHESTER et ASSENT 2015; ZEKRI 2017; W. WANG, J. ZHANG, SUN et al. 2019) et tout particulièrement la remarquable solution SkyCell (Chuanwen LI, GU, QI et al. 2021), un algorithme de calcul parallèle Skyline basé sur le parcours spatial d'une grille discrétisée.

Le Skyline étant un outil d'aide à la décision, l'utilisateur va généralement calculer plusieurs Skylines avant de trouver celui qui l'intéresse vraiment. Pour répondre à cette problématique, nous rappelons que le concept de Skycube (YUAN, LIN, Q. LIU et al. 2005; PEI, JIN, ESTER et al. 2005) a été proposé. L'idée générale étant de pré-calculer tous les Skylines, il est impératif de restreindre autant que possible le coût de stockage du résultat.

Une première approche de réduction a été proposée par PEI, YUAN, LIN et al. 2006. Son objectif est de résoudre le problème de l'appartenance d'un objet donné à différents cuboïdes du Skycube, ou Skycuboïdes. Sa solution consiste à proposer une représentation du Skycube fondée sur l'analyse formelle des concepts où chaque nœud correspond à un couple composé d'une classe d'équivalence (ensemble d'objets) et des sous-espaces dans lesquels elle est Skyline. Le principal défaut de cette approche orientée valeur est que le nombre de nœuds est borné par la cardinalité du treillis des parties des tuples ($|\mathcal{P}(Tid(r))|$). Les objets Skylines sont considérés avant tout, ce qui fait que pour reconstruire un Skycuboïde il faut parcourir un grand nombre de nœuds du treillis. Dans les approches orientées attribut comme celle présentée ici, le nombre de nœuds est bien plus restreint car il est borné par $|\mathcal{P}(\mathscr{C})|$.

Aux autres approches remarquables (cf. la sous-sous-section 2.4.3.3), ajoutons la proposition de XIA et D. ZHANG 2006. Elle est orientée cuboïde (orientée attribut). L'objectif ici est de proposer une réduction très importante du Skycube en éliminant les éléments considérés comme redondants des différents cuboïdes. Son inconvénient

est que la reconstruction d'un Skycuboïde est délicate et relativement coûteuse car il faut parcourir une structure de données pour retrouver les éléments redondants. De plus, contrairement à l'approche de PEI, YUAN, LIN et al. 2006 ; PEI, A. W.-C. FU, LIN et al. 2007 et à celle présentée, celle-ci n'est pas fondée sur un support théorique aussi solide que l'analyse des concepts formels. Cependant, l'un des principaux intérêts de cette approche en plus de la réduction importante du coût de stockage est l'efficacité de la mise à jour des données.

L'approche présentée ayant pour objectif de reconstruire les Skycuboïdes à moindre coût, elle se comporte idéalement dans ce cas-là. Malgré cette orientation ciblée, elle représente un bon compromis que ce soit pour la mise à jour ou pour la réduction de l'espace de stockage.

6.4. Calcul du Skycube émergent

Comme pour le cube de données émergent (cf. sous-section 3.4.2 p. 98), il n'existe à notre connaissance pour l'instant aucun moyen intégré aux SGBD pour calculer le Skycube émergent. Afin que le Skycube émergent soit un opérateur directement exploitable par le décideur, les algorithmes de calcul que nous présentons se doivent d'être intégrables directement dans les SGBD. Avec une telle approche relationnelle, il est possible de tirer pleinement partie des outils d'analyse ROLAP existants. Le Skycube émergent n'est ainsi qu'un cube de données particulier et même, qu'un cube de données émergent particulier, et comme cela est aussi faisable avec les cubes de données traditionnels, il est possible de l'interroger, de l'explorer et d'y naviguer. Le concept de cube de données émergent est directement transposable en Skycube émergent, indépendamment de la réduction préalable des Skycubes, en prenant en entrée deux Skycubes, unicompatibles, représentés sous forme de relations que nous appellerons r_1 et r_2 comme pour le cube de données émergent (cf. définition 3.2 p. 89).

Nous rappelons qu'un tuple émergent de r_1 vers r_2 a une valeur de mesure faible dans r_1 alors qu'elle est significative dans r_2. Cependant, une valeur faible peut être aussi bien une valeur basse que haute, *idem* pour une valeur significative. Nous redéfinissons le tuple émergent (cf. définition 3.2 p. 89) en prenant en compte ces possibilités.

Définition 6.16 (Tuple émergent) **-** Un tuple $t \in CL(r_1 \cup r_2)$ est dit émergent de r_1 vers r_2 si et seulement s'il satisfait les deux contraintes C_1 et C_2 de la manière suivante :

$$\begin{cases} f_{val}(t, r_1) \geq SeuilMin_1(C_1) \\ f_{val}(t, r_2) < SeuilMin_2(C_2) \end{cases}$$

Ou :

$$\begin{cases} f_{val}(t, r_1) < SeuilMin_1(C_1) \\ f_{val}(t, r_2) \geq SeuilMin_2(C_2) \end{cases}$$

Dans un Skycube émergent, le terme critère, associé à une préférence Skyline, se confond avec celui de mesure. Dans ce document, nous emploierons l'un ou l'autre

en fonction du contexte afin de préserver du mieux possible l'usage traditionnel : dans le cadre de l'analyse multicritère, nous préfèrerons critère, mais dans un cadre plus OLAP, nous choisissons plutôt celui de mesure. Avant cette approche, le calcul du cube de données émergent était réalisé selon une seule attribut mesure dans r_1, et son homologue dans r_2, ce qui correspond à un seul critère avec une seule préférence Skyline, et cela n'a pas vraiment d'intérêt dans le cadre d'une analyse décisionnelle multicritère. Avec le calcul des Skycubes, et donc avec plusieurs préférences Skyline, y compris potentiellement hétérogènes, il devient nécessaire d'employer plusieurs attributs mesures dans r_1 et le même nombre dans r_2.

Le coût de calcul et celui d'enregistrement du Skycube de données émergent dépendent du nombre de tuples de l'ensemble des Skycuboïdes préservés, soit des critères, des préférences Skyline, et des doubles contraintes d'émergence correspondantes, fixée par le décideur, comme pour le cube de données émergent traditionnel. Le nombre de tuples émergents est borné par la taille de l'espace de recherche, autrement dit l'ensemble des tuples possibles. Plus les doubles contraintes d'émergence sont fortes, plus le Skycube de données émergent associé est réduit. Le calibrage des contraintes permettant d'obtenir un Skycube de données émergent d'un volume exploitable est important, d'autant plus que la cardinalité des Skycuboïdes peut être relativement élevée, en fonction des critères exprimés par les mesures, bien qu'elle soit significativement réduite du fait de la matérialisation partielle des Skycubes vue précédemment. C'est ainsi que le nombre et la qualité des mesures influencent indirectement le coût de calcul et celui d'enregistrement du Skycube émergent, mais ils l'influencent aussi directement en ajoutant des dimensions, et les valeurs de tuples associées, à la relation d'entrée.

La principale différence avec un cube de données émergent, c'est qu'un tuple du Skycube n'est émergent qu'à la condition qu'il le soit pour toutes les mesures des Skycubes. En conséquence, nous introduisons le tuple émergent mesuré.

Définition 6.17 (Tuple émergent mesuré) **-** Soit r_1 et r_2 deux relations unicompatibles, $t \in CL(r_1 \cup r_2)$ un tuple et $f^{\mathcal{M}}$ une fonction agrégative correspondant à la mesure $\mathcal{M}$. t est dit tuple émergent mesuré si et seulement s'il satisfait les deux contraintes $C_1^{\mathcal{M}}$ et $C_2^{\mathcal{M}}$ de la manière suivante :

$$\begin{cases} f_{val}^{\mathcal{M}}(t, r_1) \geq SeuilMin_1(C_1^{\mathcal{M}}) \\ f_{val}^{\mathcal{M}}(t, r_2) < SeuilMin_2(C_2^{\mathcal{M}}) \end{cases}$$

Ou :

$$\begin{cases} f_{val}^{\mathcal{M}}(t, r_1) < SeuilMin_1(C_1^{\mathcal{M}}) \\ f_{val}^{\mathcal{M}}(t, r_2) \geq SeuilMin_2(C_2^{\mathcal{M}}) \end{cases}$$

Traditionnellement, une mesure dont les valeurs n'évolueraient jamais d'une relation à une autre n'a pas grand intérêt dans le calcul d'un cube de données émergent, et serait ignorée en étant supprimée des relations d'entrée. Cependant, dans le cadre du Skycube émergent, nous avons vu qu'une telle mesure pourrait être un critère utile, par ailleurs, lors du calcul des Skycubes, et donc héritée par la suite. Ainsi, nous

définissons une mesure invariante.

Définition 6.18 (Mesure invariante) - Soit r_1 et r_2 deux relations unicompatibles et $f^{\mathcal{M}}$ une fonction agrégative correspondant à la mesure $\mathcal{M}$. $\mathcal{M}$ est appelée mesure invariante si et seulement si : $\forall t \in CL(r_1 \cup r_2), f^{\mathcal{M}}_{val}(t, r_1) = f^{\mathcal{M}}_{val}(t, r_2)$.

Préalablement au calcul même du Skycube émergent, un travail de fusion des Skycuboïdes des deux Skycubes en une seule relation est nécessaire. En effet, afin de se dispenser du calcul de l'ensemble des Skycuboïdes deux à deux, nous effectuons une « fusion » des deux relation opérandes (DIOP, GIACOMETTI, LAURENT et al. 2002). Le résultat de cette fusion est une nouvelle relation r dont le schéma est le suivant : $\mathscr{R} = \mathscr{D} \cup \mathcal{M}_1 \cup \mathcal{M}'_1 \cup \mathcal{M}_2 \cup \mathcal{M}'_2 \ldots \cup \mathcal{M}_n \cup \mathcal{M}'_n$ où $\mathcal{M}_1$, $\mathcal{M}_2$, …, $\mathcal{M}_n$ sont les attributs mesures de r_1 et $\mathcal{M}'_1, \mathcal{M}'_2, \ldots, \mathcal{M}'_n$ ceux de r_2. En nous inspirant de la valeur choisie pour les attributs hors partitionnement de l'OLAP (cf. sous-section 2.2.4), nous proposons que les attributs ne faisant pas partie des attributs des Skycuboïdes prennent la valeur *ALL*. Nous rappelons que cette valeur (GRAY, CHAUDHURI, BOSWORTH et al. 1997) est une généralisation de toutes les valeurs du domaine d'un attribut. Autrement dit, les Skycuboïdes partagent tous le même schéma et peuvent être regroupés dans une seule et même relation. Subséquemment, la fusion va engendrer nombre de valeurs *ALL* dans la relation, correspondant, pour chaque tuple, à autant d'attributs absents dans son Skycuboïde initial. Contrairement au cube de données émergent traditionnel, l'opération de dédoublonnage est inutile car réalisée naturellement, et préalablement, par le calcul des Skycubes. Cette étape est donc économisée, et une relation fusionnée, et non plus fusionnée dédoublonnée (cf. tableau 6.4), suffit.

Exemple 6.17 - La relation fusionnée est donnée dans le tableau 6.5 (avec `R` pour `Rareté`, $\texttt{D}_1$ pour $\texttt{Durée}_1$, $\texttt{É}_1$ pour $\texttt{Échec}_1$, $\texttt{D}_2$ pour $\texttt{Durée}_2$ et $\texttt{É}_2$ pour $\texttt{Échec}_2$). Les séparations présentes dans le tableau délimitent les tuples issus des Skycuboïdes unicompatibles des deux Skycubes d'entrée.

Dans l'objectif de pouvoir quantifier clairement un renversement de tendances comportant plusieurs mesures, nous introduisons le calcul du taux d'emergence mesuré.

Définition 6.19 (Taux d'emergence mesuré) - Soit r_1 et r_2 deux relations unicompatibles, $t \in CL(r_1 \cup r_2)$ un tuple et $f^{\mathcal{M}}$ une fonction agrégative correspondant à la mesure $\mathcal{M}$. Le taux d'émergence mesuré de t de r_1 vers r_2 par rapport à la mesure $\mathcal{M}$, noté $ER_{\mathcal{M}}(t)$, est défini par :

$$ER_{\mathcal{M}_1}(t) = \begin{cases} 0 \text{ si } f^{\mathcal{M}}_{val}(t, r_1) = 0 \text{ et } f^{\mathcal{M}}_{val}(t, r_2) = 0 \\ \infty \text{ si } f^{\mathcal{M}}_{val}(t, r_1) = 0 \text{ et } f^{\mathcal{M}}_{val}(t, r_2) \neq 0 \\ \dfrac{f^{\mathcal{M}}_{val}(t, r_2)}{f^{\mathcal{M}}_{val}(t, r_1)} \text{ sinon.} \end{cases}$$

Un Skycube émergent possède alors autant de taux d'emergence mesurés que d'attributs mesures.

TABLEAU 6.5. – Relation fusionnée $\texttt{Pokémon}_{Fus}$

RowId	Tier	Joueur	Adversaire	...	R [a]	D_1 [b]	$É_1$ [c]	D_2 [d]	$É_2$ [e]
1	*UU*	*A*	*D*	...	5	25	30	20	30
2	*OU*	*B*	*F*	...	4	35	40	*ALL*	*ALL*
3	*OU*	*C*	*B*	...	1	35	50	30	30
4	*UU*	*A*	*D*	...	5	25	*ALL*	*ALL*	*ALL*
5	*OU*	*C*	*B*	...	1	35	*ALL*	*ALL*	*ALL*
6	*UU*	*A*	*D*	...	5	*ALL*	30	*ALL*	*ALL*
7	*OU*	*B*	*F*	...	4	*ALL*	40	*ALL*	*ALL*
8	*OU*	*B*	*A*	...	4	*ALL*	40	*ALL*	*ALL*
9	*OU*	*C*	*B*	...	1	*ALL*	50	*ALL*	*ALL*
10	*UU*	*A*	*D*	...	*ALL*	*ALL*	*ALL*	20	30
11	*UU*	*E*	*D*	...	*ALL*	*ALL*	*ALL*	20	30
12	*OU*	*C*	*A*	...	1	*ALL*	*ALL*	*ALL*	*ALL*
13	*OU*	*C*	*B*	...	1	*ALL*	*ALL*	*ALL*	*ALL*
14	*UU*	*A*	*D*	...	*ALL*	25	*ALL*	*ALL*	*ALL*
15	*UU*	*E*	*D*	...	*ALL*	25	*ALL*	*ALL*	*ALL*
16	*UU*	*A*	*D*	...	*ALL*	*ALL*	30	*ALL*	30
17	*OU*	*B*	*F*	...	*ALL*	*ALL*	*ALL*	*ALL*	30
18	*OU*	*C*	*B*	...	*ALL*	*ALL*	*ALL*	*ALL*	30
19	*UU*	*E*	*D*	...	*ALL*	*ALL*	*ALL*	*ALL*	30
20	*UU*	*E*	*E*	...	*ALL*	*ALL*	30	*ALL*	*ALL*

a. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de R r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

b. En nombre de tours de combat au total.

c. En pourcentage.

d. *idem* D_1.

e. *idem* $É_1$.

Exemple 6.18 - Le tableau 6.6 donne les tuples émergents mesurés de la relation $\texttt{Pokémon}_{Fus}$ selon Échec avec les seuils $SeuilMin_1 = 45$, à dépasser, et $SeuilMin_2 = 45$, à ne pas atteindre, et présentés par Skycuboïde. Les tuples émergents mesurés, visibles après la dernière séparation du tableau, correspondent au Skycuboïde n'ayant que le critère Échec, et ne sont pas conservés par le Skycube émergent (cf. tableau 6.8) car ces tuples n'ont que la valeur *ALL* pour la mesure Durée.

De la relation fusionnée, nous obtenons la relation fusionnée abrégée en :
— supprimant les mesures invariantes et les valeurs des tuples correspondantes ;
— supprimant les tuples issus de Skycuboïdes n'ayant pas au moins comme critères l'ensemble de ceux du second Skycube (susceptible de créer un renversement de tendances dans un contexte avec plusieurs mesures).

En effet, les mesures invariantes ne sont pas utiles lors du calcul du Skycube émergent, et peuvent donc être retirées. Les tuples issus de Skycuboïdes n'ayant pas comme critères l'ensemble des mesures de la relation fusionnée, même dans le cas où ils sont tuples émergents mesurés, ne seront pas retenu comme tuples émergents car ils ne le sont pas pour toutes les mesures des Skycubes.

Exemple 6.19 - La relation abrégée de la relation fusionnée (cf. tableau 6.5) est donnée dans le tableau 6.7 (avec $\texttt{D}_1$ pour $\texttt{Durée}_1$, $\texttt{É}_1$ pour $\texttt{Échec}_1$, $\texttt{D}_2$ pour $\texttt{Durée}_2$ et $\texttt{É}_2$) pour $\texttt{Échec}_2$). Les séparations présentent dans le tableau de la relation fusionnée sont préservées par commodité. Cette relation qui sert d'entrée aux algorithmes de la plateforme IDEA.

Exemple 6.20 - Le tableau 6.8 donne le cube de données émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$ avec les seuils suivants :
— Mesure Durée : $SeuilMin_1 = 35$ et $SeuilMin_2 = 35$;
— Mesure Échec : $SeuilMin_1 = 45$ et $SeuilMin_2 = 45$;

Nous choisissons la fonction agrégative AVG (Pei, Han *et* Lakshmanan *2004*) pour l'exemple. Les séparations présentent dans le tableau de la relation $\texttt{Pokémon}^{+}_{Fus}$ sont préservées par commodité.

Le Skycube émergent est calculable grâce à l'algorithme E-IDEA, à partir du moment où le format d'entrée est respecté, de manière transparente comme n'importe quel cube de données émergent, à ceci prêt que l'algorithme doit être appelé pour chaque attribut mesure dans son implémentation actuelle.

6.5. Skycube fermé émergent

L'intérêt d'une représentation réduite du cube de données émergent, avec notamment le cube de données émergent fermé, se retrouve également naturellement concernant le Skycube fermé émergent.

Comme nous l'avons vu (cf. sous-section 3.3.1 p. 93), bien que très utiles et offrant une représentant très réduite, les bordures L et U occasionnent une perte d'infor-

TABLEAU 6.6. – Tuples émergents mesurés de la relation $\texttt{Pokémon}_{Fus}$ selon $\texttt{Échec}$

Tuple émergent mesuré	$ER_{\texttt{Échec}}$
$(ALL, ALL, B, \ldots)$	1.67
$(ALL, C, ALL, \ldots)$	1.67
$(OU, ALL, ALL, \ldots)$	3
$(ALL, C, B, \ldots)$	1.67
$(OU, ALL, B, \ldots)$	1.67
$(OU, C, ALL, \ldots)$	1.67
$(OU, C, B, \ldots)$	1.67
$(ALL, ALL, D, \ldots)$	∞
$(ALL, A, ALL, \ldots)$	∞
$(ALL, E, ALL, \ldots)$	∞
$(ALL, A, D, \ldots)$	∞
$(ALL, E, D, \ldots)$	∞
$(UU, ALL, ALL, \ldots)$	∞
$(UU, ALL, D, \ldots)$	∞
$(UU, A, ALL, \ldots)$	∞
$(UU, E, ALL, \ldots)$	∞
$(UU, A, D, \ldots)$	∞
$(UU, E, D, \ldots)$	∞
$(ALL, ALL, F, \ldots)$	∞
$(ALL, ALL, B, \ldots)$	∞
$(ALL, B, ALL, \ldots)$	∞
$(ALL, C, ALL, \ldots)$	∞
$(OU, ALL, ALL, \ldots)$	∞
$(ALL, B, F, \ldots)$	∞
$(ALL, C, B, \ldots)$	∞
$(ALL, E, D, \ldots)$	∞
$(OU, ALL, F, \ldots)$	∞
$(OU, ALL, B, \ldots)$	∞
$(OU, B, ALL, \ldots)$	∞
$(OU, C, ALL, \ldots)$	∞
$(OU, B, F, \ldots)$	∞
$(OU, C, B, \ldots)$	∞
$(UU, E, D, \ldots)$	∞

TABLEAU 6.7. – Relation fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$

RowId	Tier	Joueur	Adversaire	...	$\texttt{D}_1$ [a]	$\texttt{É}_1$ [b]	$\texttt{D}_2$ [c]	$\texttt{É}_2$ [d]
1	*UU*	*A*	*D*	...	25	30	20	30
2	*OU*	*B*	*F*	...	35	40	*ALL*	*ALL*
3	*OU*	*C*	*B*	...	35	50	30	30
4	*UU*	*A*	*D*	...	*ALL*	*ALL*	20	30
5	*UU*	*E*	*D*	...	*ALL*	*ALL*	20	30

a. En nombre de tours de combat au total.
b. En pourcentage.
c. *idem* $\texttt{D}_1$.
d. *idem* $\texttt{É}_1$.

TABLEAU 6.8. – Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$

Tuple émergent	$ER_{\texttt{Durée}}$	$ER_{\texttt{Échec}}$
$(ALL, ALL, B, \ldots)$	1.17	1.67
$(ALL, C, ALL, \ldots)$	1.17	1.67
$(OU, ALL, ALL, \ldots)$	2.33	3
$(ALL, C, B, \ldots)$	1.17	1.67
$(OU, ALL, B, \ldots)$	1.17	1.67
$(OU, C, ALL, \ldots)$	1.17	1.67
$(OU, C, B, \ldots)$	1.17	1.67
$(ALL, ALL, F, \ldots)$	∞	∞
$(ALL, ALL, B, \ldots)$	∞	∞
$(ALL, B, ALL, \ldots)$	∞	∞
$(ALL, C, ALL, \ldots)$	∞	∞
$(OU, ALL, ALL, \ldots)$	∞	∞
$(ALL, B, F, \ldots)$	∞	∞
$(ALL, C, B, \ldots)$	∞	∞
$(ALL, E, D, \ldots)$	∞	∞
$(OU, ALL, F, \ldots)$	∞	∞
$(OU, ALL, B, \ldots)$	∞	∞
$(OU, B, ALL, \ldots)$	∞	∞
$(OU, C, ALL, \ldots)$	∞	∞
$(OU, B, F, \ldots)$	∞	∞
$(OU, C, B, \ldots)$	∞	∞
$(UU, E, D, \ldots)$	∞	∞

mation. En effet, les valeurs des attributs mesures, agrégées, ne peuvent plus être retrouvées. Si cette perte est acceptable pour le décideur, alors l'usage des bordures comme représentation est tout indiqué, de même donc que l'algorithme F-IDEA.

Exemple 6.21 - Avec la relation exemple fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$, le tableau 6.9 donne la bordure L pour le Skycube émergent, en conservant une valeur similaire pour les seuils (pour Durée $SeuilMin_1 = 35, SeuilMin_2 = 35$ et pour Échec $SeuilMin_1 = 45, SeuilMin_2 = 45$). Les séparations du tableau de la relation $\texttt{Pokémon}^{+}_{Fus}$ sont préservées par commodité.

TABLEAU 6.9. – Bordure L du Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$

L	$(ALL, ALL, B, \ldots)$
	$(ALL, C, ALL, \ldots)$
	$(OU, ALL, ALL, \ldots)$
	$(ALL, ALL, F, \ldots)$
	$(ALL, ALL, B, \ldots)$
	$(ALL, B, ALL, \ldots)$
	$(ALL, C, ALL, \ldots)$
	$(OU, ALL, ALL, \ldots)$
	$(ALL, E, D, \ldots)$

En revanche, si le décideur souhaite pouvoir répondre à n'importe quelle requête OLAP, et pouvoir retrouver les valeurs des mesures, il lui faut s'orienter vers la représentation réduite du Skycube fermé émergent, concept également directement transposé de celui du cube fermé émergent, puisque nous le rappelons, à partir du moment où le format d'entrée est respecté, l'usage est transparent pour les algorithmes de la plateforme IDEA. Plus exactement, cette représentation comprend l'ensemble des tuples fermés émergents du Skycube fusionné auquel il est possible d'ajouter toute l'information nécessaire permettant de garantir une représentation sans perte, reproduite par la bordure L, d'où la transposition du L-cube de données fermé émergent traditionnel (cf. sous-section 3.5.1 p. 110) en L-Skycube fermé émergent. Côté calcul, c'est l'algorithme $\mathbb{C}$-IDEA qui est employé pour générer cette représentation réduite.

Exemple 6.22 - Avec la relation exemple fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$, le tableau 6.10 donne les tuples fermés émergents pour le Skycube émergent, en conservant une valeur similaire pour les seuils (pour Durée $SeuilMin_1 = 35, SeuilMin_2 = 35$ et pour Échec $SeuilMin_1 = 45, SeuilMin_2 = 45$). Les séparations du tableau de la relation $\texttt{Pokémon}^{+}_{Fus}$ sont préservées par commodité.

Exemple 6.23 - Avec la relation exemple fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$, le tableau 6.11 donne le L-Skycube fermé émergent, en conservant une valeur similaire pour les seuils (pour Durée $SeuilMin_1 = 35, SeuilMin_2 = 35$ et pour Échec $SeuilMin_1 =$

TABLEAU 6.10. – Tuples fermés émergents de la relation $\texttt{Pokémon}^{+}_{Fus}$

Tuple fermé émergent	$ER_{\texttt{Durée}}$	$ER_{\texttt{Échec}}$
$(OU, ALL, ALL, \ldots)$	2.33	3
$(OU, C, B, \ldots)$	1.17	1.67
$(OU, ALL, ALL, \ldots)$	∞	∞
$(OU, B, F, \ldots)$	∞	∞
$(OU, C, B, \ldots)$	∞	∞
$(UU, E, D, \ldots)$	∞	∞

$45, SeuilMin_2 = 45$). Les séparations du tableau de la relation $\texttt{Pokémon}^{+}_{Fus}$ sont préservées par commodité.

Dans tous les cas de représentations réduites du Skycube émergent, il est à noter qu'elles sont d'autant plus diminuées que les Skycubes ne sont que partiellement matérialisés. Remarquons également que, en principe, le calcul d'un Skycube a pour effet de diminuer largement le nombre de tuples par rapport à la relation initiale. De ce fait, les représentations réduites du Skycube émergent sont utiles bien que moins efficaces que celles du cube de données émergent.

6.6. Conclusion

Dans ce chapitre, nous avons présenté le treillis des concepts Skylines qui est une instance contrainte d'un cadre formel plus général : le treillis des concepts Accords. Cette structure permet non seulement la matérialisation partielle sans perte d'information du Skycube mais aussi la reconstruction efficace des cuboïdes, ou Skycubïdes, manquants. Nous pouvons d'ailleurs facilement étendre cette matérialisation au cas des ϵ-Skylines ou Skylines approximatifs (PAPADIAS, TAO, G. FU et al. 2005), en généralisant la définition d'ensemble en accord en remplaçant l'égalité stricte par une égalité à ϵ près. Une telle extension permet au décideur de relâcher la contrainte de dominance lorsque le nombre de résultats n'est pas suffisant.

Les Skycubes ainsi partiellement matérialisés, et une fois fusionnés, servent de relation d'entrée pour le calcul du Skycube émergent ou de ses réductions, le Skycube fermé émergent, avec perte d'information au niveau des mesures, et le L-Skycube fermé émergent, sans perte d'information. À partir du moment où le format d'entrée est respecté, les concepts sont naturellement transposés du monde cube de émergent à celui Skycube de données émergent. De même, l'usage des algorithmes E-IDEA, F-IDEA et $\mathbb{C}$-IDEA de la plateforme IDEA est encouragé et se fait de manière transparente.

TABLEAU 6.11. – L-Skycube fermé émergent

Tuple fermé émergent [a]	$ER_{\texttt{Durée}}$	$ER_{\texttt{Échec}}$
$(ALL, ALL, B, \ldots)$	1.17	1.67
$(ALL, C, ALL, \ldots)$	1.17	1.67
$(OU, ALL, ALL, \ldots)$	2.33	3
$(OU, C, B, \ldots)$	1.17	1.67
$(ALL, ALL, F, \ldots)$	∞	∞
$(ALL, ALL, B, \ldots)$	∞	∞
$(ALL, B, ALL, \ldots)$	∞	∞
$(ALL, C, ALL, \ldots)$	∞	∞
$(OU, ALL, ALL, \ldots)$	∞	∞
$(ALL, E, D, \ldots)$	∞	∞
$(OU, B, F, \ldots)$	∞	∞
$(OU, C, B, \ldots)$	∞	∞
$(UU, E, D, \ldots)$	∞	∞

a. Avec :
— A : 121, 113, 006 (Staross, Leveinard, Dracaufeu) ;
— B : 065, 103, 065 (Alakazam, Noadkoko, Alakazam) ;
— C : 121, 113, 080 (Staross, Leveinard, Flagadoss) ;
— D : 065, 113, 143 (Alakazam, Leveinard, Ronflex) ;
— E : 065, 040, 065 (Alakazam, Grodoudou, Alakazam) ;
— F : 121, 113, 121 (Staross, Leveinard, Staross).

7. CoSky

Sommaire

7.1. Introduction et motivations 215
7.2. Cas d'utilisation 216
7.3. Classement de Skyline 219
7.3.1. Méthode dp-idp 219
7.3.2. Amélioration de la méthode dp-idp 220
7.3.3. Méthode CoSky 222
7.3.3.1. Étape 1 : normalisation des attributs par la somme . . . 223
7.3.3.2. Étape 2 : pondération avec indice de Gini 224
7.3.3.3. Étape 3 : détermination du point idéal 224
7.3.3.4. Étape 4 : calcul des scores avec le cosinus de Salton . . 225
7.3.3.5. Étape 5 : classement des résultats 225
7.3.4. Méthode top-k 227
7.3.4.1. DeepSky 227
7.4. Conclusion 228

7.1. Introduction et motivations

L'une des principales problématiques de l'opérateur Skyline apparait lorsque le jeu de résultats possède une cardinalité forte. Dans ce cas-là, il est le plus souvent peu évident d'extraire une quelconque information significative. En effet, plus les points d'un Skyline sont nombreux, plus ils tendent à présenter des intérêts proches, et moins il est possible de constater des discriminations significatives entre eux. Afin de palier à cela, un classement efficace peut être nécessaire. Cette solution est surtout efficace dans le cas de données présentant de nombreuses dimensions ou une faible corrélation. Peu d'approches de classement de Skyline prenant en compte le problème du nombre important de résultats ont été proposées.

L'approche dp-idp est certainement une des approches les plus récentes. Elle utilise la dominance de Pareto (cf. paragraphe 2.4.2.1 p. 76) afin de déterminer un score pour chaque point d'un Skyline. La méthode dp-idp s'inscrit dans le même esprit que le schéma de pondération tf-idf utilisé en recherche d'information [1].

1. La recherche d'information (RI, ou IR pour *information retrieval*), est un domaine qui porte sur la recherche et la récupération de données ou d'informations à partir d'un ensemble de données non structurées, telles que des textes, des images, des vidéos ou des sons. Son objectif est de fournir des résultats pertinents à une demande du décideur.

Dans ce chapitre, nous proposons tout d'abord d'améliorer la méthode dp-idp en utilisant le concept de hiérarchie de dominance afin de perfectionner le calcul des scores des points d'un Skyline, puis nous définissons la méthode CoSky ayant pour objectif de classer efficacement les points d'un Skyline sans calculer la dominance de Pareto, avant de présenter l'algorithme DeepSky caractérisé par la méthode CoSky.

7.2. Cas d'utilisation

Pour ce chapitre, nous reprenons l'exemple appliqué au jeu vidéo Pokémon Showdown! présenté au chapitre 6 p. 174, celui que nous avons étudié et expérimenté initialement lors de la recherche de cette contribution, et ses relations exemples $\texttt{Pokémon}_1$ et $\texttt{Pokémon}_2$ remaniées (cf. tableau 7.1 et 7.2).

TABLEAU 7.1. – La relation $\texttt{Pokémon}_1$

RowId	...	Joueur [a]	Adversaire [b]	...	Rareté [c]	Durée [d]	Échec [e]
1	...	121,113,103(*G*)	121,113,121(*F*)	...	5	20	30
2	...	065,103,065(*B*)	065,143,065(*E*)	...	4	60	50
3	...	121,113,121(*F*)	065,103,065(*B*)	...	5	30	40
4	...	121,113,080(*C*)	065,143,065(*E*)	...	1	80	40
5	...	121,113,128(*A*)	121,113,121(*F*)	...	5	90	60
6	...	065,113,143(*D*)	065,113,143(*D*)	...	9	30	50
7	...	065,143,065(*E*)	121,113,143(*H*)	...	7	80	60
8	...	065,113,143(*D*)	065,103,065(*B*)	...	9	90	70

a. Avec les numéros officiels des Pokémon :
— n°065 : Alakazam ;
— n°080 : Flagadoss ;
— n°103 : Noadkoko ;
— n°113 : Leveinard ;
— n°121 : Staross ;
— n°128 : Tauros ;
— n°143 : Ronflex.

b. idem supra.

c. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de Rareté r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

d. En nombre de tours de combat au total.

e. En pourcentage.

Exemple 7.1 - Les relations $\texttt{Pokémon}_1$ et $\texttt{Pokémon}_2$ illustrées respectivement par le tableau 7.1 et le tableau 7.2 sont typiques pour l'utilisation de calcul de Skyline, avec ou sans opérateur. La relation $\texttt{Pokémon}_1$ conserve les attributs Joueur, représentant la séquence de Pokémon jouée par le joueur, et Adversaire, la séquence

FIGURE 7.1. – Alakazam, Flagadoss, Noadkoko, Leveinard, Staross, Tauros, Ronflex [a]

a. De gauche à droite : Alakazam (n°065), Flagadoss (n°080), Noadkoko (n°103), Leveinard (n°113), Staross (n°121), Tauros (n°128), Ronflex (n°143) ; illustrations de Pokémon Versions Rouge Feu et Vert Feuille sur Poképédia (https://www.pokepedia.fr/).

TABLEAU 7.2. – La relation Pokémon$_2$

RowId	Rareté [a]	Durée [b]	Victoire [c]
1	5	20	70
2	4	60	50
3	5	30	60
4	1	80	60
5	5	90	40
6	9	30	50
7	7	80	40
8	9	90	30

a. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de Rareté r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

b. En nombre de tours de combat au total.

c. En pourcentage.

de l'adversaire. Pour des raisons de commodités, la relation Pokémon$_2$ ne conserve aucun attribut, seulement le RowId des tuples. Dans les deux cas, les critères de choix afin de déterminer le « meilleur ordre d'apparition de Pokémon dans un combat » sont la Rareté des Pokémon de la séquence jouée par le joueur, la Durée (en nombre de tours) du combat et le taux d'Échec de l'ordre d'apparition des Pokémon du joueur par rapport à celui de l'adversaire. Tous ces critères sont, de préférence, à minimiser. Pour la relation Pokémon$_2$, le taux d'Échec est remplacé par le taux de Victoire, que nous n'utilisons pas habituellement, mais qui est simplement son complémentaire. Pour l'exemple, nous nous limitons à trois Pokémon par séquence, qu'elle soit du joueur ou de l'adversaire, et afin de simplifier leur usage dans leurs représentations nous les indiquons par des listes de numéros de Pokémon, et leur affectons une lettre pour les désigner (en conservant celles de la relation exemple du chapitre 6 p. 174 pour des raisons de commodités). Ainsi, la liste $121,113,128(A)$ correspond à la séquence des trois Pokémon Staross, Leveinard et Tauros. Pour la lisibilité, nous arrondissons par un multiple de cinq les durées des combats, les taux d'échecs et les taux de victoires.

La relation Pokémon$_1$ est donnée afin de rendre plus immédiat le lien avec la relation exemple du chapitre 6 p. 174 et la relation Pokémon$_2$, mais dans le reste de ce chapitre, nous ne considérons plus que la relation Pokémon$_2$ afin de pouvoir se concentrer sur des calculs numériques sur les critères et en montrant que les préférences Skyline peuvent être mixtes, et pas uniquement MIN (ou MAX). La requête SQL correspondante à la relation Pokémon$_2$ utilisant l'opérateur Skyline est la suivante :

```
01 |  SELECT *
02 |  FROM Pokémon
03 |  SKYLINE OF Rareté MIN, Durée MIN, Victoire MAX
```

Et, la requête associée sans opérateur Skyline est la suivante :

```
01 |  SELECT *
02 |  FROM Pokémon AS P1
03 |  WHERE NOT EXISTS (SELECT *
04 |                    FROM Pokémon AS P2
05 |                    WHERE (P2.Rareté <= P1.Rareté
06 |                      AND P2.Durée <= P1.Durée
07 |                      AND P2.Victoire >= P1.Victoire)
08 |                      AND (P2.Rareté < P1.Rareté
09 |                       OR P2.Durée < P1.Durée
10 |                       OR P2.Victoire > P1.Victoire));
```

L'ensemble obtenu est composé des tuples $(1,5,20,70)$, $(2,4,60,50)$ et $(4,1,80,60)$. Il s'agit de l'ensemble des ordres d'apparition de Pokémon dans un combat qui sont aussi bons ou meilleurs selon toutes les dimensions critères considérées (Rareté, Durée et Victoire) et meilleur pour au moins l'un de ces critères. Dans une représentation graphique, spécifiée par la dominance de Pareto (cf. définition 2.21 p. 76), nous appelons point appartenant à un Skyline, ou point du Skyline, ou encore point Pareto-optimal, sp, chacun de ces tuples, et leur ensemble forme un Skyline S.

7.3. Classement de Skyline

De nombreux travaux ont été consacrés à l'étude du classement de Skyline. Dans CHAN, JAGADISH, TAN et al. 2006, une métrique appelée fréquence Skyline est proposée afin d'ordonner un Skyline selon les points avec une haute fréquence Skyline. Cette méthode s'adapte bien à un grand nombre de dimensions et les expérimentations présentent une belle efficacité de l'algorithme.

Les requêtes top-k (YIU et MAMOULIS 2007) sont, en quelque sorte, une alternative aux requêtes Skyline. Il s'agit d'une méthode de classement sans fonction d'évaluation. Une autre technique se base sur la définition d'une « forme » représentant la recherche de l'utilisateur en spécifiant des régions définies par le décideur qui dominent toutes les autres régions (BARTOLINI, CIACCIA, ORIA et al. 2007). Une approche de classement de Skyline pour un Skycube (LAKHAL, NEDJAR et CICCHETTI 2017) se concentrant sur les points de Skyline les plus chargés en information a aussi été proposé (VLACHOU et VAZIRGIANNIS 2010). Cette méthode capture les relations de dominance entre les points de Skyline appartenant à différents sous-espaces. Un nouvel opérateur a aussi été introduit afin de trouver le point de Skyline le plus digne d'intérêt (L. CHEN, GANG et Q. LI 2015).

7.3.1. Méthode dp-idp

La méthode dp-idp (pour *dominance power and inverse dominance power*) est inspirée du schéma de pondération tf-idf (pour *term frequency-inverse document frequency*) utilisé en recherche d'information, qui attribue à un terme t un poids dans un document d. L'idée sous-jacente n'est pas de déterminer le nombre d'occurrences de chaque terme de la requête t dans d, mais plutôt le poids tf-idf de chaque terme dans d. L'objectif étant de trouver des mots-clefs importants dans un corpus documentaire. Dans le contexte Skyline, les points dominés ont des impacts différents sur les points du Skyline. Ainsi, leur contribution dépend de caractéristiques locales correspondant à des points du Skyline et de caractéristiques globales correspondant au Skyline entier. Dans VALKANAS, PAPADOPOULOS et GUNOPULOS 2014, il est proposé que l'importance d'un point soit inversement proportionnelle au nombre de points qui le domine. Le schéma dp-idp prend en compte les positions relatives des points dominés afin de les différencier. Il est centré sur les points qui sont peu dominés par d'autres : *e.g.* soit sp un point d'un Skyline, si $sp \prec_d p_1, sp \prec_d p_2$ et ni p_1 ni p_2 ne dominent l'autre, ils contribuent autant l'un que l'autre à sp. Sinon, si $p_1 \prec_d p_2$, alors $score(p_1) > score(p_2)$, et par conséquent la contribution de p_1 est plus importante. L'idp d'un point $p \in D \backslash S$ correspond au nombre de points du Skyline qui dominent p. Moins un point p apparaît de manière fréquente dans un ensemble de points dominés d'un Skyline, plus il est considérable :

$$idp(p) = \log \frac{|S|}{|\{sp \in S : sp \prec_d p\}|} \tag{7.1}$$

Afin de calculer le dp d'un point dominé p, sa position relative par rapport à un

point du Skyline sp est essentielle. Ainsi, un même point dominé peut contribuer différemment à différents points du Skyline. Pour cette raison, il est nécessaire de calculer la couche de *minima* (ou *layer of minima*) $lm(p, sp)$ où se situe le point dominé p par rapport à sp. Le pouvoir de dominance de p est alors l'inverse de la valeur de sa couche : le $Score(sp)$ qui mesure l'importance d'un point d'un Skyline sp est défini de la manière suivante :

$$Score(sp) = \sum_{p:sp \prec_d p} \frac{1}{lm(p, sp)} \log \frac{|S|}{|\{sp' \in S : sp' \prec_d p\}|} \tag{7.2}$$

L'algorithme Baseline (VALKANAS, PAPADOPOULOS et GUNOPULOS 2014) est proposé pour classer un Skyline sur la base du schéma dp-idp. Les principales étapes de cet algorithme sont :

— extraction des couches de *minima* de chaque point du Skyline sp ;
— mettre à jour $Score(sp)$ en considérant pour chaque point p dans chaque couche de *minima* $lm(p, sp)$ le nombre de points qui le dominent ;
— ordonner le Skyline en fonction des différents calculs.

Cet algorithme n'est pas parfait, et il est notamment assez couteux en temps (VALKANAS, PAPADOPOULOS et GUNOPULOS 2014). Son principal défaut est, selon nous, la complexité induite par les calculs des couches de *minima* $lm(p, sp)$. Pour y palier, nous proposons de perfectionner cette approche afin d'améliorer l'efficacité de l'algorithme Baseline.

7.3.2. Amélioration de la méthode dp-idp

La relation de dominance peut être vu comme un tri hiérarchique. Autrement dit, un point d'un Skyline a nécessairement une position hiérarchique supérieure à celles des points qu'il domine. Cela nous a encouragé à mettre en correspondance la relation de dominance vue ci-dessus avec un graphe que nous appelons hiérarchie de dominance. L'utilisation d'une hiérarchie de dominance au sein de la méthode de classification dp-idp permet un meilleur calcul des couches de *minima* et conduit, par conséquent, à de meilleurs résultats en matière de classement. Le graphe que nous proposons est un graphe orienté acyclique donnant une représentation d'un ensemble partiellement ordonné en établissant son graphe de couverture [2]. Un graphe orienté acyclique offre un ordre topologique qui peut donner une excellente représentation d'une hiérarchie d'un point appartenant à un Skyline sp par rapport aux points qu'il domine.

Définition 7.1 (Hiérarchie de dominance) - Soit un ensemble de points D et un ordre de dominance $\prec_d$, alors la hiérarchie de dominance (HD, ou DH pour *dominance hierarchy* en anglais) est le graphe de couverture de l'ensemble ordonné $(D, \prec_d)$.

Exemple 7.2 - La figure 7.2 représente une hiérarchie de dominance ayant comme sommet le point du Skyline sp. L'ordre de dominance est illustré par les arrêtes entre

2. Un graphe de couverture d'un graphe est un graphe couvrant tous les sommets du graphe d'origine en utilisant pour cela le moins d'arêtes possible.

sp et les points qu'il domine (p_1, p_2, p_3 et p_4).

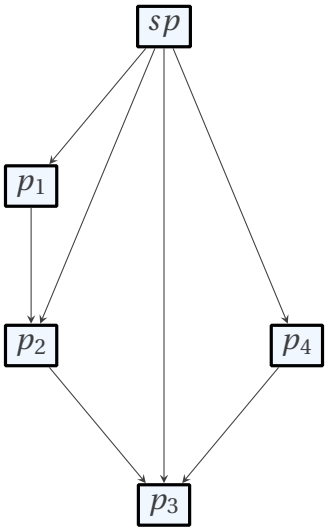


FIGURE 7.2. – Exemple de graphe de hiérarchie de dominance

Nous considérons la couche de *minima* $lm(p, sp)$ comme le nombre de sommets du chemin minimal entre sp et p dans la hiérarchie de dominance.

Exemple 7.3 - Pour calculer la couche *minima* $lm(p3, sp)$, nous voyons sur la figure 7.2 qu'il y a deux chemins allant de sp jusqu'à p_3 :

1. Premier chemin : $sp \rightarrow p_1 \rightarrow p_2 \rightarrow p_3$.
2. Second chemin : $sp \rightarrow p_4 \rightarrow p_3$.

Le premier chemin passe par quatre sommets alors que le second n'en compte que trois, donc le chemin minimal de sp à p_3 est le second et $lm(p_3, sp) = 3$.

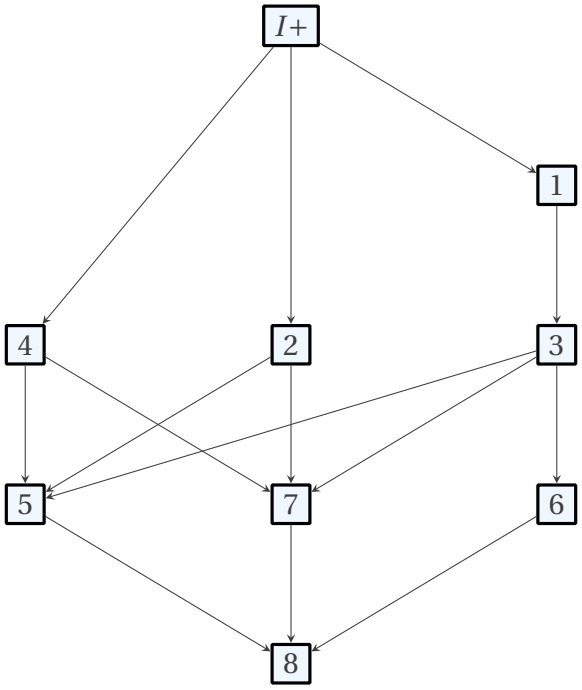


FIGURE 7.3. – Graphe de hiérarchie de dominance de Pokémon Showdown!

Exemple 7.4 - En considérant à nouveau la relation `Pokémon` (cf. tableau 7.2), le graphe de hiérarchie de dominance illustrant les relations de dominance entre les points du Skyline et les points dominés est donné par la figure 7.3 (I^+ étant le point idéal théorique, ou abstrait, qui domine tous les points du Skyline, et les points sont représentés par leur valeur de `RowId`).

Pour calculer le score de chaque point du Skyline, nous utilisons la formule 7.2. Ainsi, le classement descendant des points du Skyline obtenu, représentés par leur valeur de `RowId`, est soit 1,2,4, soit 1,4,2.

Sur la base des résultats obtenus (cf. tableau 7.3), nous pouvons en conclure que la méthode dp-idp n'offre pas toujours la possibilité de distinguer deux points du Skyline comme c'est le cas ici concernant ceux de valeur de `RowId` 2 et 4.

7.3.3. Méthode CoSky

Afin de proposer une solution de classement de Skyline qui permette, à l'inverse de dp-idp, de toujours différencier et d'ordonner des points dissociés d'un Skyline, nous présentons à présent la méthode CoSky. CoSky (pour cosinus Skyline) est une approche en plusieurs étapes qui n'utilise pas de relation de dominance. C'est, à notre connaissance, la première méthode de type TOPSIS [3] (LAI, T.-Y. LIU et C.-L. HWANG 1994; BEHZADIAN, KHANMOHAMMADI OTAGHSARA, YAZDANI et al. 2012) appliqué au

3. TOPSIS pour *technique for order preference by similarity to ideal solution*, est une méthode dont l'objectif est de classer par ordre de choix des alternatives sur la base de critères favorables ou défavorables.

TABLEAU 7.3. – Calcul de score avec la méthode dp-idp améliorée

sp [a]	Point dominé p [b]	$lm(p, sp)$	$Score(sp)$
1	3	$lm(3,1) = 2$	0.297
	5	$lm(5,1) = 3$	
	6	$lm(6,1) = 3$	
	7	$lm(7,1) = 3$	
	8	$lm(8,1) = 4$	
2	5	$lm(5,2) = 2$	0
	7	$lm(7,2) = 2$	
	8	$lm(8,2) = 3$	
4	5	$lm(5,4) = 2$	0
	7	$lm(7,4) = 2$	
	8	$lm(8,4) = 3$	

a. `RowId` des tuples sp.
b. `RowId` des tuples des points dominés.

classement de Skyline. TOPSIS est basé sur une normalisation vectorielle, un calcul de poids de chaque attribut, et un calcul de score de chaque point déterminé par une mesure géométrique des distances entre chaque alternative, représentée par un point, et les solutions idéales/anti-idéales. Dans la méthode CoSky, la normalisation des attributs est effectuée avec la somme, une pondération automatique des attributs normalisés selon l'indice de Gini, et le score utilise le cosinus de Salton de l'angle entre un point du Skyline et le point idéal.

Les détails de ce calcul sont présentés dans cette sous-section. Lors de toutes les étapes, nous considérons que $i \in [1..n]$ et $j \in [1..m]$ (où n est le nombre de tuples et m le nombre d'attributs).

7.3.3.1. Étape 1 : normalisation des attributs par la somme

Le Skyline est normalisé de manière à ce que chaque valeur d'attribut de chaque point du Skyline soit convertie en une valeur sur une échelle allant de 0 à 1. Cela permet d'éliminer les anomalies liées à des unités de mesure et des échelles différentes tout en s'assurant que les attributs soient toujours mesurables et comparables entre eux.

Soit S_N l'ensemble des points de Skyline normalisés, ou Skyline normalisé, et le tuple $p_i = (v_{i1}, v_{i2}, \ldots, v_{im}) \in S_N$, alors nous avons :

$$v_{ij} = \frac{t_i[A_j]}{\sum_{i'=1}^{n} t_{i'}[A_j]}, \forall t_i \in S \tag{7.3}$$

7.3.3.2. Étape 2 : pondération avec indice de Gini

Le classement de Skyline vise le plus souvent à distinguer de manière stricte les points du Skyline. A cette fin, il est crucial de déterminer une mesure qui permette cette distinction. Pour ce faire, plusieurs mesures ont été proposées dans la littérature. Le concept d'entropie a ainsi été utilisé dans des problèmes multicritères et une méthode basée sur l'entropie a été proposée (HUANG 2008 ; LOTFI et FALLAHNEJAD 2010). Cette méthode s'adapte bien à un contexte Skyline, dans lequel nous cherchons à différencier les valeurs des attributs par un classement de Skyline afin d'avoir une meilleure prise de décision. Cependant, cette méthode a aussi ses limites, notamment en ce qui concerne le calcul de l'entropie qui nécessite l'usage d'une fonction logarithmique, conséquente en temps.

Nous préférons présenter une autre mesure, l'indice de Gini [4], qui est plus rapide à calculer que l'entropie et n'utilise pas de logarithme pour la pondération automatique des attributs. Dans la méthode présentée, l'indice de Gini est utilisé pour dériver les poids des attributs afin de déterminer le degré de divergence des valeurs des attributs. L'indice de Gini de A_j, $Gini(A_j)$, est déterminé à l'aide de l'équation suivante :

$$Gini(A_j) = 1 - \sum_{i=1}^{n} t_i[A_j]^2 \tag{7.4}$$

Nous appelons W le poids de l'attribut A_j. Le décideur peut ne pas faire appel à l'indice de Gini et spécifier directement les poids $(W(A_1), W(A_2), \ldots, W(A_m))$ avec la formule $\texttt{SUM}(W(A_1), W(A_2), \ldots, W(A_m)) = 1$. Le poids d'un attribut, dans ce cas, est alors son importance pour le décideur, et est donné par la formule suivante :

$$W(A_j) = \frac{Gini(A_j)}{\sum_{j'=1}^{m} Gini(A_{j'})} \tag{7.5}$$

Soit S_P l'ensemble des points de Skyline après pondération des attributs, ou Skyline pondéré, et le tuple $p_i = (a_{i1}, a_{i2}, \ldots, a_{im}) \in S_P$, alors nous avons :

$$a_{ij} = W(A_j) \times t_i[A_j], \forall\, t_i \in S_N \tag{7.6}$$

7.3.3.3. Étape 3 : détermination du point idéal

Le point idéal théorique, ou abstrait, noté I^+, qui domine tous les points du Skyline. Il correspond au tuple répondant de manière optimale aux préférences du décideur.

Ainsi, soit $I^+ = (v_1, v_2, \ldots, v_m)$, alors nous avons :

$$v_j = \begin{cases} \texttt{MAX}(A_j) \equiv Pref(A_j) = \texttt{MAX} \\ \texttt{MIN}(A_j) \equiv Pref(A_j) = \texttt{MIN} \end{cases} \tag{7.7}$$

4. L'indice (ou coefficient) de Gini est une mesure statistique utilisée pour évaluer l'inégalité d'une variable par rapport à une population donnée. Il varie entre 0 (égalité parfaite) et 1 (inégalité totale), et l'inégalité est d'autant plus forte que l'indice est élevé.

Exemple 7.5 - En considérant la relation `Pokémon`$_2$ (cf. tableau 7.2), rechercher l'ordre idéal d'apparition de Pokémon dans un combat combine des conditions sur la `Rareté`, qui doit être la plus basse possible, la `Durée`, la plus courte possible, et le taux de `Victoire`, le plus fort possible.

7.3.3.4. Étape 4 : calcul des scores avec le cosinus de Salton

Cette étape vise à déterminer le score d'un point de Skyline avec le cosinus de Salton[5]. Pour ce faire, nous calculons le cosinus de l'angle entre le point idéal et le point du Skyline. Plus l'angle est faible (et donc le cosinus de l'angle est élevé), plus le point du Skyline est important.

Soit S_{Score} l'ensemble des scores des points du Skyline, le tuple $p_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \in S_p$ et le point idéal $I^+ = (y_1, y_2, \ldots, y_m)$, alors nous avons :

$$t_i[Score] = cos(p_i, I^+) = \frac{p_i.I^+}{||p_i||.||I^+||} \tag{7.8}$$

$$t_i[Score] = \frac{\sum_{j=1}^{m} x_{ij}.y_j}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}.\sqrt{\sum_{j=1}^{m} y_j^2}}, \forall\, t_i \in S_{Score} \tag{7.9}$$

Une conséquence de cette équation est que $t_i[score] = 1$ si et seulement si le point du Skyline est le plus intéressant, et $t_i[score] = 0$ si et seulement si le point du Skyline est le moins intéressant.

Nous pouvons utiliser le principe de similarité de TOPSIS pour calculer le score de chaque point d'un Skyline de la manière suivante : soit I^+ le point idéal et $Score_I$ son score, I^- le point anti-idéal et $Score_{AI}$ son score, alors $\forall p_i \in S_P$, $t_i[Score_I] = cos(p_i, I^+)$ et $ti[Score_{AI}] = cos(p_i, I^-)$. Alors, si nous considérons $I^- = (v_1, v_2, \ldots, v_m)$, alors nous avons :

$$v_j = \begin{cases} \texttt{MAX}(A_j) \equiv Pref(A_j) = \texttt{MIN} \\ \texttt{MIN}(A_j) \equiv Pref(A_j) = \texttt{MAX} \end{cases} \tag{7.10}$$

7.3.3.5. Étape 5 : classement des résultats

La dernière étape de la méthode CoSky a pour objectif d'ordonner les points du Skyline par rapport aux scores calculés.

Exemple 7.6 - Il est possible d'appliquer les différentes étapes de la méthode CoSky à la relation `Pokémon`$_2$ (cf. tableau 7.2) en utilisant la requête SQL suivante :

```
01 | WITH S AS (
02 |   SELECT *
03 |   FROM Pokémon
04 |   SKYLINE OF Rareté MIN, Durée MIN, VICTOIRE MAX
05 | ),
```

5. Le cosinus de Salton, ou mesure de similarité cosinus, est une mesure de similarité d'une échelle allant de 0 à 1 permettant de comparer des vecteurs dans un espace vectoriel.

```
06 |  SN AS (
07 |    SELECT RowId,
08 |    Rareté / TRareté AS NRareté,
09 |    Durée / TDurée AS NDurée,
10 |    Victoire / TVic AS NVic
11 |    FROM S, (SELECT SUM(Rareté) AS TRareté,
12 |                    SUM(Durée) AS TDurée,
13 |                    SUM(Victoire) AS TVic FROM S) AS ST
14 |  ),
15 |  SGini AS (SELECT
16 |    1 - (SUM(NRareté * NRareté)) AS GRareté,
17 |    1 - (SUM(NDurée * NDurée)) AS GDurée,
18 |    1 - (SUM(NVic * NVic)) AS GVic
19 |    FROM SN
20 |  ),
21 |  SW AS (SELECT
22 |    GRareté / (GRareté + GDurée + GVic) AS WRareté,
23 |    GDurée / (GRareté + GDurée+ GVic) AS WDurée,
24 |    GVic / (GRareté + GDurée + GVic) AS WVic
25 |    FROM SGini
26 |  ),
27 |  SP AS (SELECT
28 |    RowId,
29 |    WRareté * NRareté AS PRareté,
30 |    WDurée * NDurée AS PDurée,
31 |    WVic * NVic AS PVic
32 |    FROM SN, SW
33 |  ),
34 |  Idéal AS (SELECT
35 |    MIN(PRareté) AS IRareté,
36 |    MIN(PDurée) AS IDurée,
37 |    MAX(PVic) AS IVic
38 |    FROM SP
39 |  ),
40 |  SScore AS (SELECT
41 |    RowId,
42 |    (IRareté * PRareté + IDurée * PDurée + IVic * PVic) /
43 |    (
44 |      SQRT(PRareté * PRareté + PDurée * PDurée + PVic * PVic)
45 |      *
46 |      SQRT(IRareté * IRareté + IDurée * IDurée + IVic * IVic)
47 |    )
48 |    AS Score
49 |    FROM Idéal, SP
50 |  )
51 |  SELECT P.RowId, Rareté, Durée, Victoire,
52 |         ROUND(Score, 3) AS Score
53 |  FROM S P INNER JOIN SScore rs
54 |  ON P.RowId = rs.RowId
55 |  ORDER BY Score DESC;
```

Les résultats obtenus sont donnés au tableau 7.4.

TABLEAU 7.4. – Classement de Skyline avec la méthode CoSky

RowId	Rareté	Durée	Victoire	Score
1	5	20	70	0.818
4	1	80	60	0.801
2	4	60	50	0.761

Nous voyons donc que contrairement à la méthode dp-idp, la méthode CoSky permet de distinguer clairement les points de Skyline. Les points de Skyline de `RowId` 2 et 4 ont, avec CoSky, des scores différents de 0 et différents l'un de l'autre, alors qu'ils étaient tous les deux de 0 avec la méthode dp-idp. L'utilisation de la méthode CoSky nous permet donc bien d'avoir un classement de Skyline complet.

7.3.4. Méthode top-k

Pour cette approche, nous utilisons le principe de Skyline multiniveaux (PREISINGER et ENDRES 2015) permettant de trouver les top-k points de Skyline, non ordonnés entre eux. Une requête Skyline top-k Q_k sur une relation r calcule les top-k points, en fonction des préférences Skyline de S. Soit les points du niveau 0 du Skyline multiniveaux $S_0(r)$, ou points du Skyline, tels que $S_0(r) = S$, et $Card(r)$ la cardinalité de r telle que $Card(r) > k$, alors nous avons :

— si $Card(S_0(r)) > k$, alors Q_k retourne seulement k points de $S_0(r)$;
— si $Card(S_0(r)) = k$, alors Q_k retourne le Skyline entier (*i.e.* l'ensemble des points de $S_0(r)$) ;
— si $Card(S_0(r)) < k$, alors il n'y a pas assez de points dans $S_0(r)$ pour permettre une réponse correcte avec Q_k. Une approche Skyline multiniveaux doit alors être appliquée. Cela signifie que, non seulement, des points de $S_1(r)$ de $(r \setminus S_0(r))$ sont retournés également, mais aussi potentiellement certains points de $S_2(r)$ de $(r \setminus (S_0(r) \cup S_1(r))$, de $S_3(r)$... tant que le nombre cumulé de résultats retournés est inférieur à k.

7.3.4.1. DeepSky

L'algorithme DeepSky (cf. algorithme 12) utilise ce principe multiniveaux allié à la méthode de classement CoSky afin de trouver les top-k points de Skyline ordonnés. Il retourne les k points de Skyline multiniveaux qui ont les k plus haut scores calculés par la méthode CoSky.

Exemple 7.7 - En considérant à nouveau la relation $\texttt{Pokémon}_2$ (cf. tableau 7.2), avec $k = 4$, l'algorithme DeepSky (cf. algorithme 12) renvoie les points de Skyline de `RowId` 1, 4 et 2, les points de Skyline classés au niveau 0, et le point de Skyline de `RowId` 3, le seul point classé de niveau 1.

Algorithme 12 Algorithme DeepSky

```
Entrée :
   La relation r.
   Le nombre k.
Sortie :
   Les top-k tuples/points avec les meilleurs scores top_k.
01: soit tot := 0; //Nombre total de résultats calculés
02: soit r_l := r; //Niveau courant
03: tant que tot ≥ k ∨ r_l = ∅ faire
04:   S := CoSky(r_l);
05:   //Card() calcule la cardinalité d'un ensemble
06:   tot := tot + Card(S);
07:   si tot < k alors
08:     top_k := top_k ∪ S;
09:     r_l := r_l \ S;
10:   sinon si tot ≥ k alors
11:     top_k := les k premiers points de S;
12:     retourner top_k
13:   fin si
14: fin tant que
15: retourner top_k
```

7.4. Conclusion

Dans ce chapitre, nous avons présenté trois nouvelles méthodes pour ordonner les points de Skyline.

La première proposée consiste en l'amélioration de la méthode dp-idp par une hiérarchie de dominance permettant d'avoir un classement rapide d'un Skyline.

La deuxième est la méthode CoSky basée sur d'une part sur l'approche TOPSIS du domaine de l'aide à la décision multicritère, et d'autre part, la mesure de similarité cosinus de Salton utilisé en recherche d'information. Un exemple d'implémentation SQL de la méthode proposée a également été exposé.

Enfin, la troisième méthode présentée permet de trouver les k points de Skyline les mieux classés, c'est-à-dire ayant les scores les plus élevés, en utilisant le principe de Skyline multiniveaux couplé à la méthode CoSky. Pour cela, l'algorithme DeepSky a été introduit.

Nous utilisons notamment l'algorithme DeepSky pour notre méthode TAGED, en l'appliquant à un contexte de grand volume de données de jeux vidéo.

8. TAGED

Sommaire

8.1. Introduction et motivations . . . 229
8.2. Cas d'utilisation . . . 230
8.3. Valorisation . . . 230
8.3.1. Dépôt par l'Agence de protection de programmes (APP) . . . 231
8.4. Méthode TAGED . . . 231
8.4.1. Phase 1 : configuration . . . 234
8.4.2. Phase 2 : collecte . . . 237
8.4.3. Phase 3 : analyse . . . 238
8.4.4. Phase 4 : classement . . . 242
8.4.5. Phase 5 : rendu . . . 243
8.4.6. Plateforme algorithmique TAGED . . . 244
8.5. Applications aux jeux vidéo . . . 244
8.5.1. Applications au game design . . . 244
8.5.2. Applications à l'e-sport . . . 246
8.5.2.1. Contexte et enjeux de l'e-sport . . . 246
8.5.2.2. Le statut de l'e-sport . . . 249
8.5.2.3. Arbitrage dans l'e-sport . . . 250
8.5.2.4. Détection automatique et aide à l'e-arbitrage . . . 253
8.5.2.5. Méthodes de classement . . . 255
8.5.2.6. L'e-sport comme outil éducatif . . . 259
8.5.2.7. L'e-sport comme domaine de recherche . . . 259
8.6. Conclusion . . . 260

8.1. Introduction et motivations

Créer un jeu vidéo, et espérer qu'il obtienne autant de succès que possible, devient un défi de taille dans l'environnement compétitif de la première industrie culturelle mondiale. Les jeux vidéo sont bien souvent de plus en plus complexes à concevoir, à développer et à équilibrer.

Les jeux vidéo, notamment ceux en ligne, sont en mesure d'enregistrer la moindre action réalisée par un joueur, collectionnant ainsi des milliards de données sur des milliers et parfois des millions de joueurs. Pour gérer une telle masse d'informations, les outils d'assistance semblent devenir incontournables.

La méthode TAGED (pour *tool assisted game equilibrium design*) que nous proposons offre une solution adaptable à des jeux vidéo solo ou multijoueur possédant un volume important de données chiffrées ou qualitatives exploitables, qu'elles soient intrinsèques au jeu lui-même (des données de personnages, d'objets, de niveaux...) ou liées à son utilisation par les joueurs (des durées de parties ou de boucles de jeu, des scores intermédiaires, des séquences d'actions...).

La recherche d'un bon équilibre dans un jeu vidéo est un concept central car il va fortement impacter la satisfaction des joueurs. Dans le cadre d'un jeu solo, s'il est déséquilibré en terme de difficulté, les joueurs risquent très rapidement de s'en désintéresser par absence de défi, ou inversement, par frustration lié à l'impossibilité d'atteindre son objectif. Dans les jeux multijoueurs, la problématique gagne à la fois en importance et en complexité de part la multiplicité des situations qui explose. Il devient donc crucial de détecter au plus vite ces déséquilibres qui peuvent radicalement déstabiliser un jeu vidéo lorsqu'ils sont découverts par des joueurs qui en tirent un profit perçu comme immérité. L'objectif de notre travail de recherche est donc à la fois de détecter ces déséquilibres par une analyse des données du jeu vidéo et de son moteur de règles mais aussi de les découvrir grâce à une analyse en temps réel de la dynamique des comportements des joueurs.

TAGED propose une double analyse, celle des déséquilibres intrinsèques (à propos des éléments de *game design*) d'une part et des données extrinsèques(concernant les *verbatims* des comportements de joueurs, déviants ou atypiques, et des informations de parties) d'autre part.

8.2. Cas d'utilisation

Pour ce chapitre, nous reprenons l'exemple appliqué au jeu vidéo Pokémon Showdown! présenté au chapitre 6 p. 174, celui que nous avons étudié et expérimenté initialement lors de la recherche de cette contribution.

Dans le domaine des jeux vidéo, les phénomènes qui deviennent, au fil du temps, importants dans une population donnée alors qu'ils étaient jusqu'alors négligeables, les caractéristiques différentes et marquées entre deux populations ou les différences sensibles dans leur comportement, peuvent typiquement être selon le niveau estimé des joueurs. Ainsi, des joueurs ayant des classements significativement distincts (8.5.2.5) peuvent ne pas utilisé les mêmes stratégies de jeu.

8.3. Valorisation

Les travaux présentés dans ce chapitre de cette thèse ont été réalisés en collaboration avec la SATT SE. La SATT SE est une société privée, partagée entre plusieurs organismes de recherche publics et créée pour développer et commercialiser les résultats de la recherche pour les transformer en produits innovants. Son concours est aussi très utile afin de protéger des inventions et faciliter la collaboration avec le monde industriel.

La SATT SE est intéressée à connaître les applications, potentielles ou réelles, de l'algorithme TAGED développé dans le cadre de cette thèse.

Cette collaboration de plusieurs mois a pour but de de porter à pré-maturation notre idée du niveau besoin de validation à celui réalisation de prototypes de taille réelle (soit passer TAGED d'un TRL [1] 3 à 5) sur trois jeux vidéo de types très populaires et différents, sur lesquels nous nous sommes par ailleurs répartis les travaux décrit dans ce manuscrit : jeu vidéo de collection pour Pokémon Showdown!, Hack 'n' slash pour Diablo III : Reaper of Souls et jeu vidéo *puzzle-game* de type Match 3 pour Open Match 3. Cette diversité a été décidée pour servir de véritable démonstrateur de notre proposition, et ce même pour des jeux vidéo qui semblent avoir peu en commun.

FIGURE 8.1. – Société d'Accélération du Transfert des Technologies Sud-Est (SATT SE)

8.3.1. Dépôt par l'Agence de protection de programmes (APP)

Dans le cadre de cette thèse, et en tant que créations d'œuvres numériques, les algorithmes IDEA et TAGED ont été déposé par l'Agence de protection de programmes (APP) pour le compte de l'Inter deposit digital number (IDDN) respectivement sous le numéro IDDN.FR.001.140029.001.S.C.2014.000.31235 le 30 juin 2020 et sous le numéro IDDN.FR.001.390019.000.S.C.2023.000.40000 le 28 septembre 2023 [2], pour le compte de l'Université d'Aix-Marseille (AMU) et du Centre national de la recherche scientifique (CNRS), adhérent sous le numéro 12.13.14074.

8.4. Méthode TAGED

TAGED (pour *tool assisted game equilibrium design*) est une proposition de solution méthodologique adaptable permettant d'offrir un large panel d'usages dans l'équilibrage en conception de jeu vidéo. Outil d'assistance issu des domaines mathématiques de la théorie des jeux ainsi que des treillis et de la fouille de données, il a pour objectif d'œuvrer à la suppression ou à l'atténuation de la dominance stratégique dans les jeux vidéo (cf. sous-section 4.6.2.4 p. 140). Son utilisation par les concepteurs de jeux peut se faire durant toute la durée de vie d'un dispositif ludique : qu'il soit en conception, en développement ou en production.

1. L'échelle TRL (signifiant en anglais *technology readiness level*) est une mesure permettant d'évaluer le niveau de maturité d'une technologie.

2. (Le certificat électronique du dépôt de TAGED est disponible à l'URL suivante : https://secure2.iddn.org/app.server/certificate/?sn=2023390019000&key=1fbc8463476b6e729fb40ec9feadc6adc5044861a452a63fb329a26d9b896be1

Inter Deposit Digital Number

Certificat délivré par

Agence pour la Protection des Programmes

54 rue de Paradis - 75010 PARIS - FRANCE / SIREN 385 385 844 / T. +33(0)1 40 35 03 03 / app@app.asso.fr

IDDN.FR.001.140029.001.S.C.2014.000.31235

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

Pour l'œuvre : **IDEA-Master version 1 en date du 22 octobre 2019**

Identité des titulaires de droits* :

UNIVERSITÉ D'AIX-MARSEILLE
Jardin du Pharo
58 boulevard Charles Livon
CS 60006
13284 MARSEILLE CEDEX 7
FRANCE
Siren : 130015332

CNRS - CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
3 rue Michel-Ange
75794 PARIS CEDEX 16
FRANCE
Siren : 180089013

INTERDEPOSIT CERTIFICATION

UNIVERSITÉ DE TOULON
Avenue de l'Université
83130 LA GARDE
FRANCE
Siren : 198307662

Adhérent sous le numéro : **12.13.14074**
Support utilisé : 1 CD en double exemplaire

Logibox conservée par l'adhérent : 125529
Logibox conservée par l'APP : 125530

Fait à Paris, le 30/06/2020

Le Président

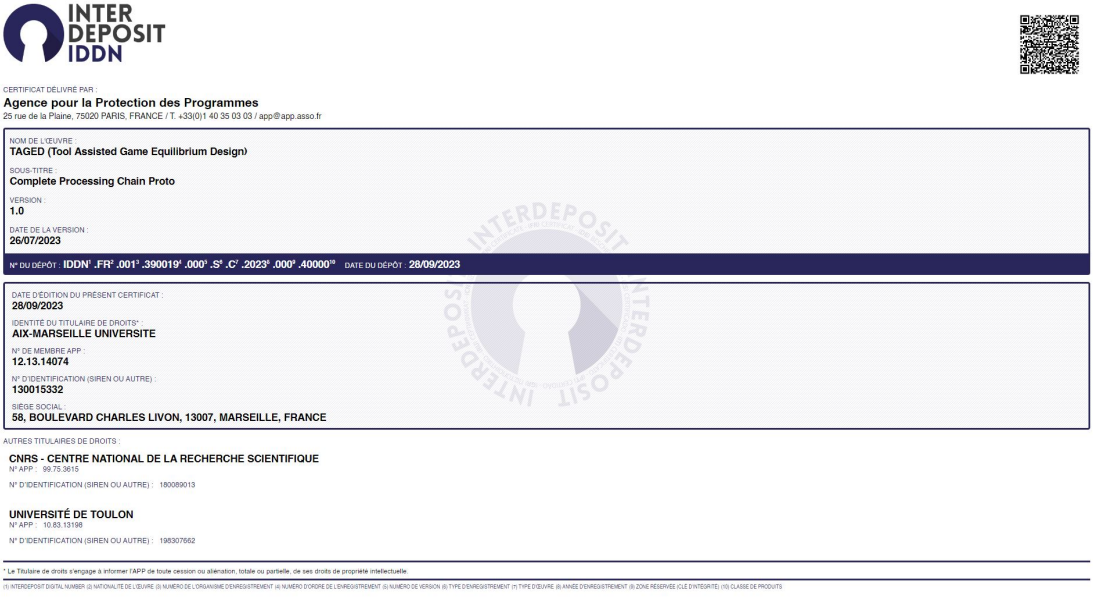
AGENCE POUR LA PROTECTION DES PROGRAMMES

(1) Inter Deposit Digital Number
(2) Nationalité de l'œuvre
(3) Numéro de l'organisme d'enregistrement
(4) Numéro d'ordre de l'enregistrement
(5) Numéro de version
(6) Type d'enregistrement
(7) Type de l'œuvre
(8) Année d'enregistrement
(9) Zone réservée (Clé d'intégrité)
(10) Classe de produits

** Le titulaire s'engage à informer l'APP de toute cession ou aliénation,totale ou partielle, de ses droits de propriété intellectuelle.*
Seules les inscriptions de type S et C permettent un éventuel accès au programme source.

APP.ASSO.FR

FIGURE 8.2. – Dépôt d'IDEA par l'Agence de protection de programmes (APP)



INTER DEPOSIT IDDN

CERTIFICAT DÉLIVRÉ PAR :
Agence pour la Protection des Programmes
25 rue de la Plaine, 75020 PARIS, FRANCE / T. +33(0)1 40 35 03 03 / app@app.asso.fr

NOM DE L'ŒUVRE :
TAGED (Tool Assisted Game Equilibrium Design)

SOUS-TITRE :
Complete Processing Chain Proto

VERSION :
1.0

DATE DE LA VERSION :
26/07/2023

N° DU DÉPÔT : **IDDN¹ .FR² .001³ .390019⁴ .000⁵ .S⁶ .C⁷ .2023⁸ .000⁹ .40000¹⁰** DATE DU DÉPÔT : **28/09/2023**

DATE D'ÉDITION DU PRÉSENT CERTIFICAT :
28/09/2023

IDENTITÉ DU TITULAIRE DE DROITS* :
AIX-MARSEILLE UNIVERSITE

N° DE MEMBRE APP :
12.13.14074

N° D'IDENTIFICATION (SIREN OU AUTRE) :
130015332

SIÈGE SOCIAL :
58, BOULEVARD CHARLES LIVON, 13007, MARSEILLE, FRANCE

AUTRES TITULAIRES DE DROITS :

CNRS - CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
N° APP : 99.75.3615
N° D'IDENTIFICATION (SIREN OU AUTRE) : 180089013

UNIVERSITÉ DE TOULON
N° APP : 10.83.1319R
N° D'IDENTIFICATION (SIREN OU AUTRE) : 198307662

* Le Titulaire de droits s'engage à informer l'APP de toute cession ou aliénation, totale ou partielle, de ses droits de propriété intellectuelle.

(1) INTERDEPOSIT DIGITAL NUMBER (2) NATIONALITÉ DE L'ŒUVRE (3) NUMERO DE L'ORGANISME D'ENREGISTREMENT (4) NUMÉRO D'ORDRE DE L'ENREGISTREMENT (5) NUMÉRO DE VERSION (6) TYPE D'ENREGISTREMENT (7) TYPE D'ŒUVRE (8) ANNÉE D'ENREGISTREMENT (9) ZONE RÉSERVÉE (CLÉ D'INTÉGRITÉ) (10) CLASSE DE PRODUITS

FIGURE 8.3. – Dépôt de TAGED par l'Agence de protection de programmes (APP)

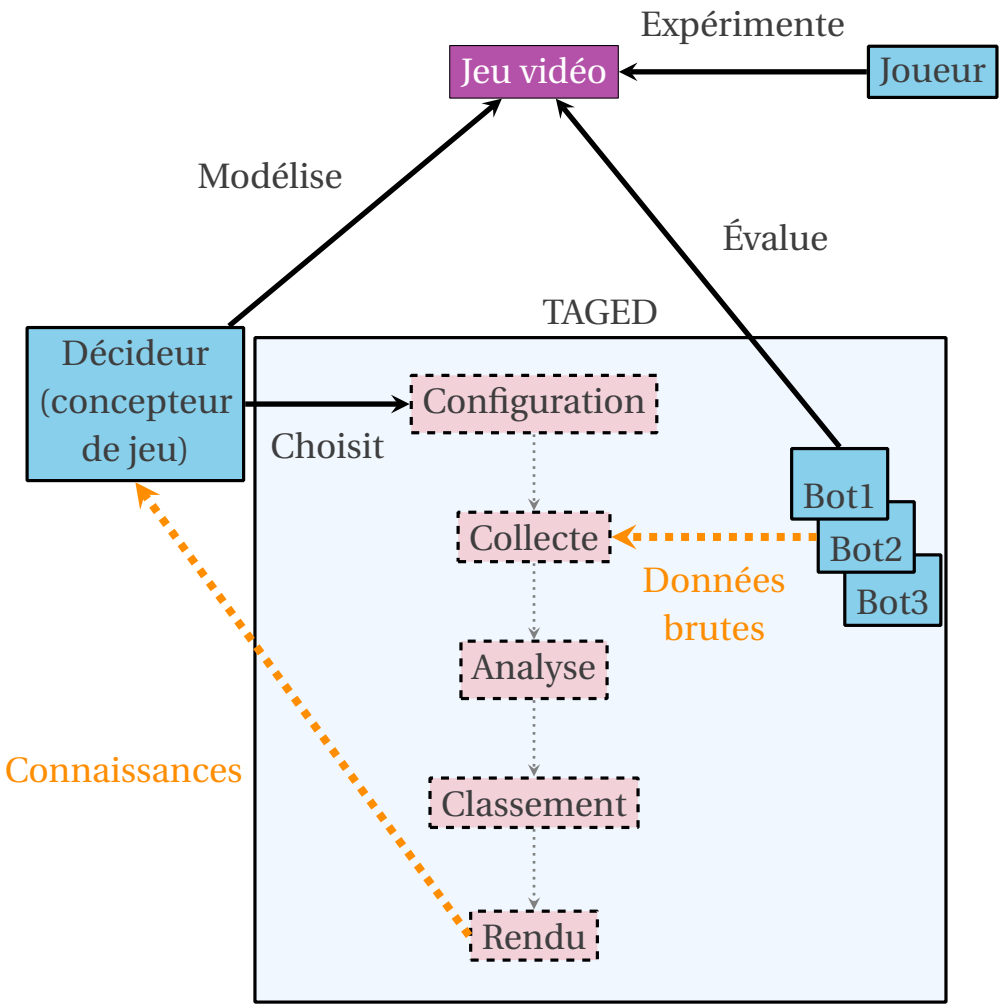


FIGURE 8.4. – Principe de fonctionnement de TAGED

L'évaluation expérimentale permet de valider les déséquilibres et aussi de les quantifier. La méthodologie que nous avons développé permet ainsi au concepteur de jeu d'être plus conscient des biais existant dans son univers vidéoludique et de pouvoir les corriger si les déséquilibres mettent en péril la bonne harmonie de l'expérience de jeu du joueur.

A des fins de modularité et pour en aider à la compréhension, nous proposons un découpage en phase de TAGED, que nous détaillons ensuite (cf. figure 8.4 pour la schématisation du principe de fonctionnement de TAGED, et figure 8.5 pour la présentation détaillée de la méthode TAGED) :

— phase 1 : configuration, phase initiale d'interaction avec le décideur visant à personnaliser l'exécution ;
— phase 2 : collecte, phase de récupération, d'extraction, de nettoyage et de transformation des données d'entrée ;
— phase 3 : analyse, phase principale de filtrage des données, cœur de la méthode avec la génération du Skycube émergent ;
— phase 4 : classement, phase d'ordonnancement des résultats obtenus par l'algorithme DeepSky, des tuples considérés comme les plus efficaces au moins pertinents ;
— phase 5 : rendu, phase de présentation visuelle ou non des résultats obtenus par les calculs de la méthode.

8.4.1. Phase 1 : configuration

La phase de configuration suit les étapes suivantes :

— étape 1 (1.1) : choix des critères et de leurs préférences Skyline ;
— étape 2 (1.2) : calibrage des seuils d'émergence ;
— étape 3 (1.3) : choix du mode de fonctionnement de TAGED ;
— étape 4 (1.4) : saisi de la valeur k pour l'exécution de l'algorithme DeepSky ;
— étape 5 (1.5) : sélection des rendus optionnels.

Pour que la découverte de nouveaux renversements de tendances soit fructueuse, il convient que le cube de données émergent ait une taille exploitable. Or cette dernière dépend de la contrainte d'émergence spécifiée par le décideur.

Dans l'implémentation de TAGED recevant une pré-maturation de la part de la SATT SE, il est prévu de pouvoir choisir les attributs à considérer et les préférences Skyline (`MIN` ou `MAX`) correspondantes (1.1).

Durant la phase de configuration, l'utilisateur peut ajuster la contrainte d'émergence et calibrer les seuils d'émergence afin qu'ils soient pertinents et d'avoir des résultats les plus manipulables possibles (1.2).

Le mode de fonctionnement de la plateforme algorithmique donne le choix de l'algorithme de calcul de cube de données émergent, E-IDEA, F-IDEA ou ℂ-IDEA de la plateforme algorithmique TAGED à utiliser pour l'exécution (1.3).

Le fonctionnement de l'algorithme DeepSky nécessite qu'une valeur soit donnée au nombre k. C'est à cette étape que sa saisie est demandée (1.4).

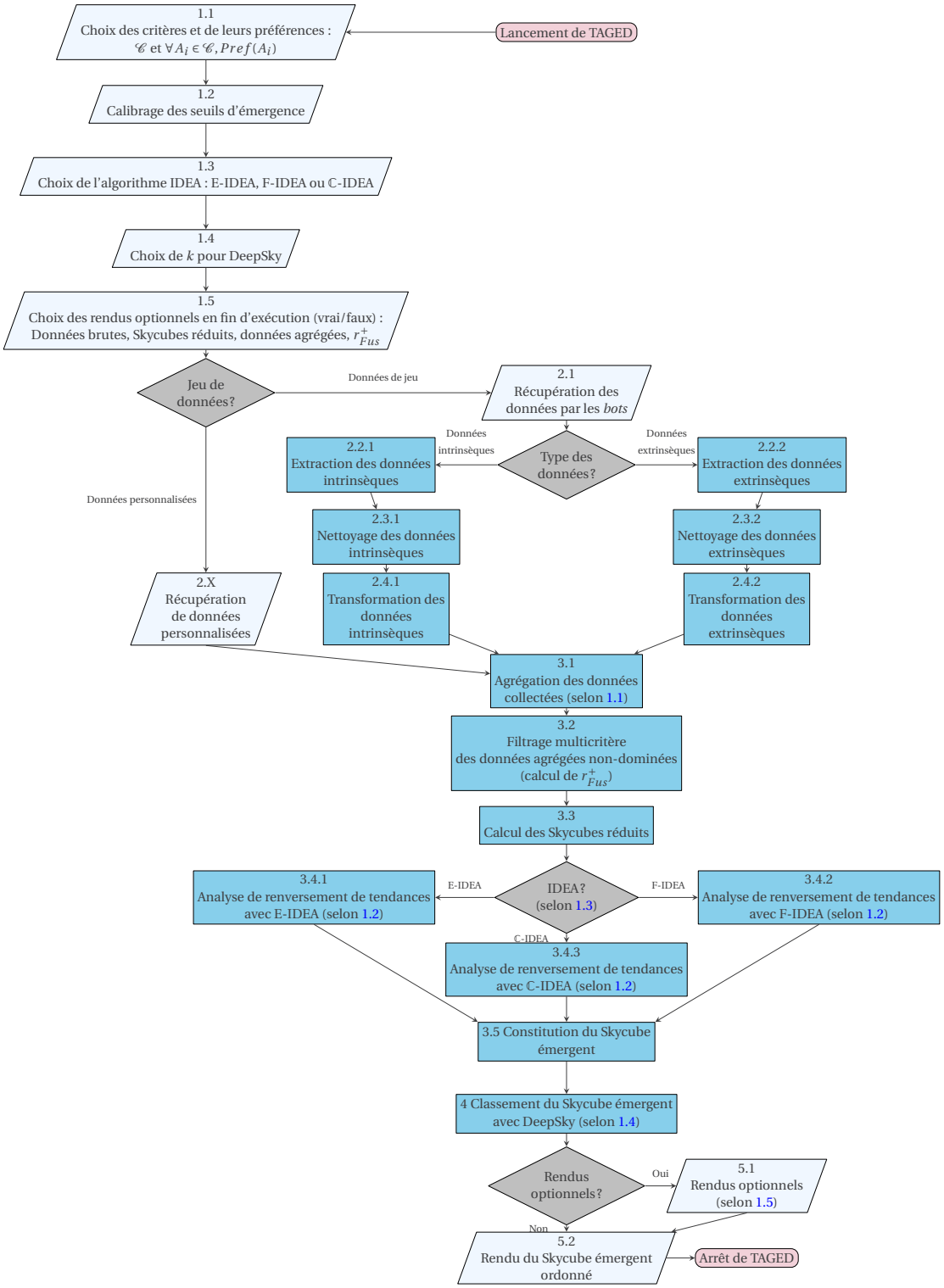


FIGURE 8.5. – Méthode TAGED

À des fins ergonomiques, il est également possible, à cette phase, de signifier comme optionnels les rendus des données calculées intermédiaires (1.5).

Enfin, de manière optionnelle est offert la possibilité pour le concepteur de jeu vidéo de dériver les données en entrée de l'analyse par un jeu de données personnalisé plutôt que par la récupération des données de jeu (2.X). Cette possibilité, initialement imprévue, s'est imposée de manière empirique lors de séances d'expérimentation de l'implémentation développée de concert avec la SATT SE en conditions réelles. Cette possibilité se montre en effet tout à fait favorisante d'itérations rapides, idéales pour les heuristiques et les tests rapides d'intégration de modifications (cf. 8.5.1).

Exemple 8.1 - Nous considérons la phase de configuration de notre exemple telle que décrite ci-après.

Nous choisissons de conserver les attributs dimensions suivants :

— `Tier` : le tier stratégique (cf paragraphe 6.2.1.5 p. 180) de la partie ;
— `Joueur` : la séquence de Pokémon jouée par le joueur ;
— `Adversaire` : la séquence Pokémon jouée par l'adversaire ;
— `Rang` : propriété (utilisée par l'analyse du reversement de tendances) qui catégorise les joueurs en deux classes, les novices (N) et les experts (E), selon le système de classement Elo (cf. sous-sous-section 8.5.2.5) du *ladder* de Pokémon Showdown! (cf. sous-sous-section 6.2.1.2 p. 178).

Afin de déterminer le « meilleur ordre d'apparition de Pokémon dans un combat », nous choisissons également les critères (ou mesures selon le contexte), avec leur préférences Skyline ainsi que leur potentielle invariance, de la manière suivante :

Critère Mesure	Description	Préférence Skyline	Invariant
`Rareté`	Rareté des Pokémon de la séquence jouée par le joueur	`MIN`	✓
`Durée`	Durée, en nombre de tours, du combat	`MIN`	
`Échec`	Taux d'échec de la séquence des Pokémon du joueur par rapport à celle de l'adversaire	`MIN`	

Nous choisissons la contrainte d'émergence suivante :

— `Durée` : $SeuilMin_1 = 25$ et $SeuilMin_2 = 35$;
— `Échec` : $SeuilMin_1 = 30$ et $SeuilMin_2 = 45$;

Nous choisissons le mode de fonctionnement (algorithme de calcul de l'émergence) E-IDEA.

Nous affectons à k la valeur 2.

Nous configurons les rendus optionnels de la manière suivante :

— Données brutes : faux ;
— Données agrégées : vrai ;
— Skycubes réduits : faux ;
— r^{+}_{Fus} : vrai ;

Enfin, nous prenons la décision de ne pas dériver les données brutes d'entrée de TAGED, et de préserver les données tirées de l'expérimentation.

8.4.2. Phase 2 : collecte

La phase de collecte des données d'entrée suit les étapes suivantes :
— étape 1 (2.1) : récupération des nouvelles données d'entrée par des script automatiques ;
— étape 2 (2.2.1 et 2.2.2) : extraction des données intrinsèques au jeu vidéo d'une part, et extrinsèque d'autre part ;
— étape 3 (2.3.1 et 2.3.2) : nettoyage des données brutes ;
— étape 4 (2.4.1 et 2.4.2) : transformation des données pour les préparer à l'analyse.

La phase de collecte s'appuie sur une comparaison à grande échelle des productions de *bots*[3] intégrants une grande partie des connaissances extraites. Ces *bots* effectuent des tâches automatiques à intervalles réguliers pour récupérer les informations mises à jour, les dernières parties jouées... depuis la dernière itération du processus de collecte (2.1). Cette étape a un coût très variable en fonction de la difficulté d'obtention des données d'entrée brutes. D'un coté à l'autre du spectre des possibilités, cela peut aller de la mise en place de mécaniques de rétro-ingénierie jusqu'à la récupération par communication directe, en passant par l'utilisation de sondes logicielles préalablement installées.

Si lors de l'étape d'extraction, les parcours des déséquilibres intrinsèques (2.2.1) et extrinsèques (2.2.2) présentent une méthodologie théorique en apparence similaire, elles nécessitent pourtant des traitements bien spécifiques et distincts. La complexité du traitement des données inhérentes aux mécaniques du jeu vidéo est proportionnelles à la difficulté d'obtention des données d'entrée. En effet, les données d'entrée peuvent être brutes ou même indirectes, ce qui ne manque pas d'occasionner des interventions plus délicates que lorsqu'elles sont disponibles, voire ouvertes, et donc peu ou prou directement réutilisables. Établir des partenariats les plus efficaces possibles avec les concepteurs de jeux vidéo est donc crucial pour atténuer la complexité de ce traitement. Il n'est pas improbable au vu de la combinatoire plus qu'explosive que le traitement des données intrinsèques ne suffise pas à isoler toutes les potentialités de déséquilibres d'un jeu. Les potentiels déséquilibres restants seront généralement plus enfouis et intraitables car uniquement accessibles par une recherche exhaustive. Dans les jeux multijoueur (massivement ou non) en ligne, le nombre d'expériences conduit aléatoirement par les joueurs lors de leur différentes sessions de jeu, fait que la probabilité de découvrir un déséquilibre devient suffisante pour que régulièrement certains soient découverts. Lorsque ces comportements atypiques seront identifiés par les joueurs comme générateur d'une espérance de gain surévaluée, il vont progressivement se répandre et leur occurrence deviendra de plus en plus fréquente. Nous cherchons dans ce cas à détecter ces changements de dynamique des signaux faibles dans les comportements des joueurs.

3. Dans le contexte des jeux vidéo et de l'informatique, un *bot* (diminutif de *robot*) est un programme, ou script, conçu pour effectuer des actions automatisées en autonomie. Les *bots* peuvent accomplir des tâches diverses telles que l'exécution de commandes répétitives, la collecte de ressources, la réalisation de calculs complexes, ou encore la simulation du comportement d'un joueur. Les *bots* peuvent être utilisés de manière légitime ou illégitime.

Le nettoyage des données (également appelé *data cleaning*, *data cleansing* ou encore *data scrubbing*) est un processus aussi classique qu'incontournable pour améliorer la qualité des données et corriger l'ensemble de données d'entrée. C'est d'ailleurs une des problématiques principales des entrepôts de données (RAHM et DO 2000). Notre proposition ne fait pas exception en utilisant une étape de nettoyage des données (2.3.1 et 2.3.2), et il important de rappeler que la nature même (fortement ou faiblement corrélées) et le volume des données à traiter est un paramètre crucial pour la suite de la méthode, comme il l'est d'ailleurs le plus souvent en fouille de données.

A l'issue de l'étape de nettoyage, les données sont cohérentes et une étape de transformation (aussi appelée *wrangling* ou *muging*) vise à convertir les données brutes dans un format adapté à l'analyse (2.4.1 et 2.4.2).

Comme pour l'étape d'extraction, celles de nettoyage et de transformation sont distinctes suivant qu'il s'agit de données intrinsèques ou extrinsèques.

8.4.3. Phase 3 : analyse

La phase d'analyse est la plus abondante et la plus importante de TAGED. Elle suit les étapes suivantes :

— étape 1 (3.1) : agrégation de données dans l'entrepôt de données ;
— étape 2 (3.2) : filtrage multicritère des données non-dominées par matérialisation partielle du Skycube avec le treillis des concepts Skylines ;
— étape 3 (3.3) : fusion et dédoublonnage des Skycuboïdes en une seule relation ;
— étape 4 (3.4.1, 3.4.2 et 3.4.3) : extraction de renversement de tendances par le calcul du Skycube de données émergent selon l'algorithme choisi ;
— étape 5 (3.5) : fusion et formatage des données atypiques pour prédiction.

L'agrégation des données issues de la phase de collecte forment un entrepôt de données qui nous est utile dans la phase d'analyse pour la génération des cubes de données (3.1). Certaines données agrégées sont aussi transformées, ou transposées, le plus souvent en numérisant des données sémantiques pour les besoins de l'analyse.

Exemple 8.2 - Pour notre exemple, la phase de collecte nous donne la relation exemple `Pokémon` (cf. tableau 8.1). Lors de l'agrégation des données, nous confondons les combats miroirs, qui sont peu pertinent de part leur caractéristiques de données symétriques, et pour l'exemple, nous nous limitons à trois Pokémon par séquence, qu'elle soit du joueur ou de l'adversaire, et afin de simplifier leur usage dans leurs représentations, nous les indiquons par des listes de numéros de Pokémon, et leur affectons une lettre pour les désigner. Ainsi, la liste $121,113,006(A)$ correspond à la séquence des trois Pokémon Staross, Leveinard et Dracaufeu. Pour la lisibilité, nous arrondissons par un multiple de cinq les durées des combats et les taux d'échecs.

La méthodologie que nous développons s'appuie sur la notion de dominance de Pareto pour définir si une stratégie en domine une autre. De cette manière, nous prolongeons les précédents résultats de l'équipe en matière de Skycube. L'étape fondamentale de filtrage multicritère des données non-dominées (3.2) est proposée selon une représentation réduite sans perte d'information, avec un coût minimal à obtenir, à

TABLEAU 8.1. – La relation exemple `Pokémon`

RowId	Tier [a]	Joueur [b]	Adversaire [c]	...	Rang [d]	Rareté [e]	Durée [f]	Échec [g]
1	*UU*	121,113,006(*A*)	065,113,143(*D*)	...	*N*	5	25	30
2	*OU*	065,103,065(*B*)	065,040,065(*E*)	...	*N*	4	65	50
3	*OU*	065,103,065(*B*)	121,113,121(*F*)	...	*N*	4	35	40
4	*OU*	065,103,065(*B*)	121,113,006(*A*)	...	*N*	4	85	40
5	*OU*	121,113,080(*C*)	121,113,006(*A*)	...	*N*	1	95	60
6	*OU*	121,113,080(*C*)	065,103,065(*B*)	...	*N*	1	35	50
7	*OU*	065,113,143(*D*)	065,103,065(*B*)	...	*N*	9	85	60
8	*OU*	065,113,143(*D*)	121,113,080(*C*)	...	*N*	9	85	70
9	*UU*	065,040,065(*E*)	065,113,143(*D*)	...	*N*	7	25	50
10	*UU*	065,040,065(*E*)	065,040,065(*E*)	...	*N*	7	65	30
11	*UU*	121,113,006(*A*)	065,113,143(*D*)	...	*E*	5	20	30
12	*OU*	065,103,065(*B*)	065,040,065(*E*)	...	*E*	4	60	45
13	*OU*	065,103,065(*B*)	121,113,121(*F*)	...	*E*	4	30	30
14	*OU*	065,103,065(*B*)	121,113,006(*A*)	...	*E*	4	80	50
15	*OU*	121,113,080(*C*)	121,113,006(*A*)	...	*E*	1	90	70
16	*OU*	121,113,080(*C*)	065,103,065(*B*)	...	*E*	1	30	30
17	*OU*	065,113,143(*D*)	065,103,065(*B*)	...	*E*	9	80	50
18	*OU*	065,113,143(*D*)	121,113,080(*C*)	...	*E*	9	90	70
19	*UU*	065,040,065(*E*)	065,113,143(*D*)	...	*E*	7	20	30
20	*UU*	065,040,065(*E*)	065,040,065(*E*)	...	*E*	7	60	45

a. Tier stratégique.

b. Avec les numéros officiels des Pokémon (et leur tier stratégique) :
— n°006 : Dracaufeu (UU) ;
— n°040 : Grodoudou (UU) ;
— n°065 : Alakazam (OU) ;
— n°080 : Flagadoss (OU) ;
— n°103 : Noadkoko (OU) ;
— n°113 : Leveinard (OU) ;
— n°121 : Staross (OU) ;
— n°143 : Ronflex (OU).

c. *idem supra.*

d. Catégorie de joueur, avec seulement deux partitions considérées : novice (N) et expert (E).

e. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de `Rareté` r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

f. En nombre de tours de combat au total.

g. En pourcentage.

enregistrer et à gérer (cf. section 6.3 p. 188). L'unique matérialisation des Skycuboïdes fermés, ou concepts Skylines, est une opération tout aussi essentielle, qui aboutie à la matérialisation partielle du Skycube.

Une fois les Skycubes réduits obtenus, nous avons alors l'étape de fusion et d'abrègement afin d'obtenir une seule relation, r^{+}_{Fus} (3.3).

Exemple 8.3 - La relation fusionnée abrégée est donnée dans le tableau 8.2 (avec $\texttt{D}_1$ pour $\texttt{Durée}_1$, $\texttt{É}_1$ pour $\texttt{Échec}_1$, $\texttt{D}_2$ pour $\texttt{Durée}_2$ et $\texttt{É}_2$) pour $\texttt{Échec}_2$). Les séparations présentent dans le tableau de la relation fusionnée sont préservées, elles délimitent les tuples issus des Skycuboïdes unicompatibles des deux Skycubes d'entrée. Cette relation qui sert d'entrée à l'algorithme de la plateforme IDEA choisi comme mode de fonctionnement lors de l'étape de configuration.

TABLEAU 8.2. – Relation fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$

RowId	Tier	Joueur	Adversaire	...	$\texttt{D}_1$ [a]	$\texttt{É}_1$ [b]	$\texttt{D}_2$ [c]	$\texttt{É}_2$ [d]
1	*UU*	*A*	*D*	...	25	30	20	30
2	*OU*	*B*	*F*	...	35	40	*ALL*	*ALL*
3	*OU*	*C*	*B*	...	35	50	30	30
4	*UU*	*A*	*D*	...	*ALL*	*ALL*	20	30
5	*UU*	*E*	*D*	...	*ALL*	*ALL*	20	30

a. En nombre de tours de combat au total.
b. En pourcentage.
c. *idem* $\texttt{D}_1$.
d. *idem* $\texttt{É}_1$.

L'étape suivante concerne l'analyse et l'extraction de tendances (cf. section 6.4 p. 205). Précédemment à ce travail de recherche, les mises en œuvre de calcul du cube de données émergent étaient réalisées selon un seul attribut mesure évoluant selon un critère décisionnel, et donc ne pouvant correspondre qu'à une seule préférence Skyline. Avec l'usage combiné de l'analyse décisionnelle multicritère et la découverte de renversements de tendances, il devient nécessaire d'en utiliser plusieurs, et potentiellement selon des préférences hétérogènes, afin que la recherche de dominance de Pareto soit propice. Le calcul des cubes de données émergents est effectué par l'un des algorithmes de la plateforme IDEA, E-IDEA, F-IDEA ou $\mathbb{C}$-IDEA (3.4.1, 3.4.2 et 3.4.3).

Exemple 8.4 - Avec la relation exemple fusionnée abrégée $\texttt{Pokémon}^{+}_{Fus}$, le tableau 8.3 donne le Skycube émergent. Les séparations du tableau de la relation $\texttt{Pokémon}^{+}_{Fus}$ sont préservées par commodité.

Comme pour la phase de collecte, TAGED comprend, en fin du processus d'analyse, une étape d'adaptation, de composition, de fusion et de formatage des données informatives obtenues afin qu'elles constituent le cube de données émergent multicritère (3.5) et qu'elles puissent servir à des prédictions, ainsi qu'être directement ordonnées

TABLEAU 8.3. – Skycube émergent de la relation $\texttt{Pokémon}^{+}_{Fus}$

Tuple émergent	$ER_{\texttt{Durée}}$	$ER_{\texttt{Échec}}$
$(ALL, ALL, D, \ldots)$	1.25	1
$(ALL, ALL, B, \ldots)$	1.17	1.67
$(ALL, A, ALL, \ldots)$	1.25	1
$(ALL, C, ALL, \ldots)$	1.17	1.67
$(UU, ALL, ALL, \ldots)$	1.25	1
$(OU, ALL, ALL, \ldots)$	2.33	3
$(ALL, A, D, \ldots)$	1.25	1
$(ALL, C, B, \ldots)$	1.17	1.67
$(UU, ALL, D, \ldots)$	1.25	1
$(OU, ALL, B, \ldots)$	1.17	1.67
$(UU, A, ALL, \ldots)$	1.25	1
$(OU, C, ALL, \ldots)$	1.17	1.67
$(UU, A, D, \ldots)$	1.25	1
$(OU, C, B, \ldots)$	1.17	1.67
$(ALL, ALL, F, \ldots)$	∞	∞
$(ALL, ALL, B, \ldots)$	∞	∞
$(ALL, B, ALL, \ldots)$	∞	∞
$(ALL, C, ALL, \ldots)$	∞	∞
$(OU, ALL, ALL, \ldots)$	∞	∞
$(ALL, B, F, \ldots)$	∞	∞
$(ALL, C, B, \ldots)$	∞	∞
$(ALL, E, D, \ldots)$	∞	∞
$(OU, ALL, F, \ldots)$	∞	∞
$(OU, ALL, B, \ldots)$	∞	∞
$(UU, A, ALL, \ldots)$	∞	1
$(OU, B, ALL, \ldots)$	∞	∞
$(OU, C, ALL, \ldots)$	∞	∞
$(OU, B, F, \ldots)$	∞	∞
$(OU, C, B, \ldots)$	∞	∞
$(UU, E, D, \ldots)$	∞	∞

dans la phase de classement, et présentées de la manière souhaitée lors de la phase de rendu.

8.4.4. Phase 4 : classement

Cette phase offre un classement des données, des plus pertinentes à celles qui le sont le moins, grâce à l'utilisation de l'algorithme DeepSky (4) caractérisé par l'utilisation de la méthode d'ordonnement CoSky et le principe multiniveaux (cf. sous-section 7.3.3 p. 222), ainsi que son nombre k, déterminé lors la phase de configuration (1.4).

Le classement des données permet de les organiser de manière logique et cohérente, ce qui facilite la recherche, la récupération et la gestion des informations obtenues par TAGED, et permet donc d'accéder plus rapidement aux informations souhaitées. Les informations peuvent être ainsi analysées plus efficacement, permettre une prise de décision éclairée et favoriser un partage ou une communication optimisée, pour un usage direct ou un nouveau projet d'analyse.

TAGED offre la possibilité d'avoir un classement Skyline pour chacun des deux Skycubes calculés et à partir du Skyline reconstitué du Skycube émergent afin d'élargir la palette de données décisionnelles du concepteur de jeu vidéo. Les calculs peut être réalisés, à la phase de génération des deux Skylines car la recherche du renversement de tendances n'influe pas sur le classement des Skylines.

Exemple 8.5 - Les résultats de classement obtenus par application de l'algorithme DeepSky, avec $k = 2$, pour le Skyline des joueurs novices sont donnés au tableau 8.4. Ceux du Skyline des joueurs experts sont donnés, quant à eux, au tableau 8.5. En sortie de la méthode TAGED, avec E-IDEA, nous obtenons les tuples fermés émergents ordonnés, avec les mesures, au tableau 8.6. Selon l'exemple choisi, le combat de Pokémon Staross, Leveinard et Flagadoss contre Alakazam, Noadkoko et Alakazam est la combinaison la plus performante si l'on considère à la fois les joueurs novices et ceux experts.

TABLEAU 8.4. – Classement du Skyline des joueurs novices

RowId	Rareté	Durée	Échec	Score
6	1	35	50	0.948
3	4	35	40	0.886

TABLEAU 8.5. – Classement du Skyline des joueurs experts

RowId	Rareté	Durée	Échec	Score
6	1	30	30	0.980
1	5	20	30	0.883

TABLEAU 8.6. – Skycube émergent ordonné de la relation $\texttt{Pokémon}^{+}_{Fus}$

Ordre	Tuple émergent [a]	R [b]	$\texttt{D}_1$ [c]	$\texttt{É}_1$ [d]	$\texttt{D}_2$ [e]	$\texttt{É}_2$ [f]	$ER_{\texttt{Durée}}$	$ER_{\texttt{Échec}}$
1	$(OU, C, B, \ldots)$	5	25	40	20	30	1.17	1.67
2	$(UU, A, D, \ldots)$	1	35	30	30	30	1.25	1

a. Avec :
— A : 121, 113, 006 (Staross, Leveinard, Dracaufeu) ;
— B : 065, 103, 065 (Alakazam, Noadkoko, Alakazam) ;
— C : 121, 113, 080 (Staross, Leveinard, Flagadoss) ;
— D : 065, 113, 143 (Alakazam, Leveinard, Ronflex).

b. Soit p le pourcentage d'obtention d'une séquence de Pokémon (qui est la multiplication du pourcentage d'obtention de chaque Pokémon de la séquence), le score de `Rareté` r est calculé, sur une échelle allant de 0 à 10, de la manière suivante : si $p = 1$ alors $r = 0$, sinon $r = \lfloor max(\frac{(p-1)\times 10-(100-e\times 0.9)}{e}, 0)\rfloor + 1$.

c. En nombre de tours de combat au total.

d. En pourcentage.

e. *idem* $\texttt{D}_1$.

f. *idem* $\texttt{É}_1$.

8.4.5. Phase 5 : rendu

Cette phase est consacrée au rendu des données calculées lors de chacune des phases précédentes. Tout ou partie des produits générés suivants sont concernés :
— données brutes nettoyées et conditionnées (données d'entrée de la phase d'analyse), optionnel (5.1) ;
— données agrégées (données d'entrée du calcul des Skycubes), optionnel (5.1) ;
— Skycubes avec réduction, optionnel (5.1) ;
— Skycuboïdes fusionnés et agrégés sous forme de relation (donnée d'entrée du calcul du Skycube de données émergent), optionnel (5.1) ;
— données du Skycube émergent ordonné, ou données de sortie de TAGED (5.2).

Les connaissances obtenues peuvent être présentées, ou visualisées, pour consultation directe ou enregistrées pour communication (pour les besoins d'une analyse ultérieure par exemple). Il est tout à fait envisageable de proposer les résultats sous forme de service Web [4] dans une version de TAGED de TRL de niveau supérieur à 5. Les données brutes nettoyées et conditionnées ainsi que les données agrégées ne sont pas, en principe, affichables de manière satisfaisantes du fait du fort volume vraisemblable de l'ensemble des données d'entrée.

4. Un service Web est un système informatique permettant la communication et l'échange de données entre différentes applications ou plateformes via le Web ou plus largement Internet. Il fournit des fonctionnalités spécifiques et des services accessibles à distance, permettant aux utilisateurs de partager, d'accéder et de manipuler des données de manière sécurisée et interopérable. Les services Web sont largement utilisés pour la création d'applications distribuées et la construction d'architectures logicielles flexibles et évolutives.

8.4.6. Plateforme algorithmique TAGED

Dans cette sous-section, nous nous intéressons à l'adaptation du schéma algorithmique de la plateforme IDEA pour le calcul du Skycube émergent, puis d'un classement avec la méthode CoSky. À l'exception des phases de configuration, de collecte et de classement, la plateforme algorithmique TAGED est une adaptation de la plateforme algorithmique IDEA avec une phase d'analyse abondée par la recherche de dominance de Pareto multidimensionnelle au niveau des Skycubes, et une phase rendu, également légèrement modifiée pour prendre en compte la représentation de l'ensemble des informations obtenues, et notamment des données du Skycube émergent ordonné. Ainsi, de manière semblable à la plateforme IDEA, la plateforme algorithmique TAGED peut être déclinée selon son mode de fonctionnement en TAGED-E, TAGED-F ou TAGED-ℂ. TAGED utilise les mêmes algorithmes que IDEA pour son noyau (cf. sous-sous-section 3.4.2.1 p. 103). En revanche, le reste du moteur peut être relativement variable d'une implémentation à une autre, en fonction de la classification du jeu vidéo. Ainsi, il ne nous semble pas explicite, ni même pertinent, de tenter de fournir, actuellement, un algorithme précis d'une méthode (cf. figure 8.5) qui tire justement sa force de son adaptabilité et qui connait déjà trois implémentations différentes de TRL 5 minimum réalisées en collaboration avec la SATT SE et jouissant d'une déclaration d'invention, pour trois dispositifs ludiques aux *gameplays* largement populaires et hétérogènes.

8.5. Applications aux jeux vidéo

Si le *game design* est le domaine d'application direct de TAGED comme nous avons pu le voir dans ce manuscrit, d'autres domaines sont envisageables ou possibles comme l'*e-sport*, la mercatique et les stratégies commerciales du jeu vidéo, la publicité vidéo-ludique ciblée…. Après avoir décrit les apports de TAGED pour le *game design*, nous explorons, en tant qu'ouverture, brièvement la sphère de l'*e-sport*. Nous considérons qu'il s'agit d'un domaine de potentialités qui nous paraissent particulièrement prometteuses.

Nous mettons à disposition, en annexe, la présentation des dispositifs ludiques que nous avons conçu et développés par ailleurs et qui auraient pu tirer partie de la solution TAGED, comme Genius (cf. section A.1 p. 292), ou qui en profiteront certainement à l'avenir, comme Unplugged (cf. sous-section A.2 p. 296).

8.5.1. Applications au game design

Les connaissances acquises par à la méthode TAGED peuvent être prises en compte par le décideur lors de la conception, ou même plus tard, sous forme de modifications à apporter ou d'éléments à prendre en compte dans un contexte hautement combinatoire ou, notamment comme démonstrateur, pour des caractéristiques ou dimensions mesures telles que la rareté. En effet, comme nous l'avons vu, pour un jeu comme les échecs, bien que jouer les pions blancs présentent une stratégie dominante,

le déséquilibre correspondant est acceptable du fait de l'importante imprévisibilité des situations de jeu causées par le grand nombres de possibilités d'action, quoique variables, offertes à chaque coup. Par ailleurs, un élément de jeu connu, ou prévu, comme ayant un impact « supérieur » à la moyenne peut, malgré le fait qu'il déséquilibre le jeu vidéo en le transformant en dispositif ludique à somme non nulle, être l'expression d'une volonté du concepteur de jeu, comme un *boss*[5], ou une récompense, intrinsèque ou extrinsèque au joueur, particulièrement exceptionnelle. Dans ce cas, l'élément ludique doit être mis à disposition de conditions contextuelles ou déclenchantes strictes. Son obtention doit être maîtrisée afin d'éviter que le déséquilibre recherché puisse être perçu injustement par les joueurs. Pour autant, cela traduit invariablement une situation délicate qui demande un traitement spécifique, et ainsi, dans un jeu vidéo de combat par exemple, il est fréquent qu'un *boss* ne soit pas sélectionnable dans un mode de combat un contre un, ou duel, ou encore à plus forte raison, qu'il soit interdit dans un règlement de compétition (cf. sous-section 8.5.2).

La phase de configuration offre la possibilité de fournir ses propres jeux de données en entrée de l'analyse, et la phase de rendu permet, quant à elle, de récupérer, dans un format de travail adapté, les données du jeu réelles. Ainsi, il est, de manière immédiate, possible d'apporter dans un jeu de données réelles de faibles modifications issues de l'analyse mathématique ou empirique des connaissances obtenues en sortie de TAGED, et de leur classement, afin de tester les hypothèses et les propositions d'améliorations du concepteur de jeu, y compris de manière incrémentale au besoin. Cette fonctionnalité offre une flexibilité et une aisance au décideur en lui procurant un gain de temps considérable et une assistance précise à sa justesse décisionnelle.

De même, il est ainsi possible efficacement de décider des briques de *gameplay* entières (cf. sous-sous-section 4.4.2.4 p. 128) à ajouter ou à supprimer. Les ajustements fins et à haute vélocité ainsi permis par notre méthode sont un besoin couramment exprimé par les concepteurs de jeux vidéo.

Exemple 8.6 - Pour rappel, nous estimons que le cœur du *gameplay* du jeu vidéo Pokémon Showdown! actuel, donc avant équilibrage, est composé des briques de *gameplay* `Atteindre` (`At`), `Éviter` (`Ev`), `Détruire` (`Dt`), `Gérer` (`Gr`), `Choisir` (`Ch`) et `Déplacer` (`Dp`) (cf. figure 8.6). En nous plaçant un instant dans le rôle du concepteur de jeu de Pokémon Showdown!, pour l'exemple, et en considérant la relation exemple `Pokémon` (cf. tableau 8.1) ainsi que son Skycube émergent (cf. tableau 8.6) calculé grâce à la méthode TAGED, nous pouvons décider des connaissances que nous en retirons que le Pokémon Alakazam (n°065) est trop présent dans les séquences de Pokémon aussi bien du joueur que de l'adversaire, et que ses caractéristiques pourraient recevoir une révision à la baisse. Cela nous semble tout à fait envisageable, et l'utilisation de TAGED permettrait des itérations rapides favorisant des tests d'équilibrage empiriques. Cependant, nous pourrions aussi considérer que ce Pokémon, et d'autres semblables, auraient une puissance relative moins importante en inaugurant une nouvelle manière d'utiliser les capacités des Pokémon, comme par exemple avec le jeu

5. Dans un jeu vidéo, un *boss* est un ennemi majeur, souvent unique, généralement rencontré à la fin d'un niveau.

vidéo Pokémon Ranger où il est nécessaire de réussir des mouvements de précision pour les employer avec succès. Ainsi, nous pourrions y parvenir en ajoutant la brique de *gameplay* `Shooter (Sh)` (cf. figure 8.7, avec la brique de *gameplay* en écriture rouge) et tester l'équilibrage en lançant une nouvelle exécution d'une implémentation de TAGED.

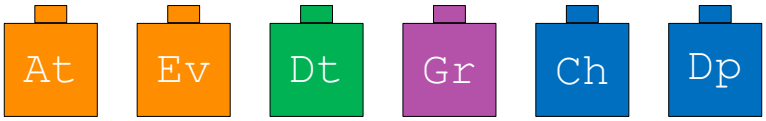


FIGURE 8.6. – Briques *gameplay* de Pokémon Showdown ! : avant équilibrage

FIGURE 8.7. – Briques *gameplay* de Pokémon Showdown ! : après équilibrage

TAGED est une solution statistique d'aide à l'équilibrage méthodique fréquemment empirique. Pour autant, certains concepteurs de jeu vidéo peuvent choisir, suivant la classification du jeu vidéo, une approche plus mécanique ou même mathématique, en utilisant la théorie des jeux dont l'équilibre de Nash par exemple (cf. sous-section 4.6.2 p. 139). Certains font aussi appel à des *bots* pour un équilibrage dynamique automatique, notamment de la conception des niveaux de jeu, et en particulier quand le nombre de niveaux est conséquent. Dans tous ces cas d'utilisation, TAGED est adapté. L'intérêt et la précision des connaissances extraites dépend directement des données fournies en entrée, qu'elles soient issues de l'expérimentation ou d'un modèle théorique. Le plus souvent, ces décisions ne relèvent pas du concepteur de jeu vidéo, ou de TAGED, mais du jeu vidéo lui-même et de ses caractéristiques, et notamment de la classification de son moteur de jeu et de son *gameplay* (cf. section 4.5).

8.5.2. Applications à l'e-sport

8.5.2.1. Contexte et enjeux de l'e-sport

Depuis ses balbutiements dans les laboratoires du Massachusetts Institute of Technology (MIT) et la création par trois étudiants de SpaceWar! en 1962, le jeu vidéo offre aux joueurs la possibilité de se confronter entre eux par le biais d'un programme informatique (BESOMBES 2016). La toute première compétition *e-sport* (ou sport électronique) du monde s'est déroulée en 1972 sur le jeu vidéo SpaceWar! lors de l'Intergalactic Space War Olympic de Stanford, et proposait au grand gagnant de remporter un an d'abonnement au magazine Rolling Stone. Dans la décennie qui s'ensuivit, les salles d'arcade se popularisèrent et vinrent constituer les points de rencontre des premières communautés de joueurs compétitifs. Les années 1980 marquent une

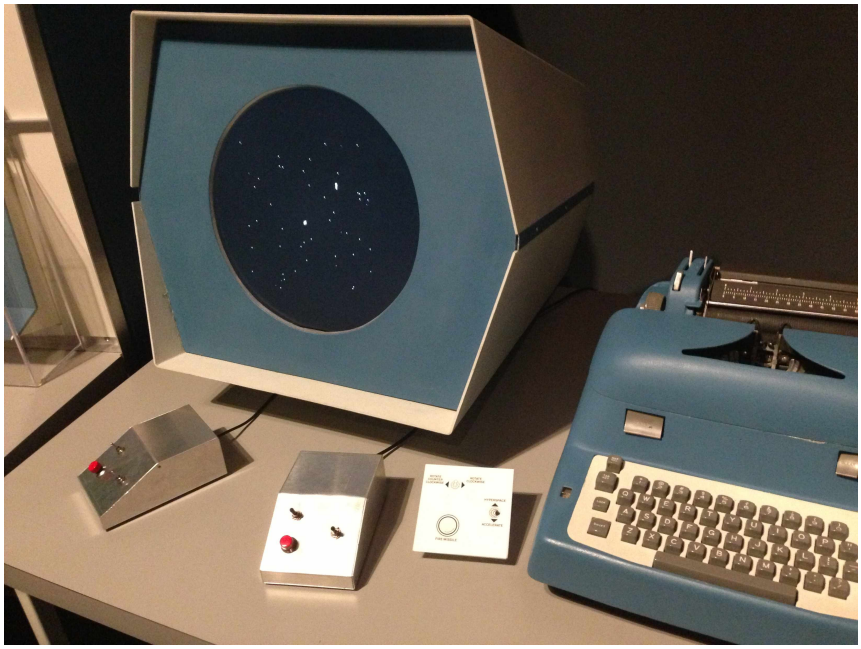


FIGURE 8.8. – SpaceWar! (1962), premier jeu vidéo compétitif [a]

a. Photographie prise au Museum of the Moving Image par Suzanne DeChillo du New York Times.

véritable révolution avec l'arrivée des consoles domestiques et le développement de véritables communautés de joueurs. En 1997, convaincu de l'intérêt majeur que pouvait avoir les rencontres compétitives de jeux vidéo, John Catmack met en jeu sa propre voiture Ferrari 328 GTS lors du tournoi Red Annihilation sur le jeu vidéo Quake. C'est alors le début des classements mondiaux, mais aussi l'émergence des premières célébrités du milieu qui deviendront plus tard les premiers joueurs professionnels [6]. Après l'arcade et les consoles de salon, c'est au tour des consoles portables puis des téléphones intelligents de rejoindre les plateformes de jeu de la sphère qui devient peu à peu l'*e-sport*.

Loin des tournois peu ou prou amateurs des années 1970 à 1990, les compétitions d'*e-sport* se sont par la suite professionnalisées et devenues lucratives. L'*e-sport* consiste en l'organisation d'affrontements codifiés entre joueurs lors de compétitions nationales et internationales de jeux vidéo. En empruntant au modèle sportif traditionnel certaines de ses caractéristiques (structuration d'équipes, mise en place de calendriers spécifiques et de classements, enregistrements de records, remises de coupes, parrainage, « *starification* » ou « spectacularisation »…), le milieu de l'*e-sport* cherche ainsi à professionnaliser sa pratique et ses acteurs depuis la fin des années 1990 (MORA et HÉAS 2003). Le processus de « sportification » [7] (PARLEBAS 1999) que

6. Un joueur professionnel, ou *pro gamer* en anglais, qualifie un personne gagnant de quoi subvenir à ses besoin vitaux en jouant aux jeux vidéo et ayant fait de cette pratique sont activité principale.

7. La « sportification » (ou « sportivisation ») est décrit comme le processus social et institutionnel par lequel une activité ludomotrice acquiert le statut de sport. Elle consiste à (HEERE 2017) :

— considérer, organiser ou réglementer une activité non sportive de manière à ce qu'elle soit

FIGURE 8.9. – Salle d'arcade en Corée du Sud [a]

a. Photographie prise à Seoul en Corée du Sud par Ciaran O'Brien (https://unsplash.com/@icidius) sur Unsplash (https://unsplash.com).

connaissent les jeux vidéo compétitifs bénéficie depuis ses débuts d'une très forte popularisation en Asie, et notamment en Corée du Sud (PABERZ 2012). Depuis les années 2010, l'*e-sport* semble sortir du cercle confiné de ses adeptes pour toucher un public également occidental plus nombreux et de plus en plus diversifié, y compris en âge et en sexe. Longtemps considérés comme réservés à une niche, l'*e-sport* est passé d'une sous-culture stigmatisée par les médias à des évènements suivis par des millions de personnes dans le monde entier dans une croissance fulgurante (TRÉMEL LAURENT FORTIN 2009).

Les jeux vidéo contemporains, tels que ceux utilisés pour l'*e-sport*, sont le plus souvent sociaux et coopératifs. Ils se déroulent également presque exclusivement dans la même temporalité pour l'ensemble des compétiteurs, pour toute catégorie de compétitions et de ligues. Les types de jeux vidéo concernés couvrent l'ensemble des briques *gameplay* actuellement recensées (cf. sous-sous-section 4.4.2.4 p. 128), leur aspect compétitif ne semble pas lié à leurs mécaniques de jeu, pas plus qu'à leur genre ou leur typologie bien qu'ils aient tout de même tous en commun la catégorie *agôn* de Caillois (cf. sous-section 4.4.2 p. 126).

La popularité grandissante des évènements d'*e-sport* induisent des enjeux économiques de plus en plus importants. À ses débuts, l'*e-sport* proposait des récompenses

considérable comme un sport et qu'elle offre un environnement équitable, agréable et sûr permettant aux individus de concourir, de coopérer, et de comparer leurs performances entre eux, ainsi que leurs performance futures et passée ;

— ajouter une composante sportive à une activité existante afin de la rendre plus attrayante pour son public.

FIGURE 8.10. – Tournoi d'*e-sport*[a]

a. Photographie montrant la scène et la foule d'une compétition *e-sport*, prise au KeyArena de Seattle pour The International 2014 par Jakob Wells sur Wikipedia.

(souvent appelées *cash prizes*) pouvant être considérées comme anecdotiques, à l'instar d'une version spéciale (*collector* le plus souvent) du jeu vidéo lui-même, ou un prix de seulement quelques dizaines voire centaines d'euros. Lors de l'écriture de ce mémoire, le montant pour une seule compétition peut atteindre des millions, voire des dizaines de millions d'euros[8], ce qui reflète par ailleurs un engouement important pour l'*e-sport*. Du côté des entrées, les organisateurs peuvent compter sur la billetterie, en présence aussi bien qu'en ligne, les revenus liés à l'audience, y compris des réseaux sociaux numériques, l'engagement engrangé par les influenceurs, les dispositifs publicitaires, le parrainage ainsi qu'à la vente des jeux vidéo elle-même. Les marchés émergents ont prit rapidement une immense ampleur, dans lesquels œuvre notamment les acteurs historiques, profitant de l'intérêt de nouveaux paradigmes d'exploitation des jeux vidéo. Dorénavant, ces évènements n'ont plus rien à envier aux compétitions sportives traditionnelles, ils sont diffusés sur des dizaines de solutions de diffusion différentes en des dizaines de langues.

8.5.2.2. Le statut de l'e-sport

Nous déconseillons de se poser la question de savoir si l'*e-sport* peut être qualifié, partiellement ou à part entière, de sport. Le sport lui-même est un phénomène polymorphe ambigu indéfinissable et toute tentative de l'appréhender clairement semble être voué à l'échec qui en limiterait la compréhension (MCBRIDE 1975).

Indépendamment de la question de savoir si l'*e-sport* peut être considéré comme

8. cf. https://www.esportsearnings.com/.

un sport, il devrait être examiné dans le cadre de la gestion du sport parce qu'il est une manifestation de « sportification » (HEERE 2017).

Par ailleurs, si les juristes se souciaient jusqu'alors du sport, l'*e-sport* demeurait exsangue. En travaillant avec le Centre de droit du sport d'Aix-Marseille Université et avec l'équipe de recherche en droit privé de l'Université de Poitier, nous avons proposé une réflexion collective sur le sujet en présentant les enjeux juridiques du secteur, de ses acteurs, de l'activité *e-sportive*, de son financement et envisagent les perspectives qu'il offre (RABU, REVERCHON-BILLOT, MARTIN NEVOT et al. 2017).

8.5.2.3. Arbitrage dans l'e-sport

Toute compétition professionnelle, que le terrain de jeu soit en herbe, en terre battue, ou en images de synthèse, est soumise à des règles strictes.

Dès ses balbutiements, la triche devient indissociable du jeu en lui-même, un aspect du jeu à part entière, un élément de *gameplay* à part entière que certains joueurs se plaisent à explorer. Il y a un attrait des joueurs pour la transgression des règles. Les joueurs ne trichent pas simplement pour terminer un jeu plus facilement ou rapidement, ils trichent pour l'intérêt réjouissant que cela apporte intrinsèquement. Dans ce genre de situation, La triche n'est pas à proprement parlé un moyen mais une fin en soi. Les joueurs-tricheurs jouent à tricher, et partagent même parfois ensuite leurs exploits à l'instar de hauts faits de guerre. Devenue une véritable institution quand il s'agit de jeux solo, et même plus généralement hors ligne, la triche a pourtant pris une toute autre dimension avec l'arrivée des jeux multijoueur en ligne.

Malgré des débuts relativement confidentiels, le jeu vidéo compétitif devenu l'*e-sport* est rapidement pris au sérieux par toute une communauté, qui y voit une toute nouvelle manière de consommer le jeu vidéo. Mais qui dit compétition dit règlement. Notamment avec l'avènement d'une communication beaucoup plus large, la triche n'amuse plus autant les joueurs qu'avant, et s'impose désormais dans le domaine compétitif comme une problématique susceptible de gâcher leur expérience de jeu.

La pratique de l'*e-sport* donne plus que jamais une place centrale à la règle du jeu, là où le jeu vidéo consacre le programme informatique comme seul arbitre de ce qu'il est possible de faire ou non dans un univers virtuel. Cependant, avec l'*e-sport*, le joueur n'affronte plus seulement le logiciel, mais des adversaires humains dotés d'intentions, de projets, de créativité et de ruse. La machine devient alors moyen et lieu de l'affrontement, tandis que le contrôle de l'arbitrage assuré par le programme informatique n'est plus suffisant (ROUSTANT et BOUTET 2003). Il est nécessaire pour les acteurs de l'*e-sport* d'établir un système de règles, fournissant un cadre de référence identique pour tous les joueurs, et d'en assurer le respect afin d'éviter les phénomènes de triche. L'adoption de ce système de contraintes et de prescriptions précises empêche alors les comportements de joueurs contraires à « l'esprit du jeu », telle que l'exploitation d'une faille de la programmation (ou *exploiting*, ou tout simplement *exploit*) ou l'installation d'un programme additionnel permettant de faire ce qui est interdit aux autres dans le monde virtuel (*hacking*).

Les formes de triche Bien que ces deux cas soient découverts et traités différemment, les organismes de parrainage ainsi que les spectateurs peuvent se détourner d'une compétition ou même d'une ligue s'il s'avère qu'un logiciel a été utilisé pour remporter la victoire ou qu'une équipe a limité volontairement ses capacités pour une compensation monétaire.

Les différents types de triche peuvent être logiciels, sous la forme d'un programme, d'un script ou d'une macro logicielle, matériels, avec l'altération de la plateforme de jeu vidéo ou de ses périphériques, être caractérisé par une faute ou bien encore par du dopage.

Nous donnons dans la suite de ce paragraphe quelques applications succinctes de triche dans l'*e-sport*.

Wallhack Le *wallhack* consiste à changer la propriété des murs ou de leur équivalents d'une carte ou zone de jeu afin d'y voir à travers. La plupart des *wallhacks* rendent les murs au minimum semi-transparents ou donnent la possibilité aux joueurs de voir de l'autre côté des murs, comme si c'était un miroir sans tain, ce qui donne un avantage considérable sur leurs adversaires, en particulier pour les jeux présentant la brique de *gameplay* `Shooter (Sh)` (cf. paragraphe 4.4.2.4 p. 128). Certains *wallhacks* n'indiquent que le modèle des personnages correspondant, d'autres n'offrent qu'une semi-transparence, d'autres affichent les murs ou leurs équivalents en fil de fer.... Certains permettent aussi, dans certains cas pour le moins, de tirer ou encore de passer à travers des murs ou leurs équivalents.

Maphacking Le *maphacking* est un précédé permettant de s'affranchir du brouillard de guerre dans les jeux vidéo en bénéficiant, afin de permettre au joueur de pouvoir tirer parti d'une vision globale de la carte et des mouvements de troupes adverses.

Aimbot Un *aimbot* est un logiciel se substituant à un joueur pour viser à sa place. Un *aimbot* est conçu pour détecter et suivre automatiquement les mouvements des autres joueurs ou des cibles dans le jeu, permettant au joueur d'effectuer des tirs précis sans avoir à ajuster manuellement leur visée.

Faute Comme dans un contexte sportif plus traditionnel, ou plus général, une faute commise dans une compétition d'*e-sport* constitue une violation d'une règle officielle de la compétition, ou parfois du jeu vidéo lui-même. Outre les fautes résultants des autres pratiques de triche, telles que l'utilisation de logiciels ou de matériels inadaptés, le comportement antisportif ou déloyal, le retard de jeu ou toute autre action considérée comme contraire à l'éthique, au *fair-play* ou à l'intégrité du jeu forment également une faute.

Dopage Le dopage se réfère à l'administration, à l'encouragement ou à la facilitation de l'utilisation de substances ou de méthodes visant à augmenter artificiellement

les capacités physiques d'une personne ou d'un animal en prévision d'une compétition sportive. Son objectif est de donner un avantage injuste aux utilisateurs ou de dissimuler l'utilisation de ces substances ou méthodes lors des contrôles.

En 2015, la triche « vidéoludique » prend un nouveau tournant lorsque le joueur professionnel de Counter-Strike : Global Offensive Kory « Semphis » Friesen dévoile pendant un entretien que son ancienne équipe, ainsi que tous les autres joueurs de la ligue professionnelle d'ailleurs, étaient sous Adderall, une substance proche de l'amphétamine qui augmente la concentration, pendant l'Electronic Sports League (ESL) One Katowice organisé en Pologne. Jusqu'alors très peu encadré, le dopage chez les joueurs professionnels devient la cible de l'ESL. L'organisateur de tournois décide alors de prendre les choses en main en s'associant avec l'Agence mondiale antidopage, et établit pour la première fois une liste précise des produits interdits en compétition.

Dans l'*e-sport* on retrouve certaines formes de dopage classiques, similaires à celles présentes dans les sports traditionnels. Ces formes de dopage comprennent [9] :

— stimulants : Les stimulants, tels que la caféine hautement concentrée, les amphétamines et certains médicaments contre le trouble déficit de l'attention avec ou sans hyperactivité (TDAH, ou en anglais *attention deficit hyperactivity disorder*, ADHD), peuvent être utilisés pour améliorer la concentration, la vigilance et la réactivité ; ils sont employés pour rester éveillé et augmenter les performances cognitives ;
— anabolisants et hormones de croissance : les anabolisants, tels que les stéroïdes, et les hormones de croissance sont utilisés pour augmenter la masse musculaire, la force et l'endurance physique ; ils peuvent améliorer les performances dans les jeux nécessitant des compétences physiques ;
— bêta-bloquants : Les bêta-bloquants sont utilisés pour réduire l'anxiété, le stress et les tremblements ; ils peuvent améliorer la précision de la visée et la stabilité des mains pour des jeux vidéo qui requièrent une grande précision.

En plus de ces formes de dopage classiques, il existe des formes spécifiques de dopage liées à l'*e-sport*, qui se résument le plus souvent à l'amélioration des réflexes : certains joueurs peuvent utiliser des substances ou des techniques pour améliorer leurs réflexes, tels que des suppléments alimentaires en grande quantité ou des méthodes de stimulation électrique.

L'arbitre humain d'*e-sport* L'arbitre *e-sport* est la personne amenée à surveiller le bon déroulement des matchs compétitifs, suivant les règles établies par l'évènement en amont. Il doit faire preuve de médiation, doit aussi s'exprimer sur les éventuels litiges durant la confrontation, et doit évidemment vérifier qu'aucune triche n'est possible.

Par rapport aux sports traditionnels, où un arbitre peut avoir une vue plus large (et même souvent avoir accès à un système de rediffusion) que les seuls écrans d'un jeu

9. La liste complète des substances interdites en sport traditionnel qui tendent à l'être de plus en plus souvent dans les compétitions d'*e-sport* est consultable ici : https://www.wada-ama.org/en/prohibited-list.

vidéo, il y a plus de risque de passer à côté d'un comportement inadapté. Ce risque d'erreur « humaine » doit être limité au maximum, et cela constitue un véritable enjeu qui peut être relevé grâce à une assistance logicielle.

Des contrôles anti-dopages peuvent être effectués sur les joueurs professionnels lors de certaines compétitions, en prélevant des échantillons biologiques pour détecter la présence de substances interdites.

Une formation adéquate des arbitres et des équipes de sécurité est essentielle pour reconnaître les signes de triche, comprendre les techniques de détection et appliquer les règles de manière équitable et cohérente.

Une fois la prise de décision fermement établie, les sanctions applicables sont diverses : elles peuvent aller d'un bannissement du jeu à titre provisoire, d'un bannissement illimité ou, dans des cas extrêmes, de poursuites judiciaires initiées par les éditeurs du jeu (en les basant notamment sur la violation de *copyright*).

8.5.2.4. Détection automatique et aide à l'e-arbitrage

Comme nous l'avons vu, la triche risque de compromettre gravement le potentiel et la nature accueillante de l'*e-sport*. Le problème est qu'une triche peut facilement passer inaperçue en raison de la nature même de l'acte : elle est rarement assez évidente pour être remarquée immédiatement sans une analyse entrainée et méticuleuse du match, ou à l'aide d'un logiciel spécifique.

Les organisateurs d'*e-sport* peuvent travailler en étroite collaboration avec les concepteurs et développeurs de jeux, d'autant plus facilement lorsqu'il s'agit des mêmes entités, pour améliorer la sécurité et la détection de la triche, en partageant des informations sur les nouvelles formes de triche et en mettant à jour les systèmes de détection.

Nous présentons maintenant les principaux paradigmes de détections automatiques d'aide à la lutte contre la triche et à l'*e-arbitrage*. Cependant, il nous semble important d'avoir en tête que les méthodes de lutte contre la triche et de détection peuvent évoluer au fil du temps à mesure que de nouvelles formes de triche émergent et que les technologies progressent.

Détection par le client de jeu Il existe un certain nombre de logiciels anti-triches et d'autres contre-mesures tangibles pour empêcher les joueurs d'*e-sport* de tricher, et il revient généralement aux concepteurs et aux développeurs de jeux eux-mêmes de s'assurer que leur jeu vidéo est à l'abri des piratages tiers ou personnalisés et d'autres pratiques du même genre. Tous les jeux vidéo n'utilisent pas de méthode de logiciel anti-triche côté client.

Parmi les principaux programmes anti-triches côté client connus à ce jour, nous pouvons citer Kaspersky Anti-Cheat, Valve Anti-Cheat (VAC), ESEA, Easy Anti-Cheat, HackShield, Warden ou Punkbuster. Ils permettent de passer au crible les logiciels en usage lors d'une partie pour repérer les triches connues en analysant les comportements suspects ou les schémas de jeu anormaux.

Comme toute autre technologie, ces programmes sont sujets à des erreurs. Les comptes peuvent être volés ou piratés et, bien entendu, la technologie ne permet pas de savoir qui joue réellement sur le compte. Il est probable que des joueurs aient connus des injustices dues à ce genre de dispositif. Dans le cas de joueurs professionnels, cela peut avoir des conséquences catastrophique et compromettre des carrières entières.

La plupart du temps, lorsque la triche a lieu lors d'événements d'*e-sport*, c'est parce que le joueur a accès à des périphériques et/ou à Internet sur l'ordinateur d'un adversaire avant le début de l'événement. Cela leur permet de préparer tout logiciel de triche et/ou de piratage. Cette pratique est extrêmement difficile à repérer. Le plus souvent, la plupart des logiciels anti-triche ne peuvent pas repérer ces tricheurs en flagrant délit, ce qui signifie qu'une sécurité et une gestion couteuse plus strictes sur le site sont nécessaires pour les éradiquer.

Détection par le serveur de jeu Pour les jeux sur des serveurs dédiés, ou les jeux en réseau pair-à-pair, il existe des programmes d'analyse des données et de *machine learning* compilant un ensemble de statistiques sur les joueurs disposant d'un système d'arbitrage permettant de valider les données et ainsi de s'assurer que les tricheurs n'envoient pas de fausses informations aux serveurs.

Un autre type d'attaque du serveur de jeu vidéo est possible, celle de déni de service (DDoS pour *distributed denial of service attack*), qui consiste à saturer la bande-passante de la cible, la privant ainsi de connexion.

Détection par les autres joueurs Dans les tournois ou parties en local (aussi appelées LAN en référence à l'architecture de réseau local utilisé pour ce genre de partie, de l'anglais *local area network*), il est parfois très simple de détecter un tricheur, en commençant par regarder tout simplement son écran de manière directe ou indirecte.

Les jeux vidéo offrant un mode « spectateur » ou de rediffusion de parties peut grandement faciliter la détection de soupçon de triche, et permet son évaluation par d'autres dispositifs.

Certains jeux vidéo proposent également une option de signalement, mais pouvant être utilisée de manière abusive par les autres joueurs, ou envahissante, les signalements sont rarement pris en compte, en tout cas lors d'un traitement manuel.

Les joueurs peuvent également parier sur leur propre défaite, dans le but de gagner de l'argent, des gains extrinsèques ou des gains intrinsèques au jeu vidéo, et ainsi fausser sciemment une partie. Cette attitude, tout aussi répréhensible que les autres, peut être très difficilement détecter par d'autre dispositif qu'un moyen humain massif.

Détection par le matériel Certaines entreprises technologiques travaillent à la création de dispositifs anti-triche qui sont attachés aux claviers et aux souris. Ces dispositifs suivent essentiellement les mouvements du clavier et de la souris et sont capables de détecter la manière dont ces périphériques interagissent avec le jeu vidéo.

Détection par les données de jeu Les données de jeu, telles que les statistiques, les mouvements des joueurs, les actions effectuées, peuvent être analysées pour détecter des modèles inhabituels ou des comportements incohérents, indiquant potentiellement une triche. Des systèmes de surveillance en temps réel d'analyse de données peuvent même être utilisées pour observer les comportements des joueurs pendant les parties et détecter toute activité suspecte ou non conforme. L'intelligence artificielle peut être utilisée pour assister les arbitres dans la prise de décisions en analysant les actions de jeu, en identifiant les infractions potentielles et en fournissant des recommandations basées sur les règles du jeu.

C'est pour ce type de détection que notre solution TAGED appliqué à l'*e-sport* peut prendre tout son sens.

8.5.2.5. Méthodes de classement

Classement Elo Le classement Elo est un système de calcul développé par Arpad Elo dans les années 1960 (ELO 2008). Il est utilisé pour évaluer la force relative des joueurs dans des jeux compétitifs équilibrés (ou à somme nulle), tels que les échecs, les jeux de société et les jeux vidéos en ligne.

Le système de classement Elo repose sur le principe que la performance d'un joueur peut être estimée en comparant ses résultats aux résultats d'autres joueurs. Chaque joueur se voit attribuer un score Elo qui représente sa virtuosité (cf. sous-sous-section 4.5.4.4 p. 135) ou son niveau de compétence. Plus le score Elo d'un joueur est élevé, plus il est considéré comme doué. La performance se conforme à une distribution de probabilité en forme de courbe en cloche.

Quand des joueurs s'affrontent, le résultat de la partie est utilisée pour mettre à jour les scores Elo des joueurs. Si un joueur avec un score Elo élevé l'emporte sur un joueur avec un score Elo inférieur, son score augmente moins que s'il avait gagné une partie avec un joueur de niveau similaire ou supérieur. À l'inverse, si un joueur avec un score Elo plus faible bat un joueur avec un score Elo supérieur, son score évoluera plus conséquemment.

Le système de classement Elo est conçu pour être auto-ajustable et évolutif. Au fil du temps, les scores Elo des joueurs peuvent augmenter ou diminuer en fonction de leurs performances. Cela permet de maintenir un classement dynamique et de mettre en relation des joueurs de niveaux de compétences similaires pour des affrontements équilibrés.

Le classement Elo a été largement adopté dans de nombreux sports traditionnels, jeux et jeux vidéo compétitifs, et il est considéré comme une mesure relativement fiable de la virtuosité d'un joueur.

Théorie du classement Elo L'évaluation de la virtuosité, ou force, d'un joueur se mesure par un classement en points distribués selon une loi normale de répartition. Le classement est basé sur la notion de force relative de deux joueurs ou de deux

équipes. Soit p la probabilité d'un joueur J_1 contre un joueur J_2, nous avons :

$$q = P(J_1 \mid J_2)$$

Le rapport entre la probabilité de gagner du joueur J_1 et la probabilité qu'il perde (ou que son adversaire J_2 l'emporte) exprime la force relative de J_1 contre J_2 de la manière suivante :

$$f_{A/B}(q) = \frac{q}{1-q}$$

La valeur de $f_{A/B}(q)$ représente donc combien, J_1, a de fois plus de chances de gagner face à J_2.

Soit :

— $q = P(J_1 \mid J_2)$, la probabilité de gain de J_1 contre J_2 ;
— $r = P(J_2 \mid J_3)$, la probabilité de gain de J_2 contre J_3 ;
— $p = P(J_1 \mid J_3)$, la probabilité de gain de J_1 contre J_3.

Alors, nous avons :

$$f(p) = f(q) \times f(r)$$

Et la probabilité de gain de J_1 contre J_3 est alors définie de la manière suivante :

$$p = \frac{f(p)}{1+f(p)}$$

Une même différence de score entre deux joueurs implique la même espérance de gain, c'est ce que garantit la force relative. Ainsi, par exemple, un joueur ayant un score Elo de 2750 a autant de chance de l'emporter sur un joueur en ayant un de 2700 qu'un joueur classé 1250 Elo a de chance de gagner une partie avec un joueur classé 1200 Elo.

Afin d'obtenir un classement Elo, il ne nous est pas possible de l'effectuer avec le produits des forces (formule 8.5.2.5), nous recherchons alors une fonction $D(p)$ telle que la différence de score entre les joueurs J_1 et J_3 soit définie de la façon suivante :

$$D(p) = D(q) + D(r)$$

La transformation obtenue d'un produit en somme est la définition d'une fonction logarithme. La formule de Elo, obtenue en introduisant le facteur multiplicatif 400, est la suivante :

$$D(p) = 400 \log f(p)$$

La fonction réciproque $p(D)$ donne la probabilité de gain en fonction de la différence de score D de la manière suivante :

$$p(D) = \frac{1}{1+10^{\frac{-D}{400}}}$$

Limites du classement Elo Les vulnérabilités et les limites du système de classement Elo sont liées à des appariements sélectifs et à des populations non repré-

sentatives, qui rendent le modèle inexact. Par exemple, compte tenu de la forme de la courbe de distribution logistique, un joueur mieux classé qui a la possibilité de choisir les parties qu'il accepte peut rejeter les parties contre des adversaires dont la probabilité de perte est élevée et jouer contre des joueurs moins bien classés dont la performance attendue est beaucoup plus faible. De même, isolément, une sélection de joueurs qui n'est pas représentative de la population globale des joueurs générera des évaluations qui ne sont pas représentatives de leur performance attendue par rapport aux joueurs de la population globale.

Eloboost Le *eloboost*, ou *boosting*, est une pratique dans les jeux vidéo en ligne compétitifs, principalement dans les jeux vidéo de type MOBA [10] tels que League of Legends, Dota 2, ou Overwatch.

Le *eloboost* consiste à faire appel à un mandataire, un joueur sensément hautement compétent, fréquemment appelé « *booster* », pour jouer au nom d'un autre joueur dans le but d'augmenter son classement Elo dans un jeu vidéo. Le joueur qui souhaite recevoir ce genre de « service » fournit généralement ses identifiants de connexion au « *booster* », qui joue grâce à cela des parties compétitives à sa place. Il existe aussi des systèmes automatiques d'*eloboost* indiquant les personnages à choisir de manière préférentielle pour la pratique afin que son incontestabilité soit la plus maximale possible.

L'objectif du *eloboost* est d'atteindre un classement Elo plus élevé que celui que le joueur aurait pu atteindre par ses propres moyens. Cela peut être motivé par le désir d'avoir un meilleur classement personnel, de déverrouiller des récompenses spécifiques associées à un classement élevé, de jouer avec des coéquipiers plus compétents, ou simplement de se vanter auprès d'une communauté.

Cependant, il est important de noter que le *eloboost* est généralement considéré comme une violation des règles de nombreux jeux vidéo en ligne. Les concepteurs de jeux vidéo considèrent le plus souvent cette pratique comme déloyale, car elle compromet l'intégrité du système de classement Elo en permettant à un joueur d'atteindre un niveau qu'il n'aurait pas pu obtenir par ses propres compétences. En conséquence, les joueurs qui sont surpris en train de s'adonner à cette pratique peuvent se voir affliger des sanctions, allant d'avertissements temporaires à des suspensions permanentes de leur compte de joueur.

Les méthodes de classement, autres que Elo, ont également héritées, par abus de langage, du vocable *eloboost*.

Match making rating (MMR) Le *match making* (ou *matchmaking*) *rating* (MMR) est un système de score caché (bien qu'il existe des sites Web présentant des estimations) utilisé dans de nombreux jeux en ligne pour classer les joueurs en fonction de leur niveau de virtuosité (cf. sous-sous-section 4.5.4.4 p. 135). Il vise à créer des parties équilibrées en regroupant des joueurs qui ont des niveaux de compétences considérés

10. Multiplayer online battle arena.

comme similaires, et répond en cela à une problématique fondamentale de tout jeu vidéo en ligne : la gestion d'affrontement, ou *matchmaking*.

Le MMR est généralement calculé à partir des résultats des parties précédentes. Lorsque un joueur gagne une partie contre des joueurs ayant un MMR plus élevé, son MMR augmente davantage. De manière similaire, si le joueur perd contre des joueurs ayant un MMR inférieur, son MMR baisse. Ce genre de système de calcul complexe prend, en principe, également en compte d'autres facteurs tels que la durée de la partie, la marge de victoire ou de défaite, et la « qualité » des adversaires. Pour un joueur de jeux en ligne, le MMR peut s'interpréter comme la difficulté de la partie. Avec cette approche, il est habituellement rare pour un joueur de rencontrer des adversaires ayant un niveau théorique bien supérieur au sien.

Le MMR est généralement représenté par un nombre ou une échelle, qui varie en fonction du jeu vidéo. Par exemple, dans le jeu Dota 2, le MMR est mesuré sur une échelle allant de 0 à 10000, tandis que dans Overwatch, il est mesuré sur une échelle de 1 à 5000. Dans League of Legends, et de manière générale dans les jeux vidéo de type MOBA, le système de classement est discrétisé par paliers.

FIGURE 8.11. – Système de classement de League of Legends [a]

a. Représentations des paliers, avec, de gauche à droite : Fer, Bronze, Argent, Or, Platine, Diamant, Maître, Grand-Maître et *Challenger* ; illustration par Riot Games.

L'objectif principal du MMR est d'améliorer l'expérience de jeu en créant des rencontres compétitives où les joueurs ont des chances égales de gagner. Cela permet d'éviter les déséquilibres et les parties où un joueur est nettement supérieur ou inférieur aux autres, ce qui peut être frustrant.

Le MMR est particulièrement sensible au *eloboost*, qui est surtout fréquent à de bons niveaux estimés de joueurs (mais pas aux meilleurs niveaux). C'est un des principaux fléaux de la méthode MMR, bien que, cependant, il n'en existe également un autre d'importance équivalente, et qui nuit grandement à l'expérience de jeu : les problèmes de file d'attente dynamique (ou *dynamic queue*, ou encore *dynamiQ*). Ce problème intervient dans les jeux vidéo en équipe. Dans cette situation, le MMR d'une équipe est calculé en fonction des MMR des joueurs qui la composent. Ce qui ne manque pas, bien souvent, de fausser inéluctablement le *matchmaking*. Nous atteignons avec les systèmes de files d'attente dynamique une limite claire du MMR dans la mesure où il n'est pas réellement pertinent d'utiliser un classement personnel dans un jeu vidéo basé sur le travail d'équipe.

8.5.2.6. L'e-sport comme outil éducatif

Jouer à des jeux vidéo, y compris lorsqu'ils peuvent être considérés à tord ou à raison comme violent, en compétition ou en coopérant avec d'autres personnes peut avoir des effets bénéfiques à grande échelle en augmentant la réactivité empathique et la coopération entre les joueurs et en faisant apprendre aux joueurs que les comportements socialement louables sont appréciables et communs. Les mécanismes de jeu, tels que les récompenses et les incitations à la coopération, peuvent jouer un rôle clef dans la promotion de comportement prosociaux dans les jeux vidéo (VELEZ, GREITEMEYER, WHITAKER et al. 2014).

La littérature explore la mise en place d'une ligue d'*e-sport* à des fins éducatives et met en évidence l'importance de l'*e-sport* en tant qu'outil pédagogique et souligne les nombreux avantages qu'il peut offrir aux étudiants, tels que le développement des compétences en collaboration, en résolution de problèmes, en communication et en pensée critique. En s'inspirant sur les ligues d'*e-sport* déjà en place et sur les études de cas afférentes, il est conseillé, afin d'intégrer l'*e-sport* au milieu éducatif, d'apporter un soin tout particulier à la structuration de la ligue avec des règles et des directives claires, la sélection et la gestion des jeux appropriés pour l'apprentissage, la création d'un environnement inclusif et respectueux, la mise en place de programmes de formation pour les enseignants et les entraîneurs, et l'utilisation d'outils de suivi et d'évaluation afin de mesurer les progrès des étudiants. De plus, la collaboration entre les établissements scolaires, les enseignants, les étudiants, les parents et les partenaires de l'industrie du jeu vidéo augmente notablement les chances de succès de la ligue *e-sportive* (CHO, TSAASAN et STEINKUEHLER 2019).

8.5.2.7. L'e-sport comme domaine de recherche

Le phénomène de l'*e-sport* implique une compréhension de l'engagement compétitif dans les jeux vidéo. Bien qu'il soit très populaire et bien connu des joueurs eux-même, la recherche scientifique et les études sur les jeux vidéo n'en est encore qu'à ses débuts concernant cette culture particulière du jeu. Les défis et opportunités de recherche dans ce domaine concernent notamment la compréhension des motivations des joueurs (cf. sous-section 4.5.6 p. 136), l'analyse des stratégies de jeu, l'impact psychologique et physique des compétitions, ainsi que les aspects socioculturels, économiques, et juridiques de l'*e-sport*. Il est important de développer un cadre théorique solide pour étudier l'*e-sport* dont les domaines potentiels majeurs sont la théorie des jeux (cf. sous-section 4.6.2 p. 139), les études culturelles, la psychologie du sport, l'économie du sport, le droit du sport, les études médiatiques, les statistiques et la fouille de données. Faire l'exploration approfondie de ces domaines permettrait de mieux comprendre l'*e-sport* en tant que phénomène culturel, social et économique (ADAMUS 2012).

Comme l'industrie du sport elle-même adopte l'*e-sport* en tant que sport, les chercheurs devraient considérer l'*e-sport* comme une manifestation de la « sportification » et examiner ses effets négatifs et positifs (HEERE 2017).

À mesure que l'*e-sport* gagne en popularité et, dans certains cas même, dépassent l'audience des sports traditionnels, ils créent de bonnes opportunités d'étudier les joueurs et les systèmes à grand échelle (REITMAN, ANDERSON-COTO, WU et al. 2019).

8.6. Conclusion

Les points forts de TAGED sont l'extraction et la fouille des déséquilibres, l'analyse de la mécanique interne du jeu vidéo et l'universalité de la méthode, au service du concepteur de jeu vidéo.

La méthode, et ses implémentations actuelles, offrent la possibilité d'extraire, de valider au besoin, et de quantifier les déséquilibres dans les mécaniques, les dynamiques et les esthétiques d'un jeu vidéo à n'importe quel moment durant la durée de vie du dispositif ludique. En tant que procédé automatisé, les connaissances rendues, représentées pour TAGED en tant que données du Skycube émergent ordonné et rendus intermédiaires optionnels, sont de nature bien plus complexe qu'avec un traitement manuel ou l'expression contextuelle de l'expérimentation des joueurs. Cette solution est flexible dans son utilisation, elle propose une configuration personnalisable de chaque exécution, admet l'usage de jeux de données réels ou sur mesure, permet des itérations d'essais rapides et accepte aussi bien les tests théoriques qu'empiriques de la conception de jeux comme la conception de niveaux.

Nous considérons que la principale faiblesse de notre proposition, et ce bien qu'elle présente une importante adaptabilité, est qu'il s'agit d'une solution spécifique qui demande un effort d'adaptation et d'intégration pour chaque utilisation, bien que les moteurs disposant de briques de *gameplay* similaires semblent présenter un haut niveau de réutilisabilité, et que les dispositifs ludiques pleinement accessibles facilitent aussi grandement l'intégration.

9. Conclusion

Sommaire

9.1. Bilan des contributions 261
9.2. Perspectives 262

9.1. Bilan des contributions

Dans cette thèse, nous avons proposé des solutions de fouille de données destinées à assister le décideur, ou concepteur de jeu vidéo en l'occurrence, à l'amélioration de l'équilibrage, ou à sa maitrise, dans la conception des jeux vidéo. En analysant de gros volumes de données, dont dispose maintenant largement l'industrie des jeux vidéo, l'objectif est d'extraire des connaissances et des recommandations de conception, ou *game design*, en fonction de la classification (ALVAREZ 2018). Après avoir construit notre état de l'art de manière à ce qu'il apporte les connaissances nécessaires à l'élaboration de nos solutions, nous avons proposé des modèles pour répondre aux besoins d'équilibrage des jeux vidéo en satisfaisant aux exigences d'intégration :

— de l'ensemble des résultats possibles, pour une navigation favorisée, en introduisant les dimensions hiérarchiques (HURTADO et A. O. MENDELZON 2002) aux réponses apportées ; afin d'en apporter une solution technique, nous avons offert un cadre et une définition théorique formelle accompagnés de la définition du cube de données hiérarchique et de sa représentation sans perte la plus concise possible, particulièrement pertinente dans un contexte de données fortement corrélées typique du domaine des jeux vidéo, le cube de données hiérarchique fermé ;
— de données optimales selon plusieurs critères et préférences, gagnants en importance de manière longitudinale, comme les stratégies de jeu par rapport au niveau estimé d'un joueur ; pour cela nous avons introduit le concept de Skycube émergent combinant celui du Skycube (PEI, YUAN, LIN et al. 2006) et celui du cube de données émergent (NEDJAR, CASALI, CICCHETTI et al. 2007) et nous avons proposé également une représentation doublement réduite, le Skycube émergent fermé, ainsi que les moyens pour les calculer ;
— de données optimales selon plusieurs critères et préférences, ordonnées entre elles par un score ; à cette fin, nous avons proposé une amélioration de la principale méthode actuelle, dp-idp, ne permettant pourtant pas toujours de distinguer les données deux à deux, et nous avons alors proposé la nôtre, Cosky, qui appliqué conjointement au principe de Skyline multiniveaux (PREISINGER et

ENDRES 2015), est une approche de type TOPSIS qui permet toujours un classement efficace, et nous avons introduit l'algorithme DeepSky correspondant ;

— de données optimales selon plusieurs critères et préférences, gagnants en importance de manière longitudinale, et ordonnées entre elles, autrement dit des données issues d'un Skycube émergent ordonné, soit, à l'exception des données ayant des dimensions hiérarchiques, la combinaison de l'ensemble des méthodes présentées ; nous avons appelé la solution permettant cette extraction de données TAGED, pour *tool assisted game equilibrium design*, et avons précisé qu'elle pouvait s'utiliser conjointement avec des données brutes d'entrée, réelles ou modélisées, intrinsèques au jeu vidéo, comme celles issues des mécaniques, dynamiques ou esthétiques de jeu (SALEN et ZIMMERMAN 2004), ou bien extrinsèques comme celles extraites des *verbatims* des comportements de joueurs, déviants ou atypiques, et des informations de parties, solo ou multijoueur ; après avoir appliqué nos méthodes, et la solution TAGED, à l'équilibrage, empirique ou théorique, du *game design*, nous avons ouvert la perspective, qui nous semble pertinente, de le porter à l'*e-sport*, mais aussi de considérer l'*e-sport* comme un outil éducatif et un domaine de recherche.

9.2. Perspectives

Le calcul d'un cube de données, quel qu'il soit, n'est pas une fin en soi car son cycle de vie en s'arrête pas là (MORFONIOS et IOANNIDIS 2008). De la sorte, nous pouvons étudier l'efficacité des réponses aux requêtes ainsi que la maintenance incrémentale. Donc, observer ces représentations de manière longitudinale nous semble être une perspective remarquable de travail de recherche.

Envisager une méthode améliorative ou alternative de classement de Skyline en utilisant l'approche Page-Rank (SOLANKI, VERMA et CHAHAR 2022) nous semble judicieux.

Il nous semble pertinent de mettre au point une méthode automatisée permettant de sélectionner les ensembles de critères les plus critiques pour l'analyse des déséquilibres.

De l'avis des auteurs eux-mêmes, le modèle structurel Orion serait grandement amélioré par la modélisation de techniques de fouille de données, qui fournirait des informations supplémentaires de manière à rechercher automatiquement des valeurs optimales au cours d'un processus de classification (SOLER 2015). Ils ajoutent d'ailleurs que l'utilisation combinée de ces solutions constituerait un outil plus pédagogique. Ce travail pourrait tout à fait être effectué par TAGED, et notre solution pourrait alors être employée à la place ou de concert avec Orion.

Nous pensons que la solution TAGED pourrait être davantage performante avec l'intégration d'un outil de détection de *bots* (DOUDECH 2016). En effet, la présence de *bots* dans les jeux vidéo multijoueur représentent des défis à relever, tels que la triche, la perturbation de l'équilibre du jeu vidéo et la dégradation de l'expérience de jeu, afin de maintenir l'intégrité et l'équité des dispositifs ludiques. Plusieurs tech-

niques basées sur la fouille de données de jeux telles que l'analyse de comportements, l'apprentissage automatique et l'extraction de caractéristiques, permettent d'identifier les schémas réflexifs et les signes distinctifs des *bots*, et de les différencier des comportements des joueurs.

TAGED pourrait aussi bénéficier de la création de plusieurs profils de *bots* apprenants pour évaluer l'impact de la connaissance des déséquilibres pour maximiser ses gains.

Enfin, dans le cadre de notre recherche, nous assumons d'appliquer notre travail, ainsi que notre méthode TAGED, au domaine exclusif des jeux vidéo. Pour autant, une telle solution pourrait avoir des utilisations possibles dans d'autres domaines, comme celui de l'industrie, de la logistique, du transport…

Bibliographie

[Abt70] Clark ABT. *Serious Games*. Viking Press, 1970 (cf. p. 123, 140).

[Ada09] Ernest ADAMS. *Fundamentals of Game Design*. 2e éd. New Riders, 2009 (cf. p. 135).

[Ada12] Tanja ADAMUS. « Playing Computer Games as Electronic Sport : In Search of a Theoretical Framework for a New Research Field ». In : *Computer Games and New Media Cultures : A Handbook of Digital Games Studies*. Sous la dir. de Johannes FROMME et Alexander UNGER. Dordrecht : Springer Netherlands, 2012, p. 477-490. ISBN : 978-94-007-2777-9. DOI : 10.1007/978-94-007-2777-9_30. URL : https://doi.org/10.1007/978-94-007-2777-9_30 (cf. p. 259).

[Aga+96] Sameet AGARWAL, Rakesh AGRAWAL, Prasad DESHPANDE et al. « On the Computation of Multidimensional Aggregates ». In : *VLDB*. Sous la dir. de T. M. VIJAYARAMAN, Alejandro P. BUCHMANN, C. MOHAN et al. Morgan Kaufmann, 1996, p. 506-521. ISBN : 1-55860-382-4 (cf. p. 17, 29, 44).

[AIS93] Rakesh AGRAWAL, Tomasz IMIELINSKI et Arun N. SWAMI. « Mining Association Rules between Sets of Items in Large Databases ». In : *SIGMOD Conference*. Sous la dir. de Peter BUNEMAN et Sushil JAJODIA. ACM Press, 1993, p. 207-216 (cf. p. 46).

[Agr+96] Rakesh AGRAWAL, Heikki MANNILA, Ramakrishnan SRIKANT et al. « Fast Discovery of Association Rules ». In : *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996, p. 307-328. ISBN : 0-262-56097-6 (cf. p. 18, 55, 60).

[Ala+20] Karim ALAMI, Nicolas HANUSSE, Patrick Kamnang WANKO et al. « The negative skycube ». In : *Inf. Syst.* 88 (2020). DOI : 10.1016/j.is.2019.101443. URL : https://doi.org/10.1016/j.is.2019.101443 (cf. p. 84).

[Alb15] Marc ALBINET. *Concevoir un jeu vidéo : les méthodes et les outils des professionnels expliqués à tous !* 3e éd. Limoges : Fyp éditions, 2015 (cf. p. 20, 122, 130, 135, 136, 140).

[Alv07] Julian ALVAREZ. « DU JEU VIDÉO AU SERIOUS GAME Approches culturelle, pragmatique et formelle ». Thèse de doct. Déc. 2007. DOI : 10.13140/RG.2.1.2527.1767 (cf. p. 20, 122, 129, 130).

[Alv18] Julian ALVAREZ. *Approche atomique du jeu vidéo : Briques Gameplay 3.0*. Déc. 2018 (cf. p. 20, 22, 128, 140, 261).

[And+05] Gustavo ANDRADE, Geber RAMALHO, Hugo SANTANA et al. « Extending Reinforcement Learning to Provide Dynamic Game Balancing ». In : jan. 2005, p. 7-12 (cf. p. 135).

[BS66] O. BARNDORFF-NIELSEN et M. SOBEL. « On the Distribution of the Number of Admissible Points in a Vector Random Sample ». In : *Theory of Probability and its Applications* 11 (1966), p. 249 (cf. p. 204).

[Bar96] Richard BARTLE. « Hearts, clubs, diamonds, spades : Players who suit MUDs ». In : (juin 1996) (cf. p. 20, 125).

[Bar+07] I. BARTOLINI, P. CIACCIA, V. ORIA et al. « Flexible integration of multimedia sub-queries with qualitative preferences ». Anglais. In : *journal of Multimed Tools Appl* 33 (2007), p. 275-300 (cf. p. 219).

[Bec+90] Norbert BECKMANN, Hans-Peter KRIEGEL, Ralf SCHNEIDER et al. « The R*-Tree : An Efficient and Robust Access Method for Points and Rectangles ». In : *SIGMOD Conference*. Sous la dir. d'Hector GARCIA-MOLINA et H. V. JAGADISH. ACM Press, 1990, p. 322-331 (cf. p. 81).

[Bee+84] Catriel BEERI, Martin DOWD, Ronald FAGIN et al. « On the Structure of Armstrong Relations for Functional Dependencies ». In : *J. ACM* 31.1 (1984), p. 30-46 (cf. p. 190).

[Beh+12] Majid BEHZADIAN, S. KHANMOHAMMADI OTAGHSARA, Morteza YAZDANI et al. « A state-of the-art survey of TOPSIS applications ». In : *Expert Systems with Applications* 39.17 (2012), p. 13051-13069. ISSN : 0957-4174. DOI : https://doi.org/10.1016/j.eswa.2012.05.056. URL : https://www.sciencedirect.com/science/article/pii/S0957417412007725 (cf. p. 22, 222).

[Ben+78] Jon Louis BENTLEY, H. T. KUNG, Mario SCHKOLNICK et al. « On the Average Number of Maxima in a Set of Vectors and Applications ». In : *J. ACM* 25.4 (1978), p. 536-543 (cf. p. 74, 204).

[Ber+97] Stefan BERCHTOLD, Christian BÖHM, Daniel A. KEIM et al. « A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space ». In : *PODS*. Sous la dir. d'Alberto MENDELZON et Z. Meral ÖZSOYOGLU. Chairman-Mendelzon, Alberto and Chairman-Özsoyoglu, Z. Meral. Tucson, Arizona, United States : ACM, 1997, p. 78-86. ISBN : 0-89791-910-6 (cf. p. 80).

[BKK96] Stefan BERCHTOLD, Daniel A. KEIM et Hans-Peter KRIEGEL. « The X-tree : An Index Structure for High-Dimensional Data ». In : *VLDB*. Sous la dir. de T. M. VIJAYARAMAN, Alejandro P. BUCHMANN, C. MOHAN et al. Morgan Kaufmann, 1996, p. 28-39. ISBN : 1-55860-382-4 (cf. p. 81).

[Ber89] Claude BERGE. *Hypergraphs : combinatorics of finite sets*. Amsterdam : North-Holland, 1989 (cf. p. 56).

[BB09] Karl BERGSTRÖM et Staffan BJÖRK. « Exploring Aesthetic Ideals of Gameplay ». In : (jan. 2009) (cf. p. 124).

[Bes16] Nicolas BESOMBES. « Les jeux vidéo compétitifs au prisme des jeux sportifs : du sport au sport électronique ». In : *Sciences du jeu* 5 (fév. 2016). DOI : 10.4000/sdj.612 (cf. p. 246).

[BR99] Kevin S. BEYER et Raghu RAMAKRISHNAN. « Bottom-Up Computation of Sparse and Iceberg CUBEs ». In : *SIGMOD Conference*. Sous la dir. d'Alex DELIS, Christos FALOUTSOS et Shahram GHANDEHARIZADEH. ACM Press, 1999, p. 359-370. ISBN : 1-58113-084-8 (cf. p. 17, 18, 29, 42, 46, 48, 55, 68, 72, 87, 99, 173).

[Bir70] Garrett BIRKHOFF. *Lattice Theory*. Third (new). T. XXV. AMS Colloquium Publications. American Mathematical Society, 1970 (cf. p. 72, 189).

[BCA15] Kenneth S. BØGH, Sean CHESTER et Ira ASSENT. « Work-Efficient Parallel Skyline Computation for the GPU ». In : *Proc. VLDB Endow.* 8.9 (2015), p. 962-973. DOI : 10.14778/2777598.2777605. URL : http://www.vldb.org/pvldb/vol8/p962-boegh.pdf (cf. p. 204).

[BL04] Francesco BONCHI et Claudio LUCCHESE. « On Closed Constrained Frequent Pattern Mining ». In : *ICDM*. Sous la dir. de Katharina MORIK et Rajeev RASTOGI. IEEE Computer Society, 2004, p. 35-42. ISBN : 0-7695-2142-8 (cf. p. 86).

[Bor21] Émile BOREL. « La théorie du jeu les équations intégrales à noyau symétrique ». In : *Comptes Rendus de l'Académie des Sciences* 173 (jan. 1921), p. 1304-1308 (cf. p. 139, 141).

[BKS01] Stephan BÖRZSÖNYI, Donald KOSSMANN et Konrad STOCKER. « The Skyline Operator ». In : *ICDE*. IEEE Computer Society, 2001, p. 421-430. ISBN : 0-7695-1001-9 (cf. p. 18, 74, 77, 78, 80, 86, 204).

[BCQ14] Tassadit BOUADI, Marie-Odile CORDIER et Rene QUINIOU. « Requêtes Skyline Hiérarchiques ». In : *Actes des 10e journées francophones sur les Entrepôts de Données et l'Analyse en Ligne, EDA 2014, Vichy, France, 5-6 Juin, 2014*. Sous la dir. de Sandro BIMONTE, Laurent D'ORAZIO et Elsa NEGRE. T. B-10. RNTI. Hermann-Éditions, 2014, p. 51-66. URL : http://editions-rnti.fr/?inprocid=1002014 (cf. p. 83).

[Bro05] Gilles BROUGÈRE. *Jouer/Apprendre*. Jan. 2005 (cf. p. 123, 140).

[Bur+05] Douglas BURDICK, Manuel CALIMLIM, Jason FLANNICK et al. « MAFIA : A Maximal Frequent Itemset Algorithm ». In : *IEEE Trans. Knowl. Data Eng.* 17.11 (2005), p. 1490-1504 (cf. p. 87).

[Cai58] R Les Jeux Et Les Hommes ; Le Masque Et Le CAILLOIS. « Les Jeux Et Les Hommes ; Le Masque Et Le Virtige ». In : (1958) (cf. p. 20, 122, 123, 127, 135, 140).

[CCL03a] Alain CASALI, Rosine CICCHETTI et Lotfi LAKHAL. « Cube Lattices : A Framework for Multidimensional Data Mining ». In : *SDM*. Sous la dir. de Daniel BARBARÁ et Chandrika KAMATH. SIAM, 2003. ISBN : 0-89871-545-8 (cf. p. 18, 57, 61, 68, 87).

[CCL03b] Alain CASALI, Rosine CICCHETTI et Lotfi LAKHAL. « Extracting semantics from data cubes using cube transversals and closures ». In : *KDD*. Sous la dir. de Lise GETOOR, Ted E. SENATOR, Pedro DOMINGOS et al. ACM, 2003, p. 69-78. ISBN : 1-58113-737-0 (cf. p. 19, 56).

[Cas+07] Alain CASALI, Sébastien NEDJAR, Rosine CICCHETTI et Lotfi LAKHAL. « Convex Cube : Towards a Unified Structure for Multidimensional Databases ». In : *DEXA*. Sous la dir. de Roland WAGNER, Norman REVELL et Günther PERNUL. T. 4653. Lecture Notes in Computer Science. Springer, 2007, p. 572-581. ISBN : 978-3-540-74467-2 (cf. p. 95).

[Cas+09] Alain CASALI, Sébastien NEDJAR, Rosine CICCHETTI, Lotfi LAKHAL et Noël NOVELLI. « Lossless Reduction of Datacubes using Partitions ». In : *Int. J. Data Warehous. Min.* 5.1 (2009), p. 18-35. DOI : 10.4018/jdwm.2009010102. URL : https://doi.org/10.4018/jdwm.2009010102 (cf. p. 74).

[CLM07] Nathalie CASPARD, Bruno LECLERC et Bernard MONJARDET. *Ensembles ordonnés finis : concepts, résultats et usages*. fr. 2007e éd. Mathématiques et Applications. Berlin, Germany : Springer, sept. 2007 (cf. p. 29).

[Cha+06] Chee Yong CHAN, H. V. JAGADISH, Kian-Lee TAN et al. « On High Dimensional Skylines ». In : *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*. Sous la dir. d'Yannis E. IOANNIDIS, Marc H. SCHOLL, Joachim W. SCHMIDT et al. T. 3896. Lecture Notes in Computer Science. Springer, 2006, p. 478-495. DOI : 10.1007/11687238_30. URL : https://doi.org/10.1007/11687238%5C_30 (cf. p. 219).

[Châ50] Jean CHÂTEAU. *L'enfant et le jeu*. T. 1. Éditions du scarabée, 1950, p. 184 (cf. p. 122).

[Châ89] Jean CHÂTEAU. *Sous couleur de jouer : le métaphorique ludique*. Paris : José Corti, 1989 (cf. p. 122).

[CD97] Surajit CHAUDHURI et Umeshwar DAYAL. « An Overview of Data Warehousing and OLAP Technology ». In : *SIGMOD Record* 26.1 (1997), p. 65-74 (cf. p. 28).

[CDN11] Surajit CHAUDHURI, Umeshwar DAYAL et Vivek R. NARASAYYA. « An overview of business intelligence technology ». In : *Commun. ACM* 54.8 (2011), p. 88-98. DOI : 10.1145/1978542.1978562. URL : https://doi.org/10.1145/1978542.1978562 (cf. p. 28).

[CGL15] L. CHEN, C. GANG et Q. LI. « Efficient algorithms for finding the most desirable skyline objects ». Anglais. In : *Knowledge Based Systems* 89 (2015), p. 250-264 (cf. p. 219).

[CM89] Meng Chang CHEN et Lawrence MCNAMEE. « On the Data Model and Access Method of Summary Data Management ». In : *IEEE Trans. Knowl. Data Eng.* 1.4 (1989), p. 519-529 (cf. p. 41).

[CYH06] Hong CHENG, Philip S. YU et Jiawei HAN. « AC-Close : Efficiently Mining Approximate Closed Itemsets by Core Pattern Recovery ». In : *ICDM*. Sous la dir. d'Yong SHI et Christopher W. CLIFTON. IEEE Computer Society, 2006, p. 839-844 (cf. p. 53).

[CTS19] Alexander CHO, A. TSAASAN et Constance STEINKUEHLER. « The building blocks of an educational esports league : lessons from year one in orange county high schools ». In : août 2019, p. 1-11. ISBN : 978-1-4503-7217-6. DOI : 10.1145/3337722.3337738 (cf. p. 259).

[Cho+05] Jan CHOMICKI, Parke GODFREY, Jarek GRYZ et al. « Skyline with Presorting : Theory and Optimizations ». In : *Intelligent Information Systems*. Sous la dir. de Mieczyslaw A. KLOPOTEK, Slawomir T. WIERZCHON et Krzysztof TROJANOWSKI. Advances in Soft Computing. Springer, 2005, p. 595-604. ISBN : 3-540-25056-5 (cf. p. 83, 201).

[Cod90] E. F. CODD. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990. ISBN : 0-201-14192-2 (cf. p. 28).

[CS02] Yann COLLETTE et Patrick SIARRY. *Optimisation multiobjectif*. fr. Paris, France : Editions Eyrolles, 2002 (cf. p. 75).

[Cox89] H S M COXETER. *Introduction to Geometry*. 2e éd. Wiley Classics Library. Nashville, TN : John Wiley et Sons, fév. 1989 (cf. p. 29).

[Cra84] Chris CRAWFORD. *The art of computer game design*. New York, NY : Osborne/McGraw-Hill, mai 1984 (cf. p. 123, 140).

[Cra03] Chris CRAWFORD. *Chris Crawford on Game Design*. Jan. 2003. ISBN : 9780131460997 (cf. p. 135, 140).

[Csi09] Mihaly CSIKSZENTMIHALYI. *Flow : the psychology of optimal experience*. New York : Harper [and] Row, 2009, p. 303. ISBN : 9780061339202 0061339202. URL : http://www.worldcat.org/oclc/553803226 (cf. p. 20, 132, 140).

[CB17] Mihaly CSIKSZENTMIHALYI et Léandre BOUFFARD. « LE POINT SUR LE FLOW ». In : *Revue québécoise de psychologie* 38 (juin 2017), p. 65. DOI : 10.7202/1040070ar (cf. p. 133).

[CM85] Mihaly CSIKSZENTMIHALYI et Fausto MASSIMINI. « On the Psychological Selection of Bio-Cultural Information ». In : *New Ideas in Psychology* 3 (déc. 1985), p. 115-138. DOI : 10.1016/0732-118X(85)90002-9 (cf. p. 132).

[Dan06] Tran Khanh DANG. « The SH-Tree : A Novel and Flexible Super Hybrid Index Structure for Similarity Search on Multidimensional Data ». In : *IJCSA* 3.1 (2006), p. 1-25 (cf. p. 81).

[Deb01] Kalyanmoy DEB. *Multi-objective optimization using evolutionary algorithms*. Wiley-Interscience series in systems and optimization. Wiley, 2001. ISBN : 978-0-471-87339-6 (cf. p. 75).

[DMB11] Antonella DELLE FAVE, Fausto MASSIMINI et Marta BASSI. « Psychological Selection and Optimal Experience ». In : *Psychological Selection and Optimal Experience Across Cultures : Social Empowerment Through Personal Growth* (oct. 2011). DOI : 10.1007/978-90-481-9876-4_3 (cf. p. 132).

[Dha05] Clarisse DHAENENS. *Optimisation Combinatoire Multi-Objectif : Apport des méthodes coopératives et contribution à l'extraction de connaissances. (Multi-objective combinatorial Optimization : contribution of cooperative methods and application to knowledge discovery)*. 2005. URL : https://tel.archives-ouvertes.fr/tel-00178895 (cf. p. 75).

[DV98] Le DIBERDER A LE DIBERDER FRÉDÉRIC. L'UNIVERS DES et Jeux VIDÉO. « L'univers Des Jeux Vidéo ». In : (1998). Sous la dir. d'Ed la DÉCOUVERTE (cf. p. 127).

[Dio+02] Cheikh Talibouya DIOP, Arnaud GIACOMETTI, Dominique LAURENT et al. « Composition of Mining Contexts for Efficient Extraction of Association Rules ». In : *EDBT*. Sous la dir. de Christian S. JENSEN, Keith G. JEFFERY, Jaroslav POKORNÝ et al. T. 2287. Lecture Notes in Computer Science. Springer, 2002, p. 106-123. ISBN : 3-540-43324-4 (cf. p. 98, 207).

[Dja+08] Damien DJAOUTI, Julian ALVAREZ, Jean-Pierre JESSEL et al. « PLAY, GAME, WORLD : ANATOMY OF A VIDEOGAME ». In : (jan. 2008) (cf. p. 135).

[Doa+18] Minh Tuan DOAN, Jianzhong QI, Sutharshan RAJASEGARAR et al. « Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data ». In : *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*. Sous la dir. de Naoki ABE, Huan LIU, Calton PU et al. IEEE, 2018, p. 106-111. DOI : 10.1109/BigData.2018.8622122. URL : https://doi.org/10.1109/BigData.2018.8622122 (cf. p. 55).

[DL99] Guozhu DONG et Jinyan LI. « Efficient Mining of Emerging Patterns : Discovering Trends and Differences ». In : *KDD*. 1999, p. 43-52 (cf. p. 86).

[DL05] Guozhu DONG et Jinyan LI. « Mining border descriptions of emerging patterns from dataset pairs ». In : *Knowl. Inf. Syst.* 8.2 (2005), p. 178-202 (cf. p. 18, 68).

[Dou16] Imen DOUDECH. « Détection des bots dans les jeux vidéo multijoueurs en ligne en appliquant les techniques de forage de données ». In : (sept. 2016) (cf. p. 262).

[EG00] Matthias EHRGOTT et Xavier GANDIBLEUX. « A survey and annotated bibliography of multiobjective combinatorial optimization ». In : *OR Spectr.* 22.4 (2000), p. 425-460. DOI : 10.1007/s002910000046. URL : https://doi.org/10.1007/s002910000046 (cf. p. 75).

[EG95] Thomas EITER et Georg GOTTLOB. « Identifying the Minimal Transversals of a Hypergraph and Related Problems ». In : *SIAM J. Comput.* 24.6 (1995), p. 1278-1304 (cf. p. 56).

[Elo08] A E ELO. *The Rating of Chessplayers Past and Present*. Bronx N.Y : Ishi Press International, 2008 (cf. p. 23, 255).

[Fav+13] Cécile FAVRE, Fadila BENTAYEB, Omar BOUSSAID et al. « Les entrepôts de données pour les nuls... ou pas! » In : jan. 2013 (cf. p. 21, 144, 153).

[FU96] Usama M. FAYYAD et Ramasamy UTHURUSAMY. « Data Mining and Knowledge Discovery in Databases (Introduction to the Special Section) ». In : *Commun. ACM* 39.11 (1996), p. 24-26. DOI : 10.1145/240455.240463. URL : https://doi.org/10.1145/240455.240463 (cf. p. 27).

[FKS13] Clive FULLAGAR, Patrick KNIGHT et Heather SOVERN. « Challenge/Skill Balance, Flow, and Performance Anxiety ». In : *Applied Psychology An International Review* 62 (avr. 2013). DOI : 10.1111/j.1464-0597.2012.00494.x (cf. p. 132).

[Ful08] Tracy FULLERTON. *Game Design Workshop : A Playcentric Approach to Creating Innovative Games*. Fév. 2008. ISBN : 9780429188008. DOI : 10.1201/b13172 (cf. p. 20, 121, 126, 130, 135).

[Gan93] Jean-Gabriel GANASCIA. « TDIS : an Algebraic Formalization ». In : *IJCAI*. 1993, p. 1008-1015 (cf. p. 68).

[GW99] Bernhard GANTER et Rudolf WILLE. *Formal Concept Analysis : Mathematical Foundations*. Springer, 1999 (cf. p. 29, 69, 72, 188, 192, 193, 195).

[GL11] S GENVO et M LETOURNEUX. *De Jeux Vidéo : Approches De L'expression Vidéoludique*. Sous la dir. de Éditions L'HARMATTAN". Paris, 2011 (cf. p. 128).

[GSG05] Parke GODFREY, Ryan SHIPLEY et Jarek GRYZ. « Maximal Vector Computation in Large Data Sets ». In : *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. Sous la dir. de Klemens BÖHM, Christian S. JENSEN, Laura M. HAAS et al. ACM, 2005, p. 229-240. URL : http://www.vldb.org/archives/website/2005/program/paper/tue/p229-godfrey.pdf (cf. p. 79).

[GSG07] Parke GODFREY, Ryan SHIPLEY et Jarek GRYZ. « Algorithms and analyses for maximal vector computation ». In : *VLDB J.* 16.1 (2007), p. 5-28 (cf. p. 79).

[Gol89] David E. GOLDBERG. « Sizing Populations for Serial and Parallel Genetic Algorithms ». In : *Proceedings of the 3rd International Conference on Genetic Algorithms, George Mason University, Fairfax, Virginia, USA, June 1989*. Sous la dir. de J. David SCHAFFER. Morgan Kaufmann, 1989, p. 70-79 (cf. p. 76).

[GS04] Sreenivas GOLLAPUDI et D. SIVAKUMAR. « Framework and algorithms for trend analysis in massive temporal data sets ». In : *CIKM*. Sous la dir. de David GROSSMAN, Luis GRAVANO, ChengXiang ZHAI et al. ACM, 2004, p. 168-177. ISBN : 1-58113-874-1 (cf. p. 110).

[GB79] J R GOODY et A BENSA. « La Domestication De La Pensée Sauvage ». In : (1979). Sous la dir. d'EDITIONS DE MINUIT (cf. p. 122).

[GZ05] Karam GOUDA et Mohammed Javeed ZAKI. « GenMax : An Efficient Algorithm for Mining Maximal Frequent Itemsets ». In : *Data Min. Knowl. Discov.* 11.3 (2005), p. 223-242 (cf. p. 87).

[GLW00] Gösta GRAHNE, Laks V. S. LAKSHMANAN et Xiaohong WANG. « Efficient Mining of Constrained Correlated Sets ». In : *ICDE*. 2000, p. 512-521 (cf. p. 65).

[Gra+97] Jim GRAY, Surajit CHAUDHURI, Adam BOSWORTH et al. « Data Cube : A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals ». In : *Data Min. Knowl. Discov.* 1.1 (1997), p. 29-53 (cf. p. 28, 29, 34, 38, 44, 86, 87, 155, 166, 172, 207).

[Gru+97] Torsten GRUST, Joachim KRÖGER, Dieter GLUCHE et al. « Query Evaluation in CROQUE - Calculus and Algebra Coincide ». In : *BNCOD*. Sous la dir. de Carol SMALL, Paul DOUGLAS, Roger G. JOHNSON et al. T. 1271. Lecture Notes in Computer Science. Springer, 1997, p. 84-100. ISBN : 3-540-63263-8 (cf. p. 80).

[GN05] Emmanuel GUARDIOLA et Stéphane NATKIN. « Game Theory and video game, a new approach of game theory to analyze and conceive game systems ». In : jan. 2005 (cf. p. 20, 130).

[Gun+97] Dimitrios GUNOPULOS, Roni KHARDON, Heikki MANNILA et al. « Data mining, Hypergraph Transversals, and Machine Learning ». In : *PODS*. Sous la dir. d'Alberto MENDELZON et Z. Meral ÖZSOYOGLU. Chairman-Mendelzon, Alberto and Chairman-Özsoyoglu, Z. Meral. Tucson, Arizona, United States : ACM, 1997, p. 209-216. ISBN : 0-89791-910-6 (cf. p. 56).

[Gut84] Antonin GUTTMAN. « R-Trees : A Dynamic Index Structure for Spatial Searching ». In : *SIGMOD Conference*. Sous la dir. de Beatrice YORMARK. ACM Press, 1984, p. 47-57 (cf. p. 81).

[HT98] Laura M. HAAS et Ashutosh TIWARY, éd. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998. ISBN : 0-89791-995-5.

[Han+05] Jiawei HAN, Yixin CHEN, Guozhu DONG et al. « Stream Cube : An Architecture for Multi-Dimensional Analysis of Data Streams ». In : *Distributed and Parallel Databases* 18.2 (2005), p. 173-197 (cf. p. 53, 87).

[HKP11] Jiawei HAN, Micheline KAMBER et Jian PEI. *Data Mining : Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011. ISBN : 978-0123814791. URL : http://hanj.cs.illinois.edu/bk3/ (cf. p. 27, 40, 42, 65, 68, 69, 169).

[Han+01] Jiawei HAN, Jian PEI, Guozhu DONG et al. « Efficient Computation of Iceberg Cubes with Complex Measures ». In : *SIGMOD Conference*. 2001, p. 1-12 (cf. p. 53).

[HPY00] Jiawei HAN, Jian PEI et Yiwen YIN. « Mining Frequent Patterns without Candidate Generation ». In : *SIGMOD Conference*. Sous la dir. de Weidong CHEN, Jeffrey F. NAUGHTON et Philip A. BERNSTEIN. ACM, 2000, p. 1-12. ISBN : 1-58113-218-2 (cf. p. 53).

[HWM16] Nicolas HANUSSE, Patrick Kamnang WANKO et Sofian MAABOUT. « Computing and Summarizing the Negative Skycube ». In : *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. Sous la dir. de Snehasis MUKHOPADHYAY, ChengXiang ZHAI, Elisa BERTINO et al. ACM, 2016, p. 1733-1742. DOI : 10.1145/2983323.2983759. URL : https://doi.org/10.1145/2983323.2983759 (cf. p. 84).

[HRU96] Venky HARINARAYAN, Anand RAJARAMAN et Jeffrey D. ULLMAN. « Implementing Data Cubes Efficiently ». In : *SIGMOD Conference*. Sous la dir. de H. V. JAGADISH et Inderpal Singh MUMICK. ACM Press, 1996, p. 205-216 (cf. p. 29).

[Hee17] Bob HEERE. « Embracing the sportification of society : Defining e-sports through a polymorphic view on sport ». In : *Sport Management Review* 21 (juill. 2017). DOI : 10.1016/j.smr.2017.07.002 (cf. p. 23, 247, 250, 259).

[HV99] D HILBERT et K David VOLKERT. « Grundlagen Der Geometrie (Festschrift 1899) ». In : (1899). Sous la dir. de Grundlagen Der GEOMETRIE (cf. p. 29).

[HV12] K. HOSE et A. VLACHOU. « A survey of skyline processing in highly distributed environments ». In : *VLDB J.* 21.3 (2012), p. 359-384. DOI : 10.1007/s00778-011-0246-6. URL : https://doi.org/10.1007/s00778-011-0246-6 (cf. p. 81).

[Hua08] Jingwen HUANG. « Combining Entropy Weight and TOPSIS Method for Information System Selection ». In : oct. 2008, p. 1281-1284. DOI : 10.1109/ICCIS.2008.4670971 (cf. p. 224).

[Hui51] J HUIZINGA. *Homo Ludens - Essai sur la fonction sociale du jeu*. Jan. 1951 (cf. p. 123, 140).

[Hun05] Robin HUNICKE. « The case for dynamic difficulty adjustment in games ». In : t. 265. Jan. 2005, p. 429-433. DOI : 10.1145/1178477.1178573 (cf. p. 135).

[HLZ04] Robin HUNICKE, Marc LEBLANC et Robert ZUBEK. « MDA : A formal approach to game design and game research ». In : *In Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence*. Press, 2004, p. 1-5 (cf. p. 19, 124, 130, 140).

[HM02] Carlos A. HURTADO et Alberto O. MENDELZON. « OLAP Dimension Constraints ». In : *PODS*. Sous la dir. de Lucian POPA. ACM, 2002, p. 169-179 (cf. p. 20, 144, 261).

[Inm96] W. H. INMON. *Building the Data Warehouse (2nd Ed.)* USA : John Wiley et Sons, Inc., 1996. ISBN : 0471141615 (cf. p. 27, 153, 172).

[JM96] H. V. JAGADISH et Inderpal Singh MUMICK, éd. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*. ACM Press, 1996.

[Jr98] Roberto J. Bayardo JR. « Efficiently Mining Long Patterns from Databases ». In : *SIGMOD Conference*. Sous la dir. de Laura M. HAAS et Ashutosh TIWARY. ACM Press, 1998, p. 85-93. ISBN : 0-89791-995-5 (cf. p. 87).

[JAG00] Roberto J. Bayardo JR., Rakesh AGRAWAL et Dimitrios GUNOPULOS. « Constraint-Based Rule Mining in Large, Dense Databases ». In : *Data Min. Knowl. Discov.* 4.2/3 (2000), p. 217-240 (cf. p. 65).

[Juu11] J JUUL. *Video Games between Real Rules and Fictional Worlds*. Cambridge Mass : MIT Press, 2011 (cf. p. 122).

[Juu03] Jesper JUUL. « The Game, the Player, the World : Looking for a Heart of Gameness ». In : jan. 2003 (cf. p. 130).

[KM11] Masato KAWABATA et Cliff MALLETT. « Flow experience in physical activity : Examination of the internal structure of flow from a process-related perspective ». In : *Motivation and Emotion* 35 (déc. 2011), p. 393-402. DOI : 10.1007/s11031-011-9221-1 (cf. p. 132).

[Kel00] Catherine KELLNER. « La médiation par le cédérom "ludo-éducatif" : approche communicationnelle ». Thèse de doct. Déc. 2000 (cf. p. 123).

[Kli03] C KLIMMT. *Dimensions and Determinants of the Enjoyment of Playing Digital Games. Copier M.* Utrecht The Netherlands, 2003 (cf. p. 136).

[KRR02] Donald KOSSMANN, Frank RAMSAK et Steffen ROST. « Shooting Stars in the Sky : An Online Algorithm for Skyline Queries ». In : *VLDB*. Sous la dir. de Frederick H. LOCHOVSKY et Wang SHAN. Morgan Kaufmann, 2002, p. 275-286 (cf. p. 80).

[Kos12] Raph KOSTER. 2012. URL : https://www.raphkoster.com/2012/01/24/an-atomic-theory-of-fun-game-design/ (visité le 22/12/2020) (cf. p. 128).

[Kos14] Raph KOSTER. *A theory of fun for game design*. CA : O'Reilly Media Inc., 2014, p. 279. ISBN : 9781449363215 1449363210. URL : http://www.worldcat.org/oclc/858311849 (cf. p. 20, 131, 140).

[KM00] Nikolaos KOTSIS et Douglas R. MCGREGOR. « Elimination of Redundant Views in Multidimensional Aggregates ». In : *DaWaK*. Sous la dir. d'Yahiko KAMBAYASHI, Mukesh K. MOHANIA et A. Min TJOA. T. 1874. Lecture Notes in Computer Science. Springer, 2000, p. 146-161. ISBN : 3-540-67980-4 (cf. p. 29).

[KV16] Bart KUIJPERS et Alejandro VAISMAN. *A Formal Algebra for OLAP*. 2016. arXiv : 1609.05020 [cs.DB] (cf. p. 20, 144).

[KLP75] H. T. KUNG, Fabrizio LUCCIO et Franco P. PREPARATA. « On Finding the Maxima of a Set of Vectors ». In : *J. ACM* 22.4 (1975), p. 469-476 (cf. p. 78-80).

[LLH94] Young-Jou LAI, Ting-Yun LIU et Ching-Lai HWANG. « TOPSIS for MODM ». In : *European Journal of Operational Research* 76.3 (1994), p. 486-500. URL : https://EconPapers.repec.org/RePEc:eee:ejores:v:76:y:1994:i:3:p:486-500 (cf. p. 222).

[LNC17] Lotfi LAKHAL, Sébastien NEDJAR et Rosine CICCHETTI. « Multidimensional skyline analysis based on agree concept lattices ». In : *Intell. Data Anal.* 21.5 (2017), p. 1245-1265. DOI : 10.3233/IDA-163111. URL : https://doi.org/10.3233/IDA-163111 (cf. p. 188, 219).

[LPH02] Laks V. S. LAKSHMANAN, Jian PEI et Jiawei HAN. « Quotient Cube : How to Summarize the Semantics of a Data Cube ». In : *VLDB*. Sous la dir. de Frederick H. LOCHOVSKY et Wang SHAN. Morgan Kaufmann, 2002, p. 778-789 (cf. p. 18, 72, 87).

[Lap+02] Marc LAPORTE, Noel NOVELLI, Rosine CICCHETTI et al. « Computing Full and Iceberg Datacubes Using Partitions ». In : *ISMIS*. 2002, p. 244-254 (cf. p. 74).

[LeB04] Marc LEBLANC. *Mechanics, Dynamics, Aesthetics : A Formal Approach to Game Design*. Fév. 2004 (cf. p. 19, 124).

[LH10] Jongwuk LEE et Seung-won HWANG. « QSkycube : Efficient Skycube Computation using Point-Based Space Partitioning ». In : *Proc. VLDB Endow.* 4.3 (2010), p. 185-196. DOI : 10.14778/1929861.1929865. URL : http://www.vldb.org/pvldb/vol4/p185-lee.pdf (cf. p. 83).

[Lev11] Guillaume LEVIEUX. « Mesure de la difficulté des jeux vidéo ». Theses. Conservatoire national des arts et metiers - CNAM, mai 2011. URL : https://theses.hal.science/tel-00612657 (cf. p. 135).

[Li+21] Chuanwen LI, Yu GU, Jianzhong QI et al. « SkyCell : A Space-Pruning Based Parallel Skyline Algorithm ». In : *CoRR* abs/2107.09993 (2021). arXiv : 2107.09993. URL : https://arxiv.org/abs/2107.09993 (cf. p. 204).

[Li+13] Nan LI, Ziyu GUAN, Lijie REN et al. « gIceberg : Towards iceberg analysis in large graphs ». In : *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*. Sous la dir. de Christian S. JENSEN, Christopher M. JERMAINE et Xiaofang ZHOU. IEEE Computer Society, 2013, p. 1021-1032. DOI : 10.1109/ICDE.2013.6544894. URL : https://doi.org/10.1109/ICDE.2013.6544894 (cf. p. 53).

[LHP01] Wenmin LI, Jiawei HAN et Jian PEI. « CMAR : Accurate and Efficient Classification Based on Multiple Class-Association Rules ». In : *ICDM*. Sous la dir. de Nick CERCONE, Tsau Young LIN et Xindong WU. IEEE Computer Society, 2001, p. 369-376. ISBN : 0-7695-1119-8 (cf. p. 56, 68).

[LHM98] Bing LIU, Wynne HSU et Yiming MA. « Integrating Classification and Association Rule Mining ». In : *KDD*. 1998, p. 80-86 (cf. p. 56, 68).

[Liu+12] Qing LIU, Yunjun GAO, Gang CHEN et al. « On Efficient Reverse k-Skyband Query Processing ». In : *Database Systems for Advanced Applications - 17th International Conference, DASFAA 2012, Busan, South Korea, April 15-19, 2012, Proceedings, Part I*. Sous la dir. de Sang-goo LEE, Zhiyong PENG, Xiaofang ZHOU et al. T. 7238. Lecture Notes in Computer Science. Springer, 2012, p. 544-559. DOI : 10.1007/978-3-642-29038-1_39. URL : https://doi.org/10.1007/978-3-642-29038-1%5C_39 (cf. p. 84).

[Liu+23] Ziyi LIU, Lei LI, Mengxuan ZHANG et al. « Multi-constraint shortest path using forest hop labeling ». In : *VLDB J.* 32.3 (2023), p. 595-621. DOI : 10.1007/s00778-022-00760-2. URL : https://doi.org/10.1007/s00778-022-00760-2 (cf. p. 81, 188).

[LS02] Frederick H. LOCHOVSKY et Wang SHAN, éd. *Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. Morgan Kaufmann, 2002.

[LPL02] Stéphane LOPES, Jean-Marc PETIT et Lotfi LAKHAL. « Functional and approximate dependency mining : database and FCA points of view ». In : *J. Exp. Theor. Artif. Intell.* 14.2-3 (2002), p. 93-114 (cf. p. 65, 74, 190).

[LF10] Farhad Hosseinzadeh LOTFI et Reza FALLAHNEJAD. « Imprecise Shannon's Entropy and Multi Attribute Decision Making ». In : *Entropy* 12.1 (2010), p. 53-62. DOI : 10.3390/e12010053. URL : https://doi.org/10.3390/e12010053 (cf. p. 224).

[Mad15] Stéphanie MADER. « Le "game design" de jeux thérapeutiques : Modèles et méthodes pour la conception du gameplay. (Game Design of Therapeutic Games : Models and methods to design the gameplay) ». Thèse de doct. Conservatoire national des arts et métiers, Paris, France, 2015. URL : https://tel.archives-ouvertes.fr/tel-01344418 (cf. p. 133).

[MZ04] Elzbieta MALINOWSKI et Esteban ZIMÁNYI. « OLAP Hierarchies : A Conceptual Perspective ». In : *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings*. 2004, p. 477-491. DOI : 10.1007/978-3-540-25975-6_34. URL : https://doi.org/10.1007/978-3-540-25975-6_34 (cf. p. 144).

[Mal80] Thomas MALONE. « What makes things fun to learn? heuristics for designing instructional computer games ». In : *SIGSMALL '80 : Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. New York, NY, USA : ACM, sept. 1980, p. 162-169 (cf. p. 133, 137).

[Mal81] Thomas MALONE. « Toward a Theory of Intrinsically Motivating Instruction* ». In : *Cognitive Science* 5 (oct. 1981), p. 333-369. DOI : 10.1016/S0364-0213(81)80017-1 (cf. p. 137).

[Mam13] Mohamed MAMMERI. « Une approche d'aide multicritère à la décision pour l'évaluation du confort dans les trains : construction d'un modèle d'évaluation. (A multiple criteria decision aiding tool for evaluating the overall comfort on board trains) ». Thèse de doct. Paris Dauphine University, France, 2013. URL : https://tel.archives-ouvertes.fr/tel-00915734 (cf. p. 75).

[MT96] Heikki MANNILA et Hannu TOIVONEN. « Multiple Uses of Frequent Sets and Condensed Representations (Extended Abstract) ». In : *KDD*. 1996, p. 189-194 (cf. p. 74).

[MT97] Heikki MANNILA et Hannu TOIVONEN. « Levelwise Search and Borders of Theories in Knowledge Discovery ». In : *Data Min. Knowl. Discov.* 1.3 (1997), p. 241-258 (cf. p. 60, 68).

[MNL23] Mickaël MARTIN NEVOT, Sébastien NEDJAR et Lotfi LAKHAL. « Hierarchical datacubes ». In : *Journal of Computer and Communications* Vol.11 No.6, June 2023 (juin 2023), p. 43-72. DOI : 10.4236/jcc.2023.116004 (cf. p. 3, 144).

[Mar+19] Mickaël MARTIN NEVOT, Sébastien NEDJAR, Lotfi LAKHAL et Rosine CICCHETTI. « C-Idea - A Fast Algorithm for Computing Emerging Closed Datacubes ». In : *Utilizing Big Data Paradigms for Business Intelligence*. 2019, p. 129-169. DOI : 10.4018/978-1-5225-4963-5.ch005. URL : https:

//hal.archives-ouvertes.fr/hal-02446921 (cf. p. 3, 21, 175).

[McB75] Frank MCBRIDE. « TOWARD A NON-DEFINITION OF SPORT ». In : *Journal of the Philosophy of Sport* 2.1 (1975), p. 4-11. DOI : 10.1080/00948705.1975.10654092. eprint : https://doi.org/10.1080/00948705.1975.10654092. URL : https://doi.org/10.1080/00948705.1975.10654092 (cf. p. 249).

[MÖ97] Alberto MENDELZON et Z. Meral ÖZSOYOGLU, éd. *PODS '97 : Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. Chairman-Mendelzon, Alberto and Chairman-Özsoyoglu, Z. Meral. Tucson, Arizona, United States : ACM, 1997. ISBN : 0-89791-910-6.

[Mie98] Kaisa MIETTINEN. *Nonlinear multiobjective optimization*. T. 12. International series in operations research and management science. Kluwer, 1998. ISBN : 978-0-7923-8278-2 (cf. p. 75).

[Mis15] Olana MISSURA. « Dynamic Difficulty Adjustment ». Thèse de doct. Oct. 2015 (cf. p. 135).

[Mit82] Tom M. MITCHELL. « Generalization as Search ». In : *Artificial Intelligence*. T. 18(2). 1982, p. 203-226 (cf. p. 57, 156, 172).

[Mit97] Tom M. MITCHELL. *Machine learning*. MacGraw-Hill Series in Computer Science, 1997 (cf. p. 57, 156, 172).

[MC96] Giovanni MONETA et Mihaly CSIKSZENTMIHALYI. « The Effect of Perceived Challenges and Skills on the Quality of Subjective Experience ». In : *Journal of personality* 64 (juill. 1996), p. 275-310. DOI : 10.1111/j.1467-6494.1996.tb00512.x (cf. p. 132).

[MH03] Ph MORA et Stéphane HÉAS. « Du joueur de jeux vidéo à l'e-sportif : vers un professionnalisme florissant de l'élite ? » In : *Consommations et Sociétés* (2003), p. 129-145. URL : https://shs.hal.science/halshs-00346159 (cf. p. 247).

[MI08] Konstantinos MORFONIOS et Yannis E. IOANNIDIS. « Supporting the data cube lifecycle : the power of ROLAP ». In : *VLDB J.* 17.4 (2008), p. 729-764 (cf. p. 46, 99, 262).

[Mor+07] Konstantinos MORFONIOS, Stratis KONAKAS, Yannis E. IOANNIDIS et al. « ROLAP implementations of the data cube ». In : *ACM Comput. Surv.* 39.4 (2007) (cf. p. 40).

[MPJ07] Michael D. MORSE, Jignesh M. PATEL et H. V. JAGADISH. « Efficient Skyline Computation over Low-Cardinality Domains ». In : *VLDB*. Sous la dir. de Christoph KOCH, Johannes GEHRKE, Minos N. GAROFALAKIS et al. ACM, 2007, p. 267-278. ISBN : 978-1-59593-649-3 (cf. p. 77).

[NBM13] Lennart NACKE, Chris BATEMAN et Regan MANDRYK. « BrainHex : A neurobiological gamer typology survey ». In : *Entertainment Computing* 5 (jan. 2013). DOI : 10.1016/j.entcom.2013.06.002 (cf. p. 125).

[NC12] J. NAKAMURA et Mihaly CSIKSZENTMIHALYI. « Flow Theory and Research ». In : *Oxford Handbook of Positive Psychology* (jan. 2012), p. 195-206. DOI : 10.1093/oxfordhb/9780195187243.013.0018 (cf. p. 133).

[Nas50] John F NASH. « Equilibrium points in n-person games ». In : *The Essential John Nash*. Sous la dir. de National Academy of SCIENCES. Princeton : Princeton University Press, jan. 1950, p. 48-49 (cf. p. 20, 139, 141).

[NB03] George E. NASR et Celine BADR. « BUC Algorithm for Iceberg Cubes : Implementation and Sensitivity Analysis ». In : *FLAIRS Conference*. Sous la dir. d'Ingrid RUSSELL et Susan M. HALLER. AAAI Press, 2003, p. 255-260. ISBN : 1-57735-177-0 (cf. p. 48).

[NV04] Stéphane NATKIN et Liliana VEGA. « A Petri Net Model for Computer Games Analysis. » In : *Int. J. Intell. Games And Simulation* 3 (jan. 2004), p. 37-44 (cf. p. 126, 127).

[Ned09] Sébastien NEDJAR. « Cubes Émergents pour l'analyse des renversements de tendances dans les bases de données multidimensionnelles ». Thèse de doct. Université de Méditerranée, nov. 2009. URL : http://tel.archives-ouvertes.fr/docs/00/46/41/13/PDF/these_cubes_emergent.pdf (cf. p. 27, 87, 108).

[Ned+06a] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Cocktail de Cubes ». In : *BDA*. Sous la dir. de Dominique LAURENT. 2006 (cf. p. 110).

[Ned+06b] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Cubes convexes ». In : *Ingénierie des Systèmes d'Information* 11.6 (2006), p. 11-31 (cf. p. 110).

[Ned+07] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Emerging Cubes for Trends Analysis in Olap Databases ». In : *DaWaK*. Sous la dir. d'Il-Yeol SONG, Johann EDER et Tho Manh NGUYEN. T. 4654. Lecture Notes in Computer Science. Springer, 2007, p. 135-144. ISBN : 978-3-540-74552-5 (cf. p. 18, 86, 261).

[Ned+08] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Upper Borders for Emerging Cubes ». In : *DaWaK*. Sous la dir. d'Il-Yeol SONG, Johann EDER et Tho Manh NGUYEN. T. 5182. Lecture Notes in Computer Science. Springer, 2008, p. 45-54. ISBN : 978-3-540-85835-5 (cf. p. 110).

[Ned+09] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Emerging Cubes : Borders, size estimations and lossless reductions ». In : *Information Systems* 34.6 (2009), p. 536-550. ISSN : 0306-4379. DOI : DOI:10.1016/j.is.2009.03.001. URL : http://www.sciencedirect.com/science/article/B6V0G-4VTKKWF-2/2/42b06dc12bb8bb1eca73c03a (cf. p. 86).

[Ned+10a] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Cubes convexes ». In : *CoRR* abs/1003.5056 (2010). arXiv : 1003.5056. URL : http://arxiv.org/abs/1003.5056 (cf. p. 43).

[Ned+10b] Sébastien NEDJAR, Alain CASALI, Rosine CICCHETTI et al. « Reduced representations of Emerging Cubes for OLAP database mining ». In : *IJBIDM* 5.1 (2010), p. 268-300 (cf. p. 19, 86, 110).

[NCL11] Sébastien NEDJAR, Rosine CICCHETTI et Lotfi LAKHAL. « Extracting semantics in OLAP databases using emerging cubes ». In : *Inf. Sci.* 181.10 (2011), p. 2036-2059 (cf. p. 86).

[NLC13] Sébastien NEDJAR, Lotfi LAKHAL et Rosine CICCHETTI. « Emerging Data Cube Representations for OLAP Database Mining ». In : *Contrast Data Mining*. Sous la dir. de Guozhu DONG et James BAILEY. CRC Press, 2013, p. 109-128. ISBN : 978-1-43-985432-7 (cf. p. 86).

[Ned+11a] Sébastien NEDJAR, Fabien PESCI, Lotfi LAKHAL et al. « The Agree Concept Lattice for Multidimensional Database Analysis ». In : *Formal Concept Analysis - 9th International Conference, ICFCA 2011, Nicosia, Cyprus, May 2-6, 2011. Proceedings*. Sous la dir. de Petko VALTCHEV et Robert JÄSCHKE. T. 6628. Lecture Notes in Computer Science. Springer, 2011, p. 219-234. DOI : 10.1007/978-3-642-20514-9_17. URL : https://doi.org/10.1007/978-3-642-20514-9%5C_17 (cf. p. 188).

[Ned+11b] Sébastien NEDJAR, Fabien PESCI, Lotfi LAKHAL et al. « Treillis des Concepts Skylines : analyse multidimensionnelle des Skylines fondée sur les ensembles en accords ». In : *Extraction et gestion des connaissances (EGC'2011), Actes, 25 au 29 janvier 2011, Brest, France*. Sous la dir. d'Ali KHENCHAF et Pascal PONCELET. T. RNTI-E-20. Revue des Nouvelles Technologies de l'Information. Hermann-Éditions, 2011, p. 227-238. URL : http://editions-rnti.fr/?inprocid=1000949 (cf. p. 188).

[Nei12] Katharine NEIL. « Game Design Tools : Time to Evaluate ». In : (mai 2012) (cf. p. 20, 130, 140).

[NM72] J.L. NEUMANN et O. MORGENSTERN. *The Theory of Games and Economic Behavior*. T. 38. Jan. 1972 (cf. p. 20, 139, 141).

[Ng+98] Raymond T. NG, Laks V. S. LAKSHMANAN, Jiawei HAN et al. « Exploratory Mining and Pruning Optimizations of Constrained Association Rules ». In : *SIGMOD Conference*. Sous la dir. de Laura M. HAAS et Ashutosh TIWARY. ACM Press, 1998, p. 13-24. ISBN : 0-89791-995-5 (cf. p. 65).

[NM03] Noel NOVELLI et Sofian MAABOUT. « Algorithme Efficace de Calcul du Produit de Partitions et ses Applications ». In : *BDA*. Sous la dir. de Claude CHRISMENT. 2003 (cf. p. 74).

[OR94] Martin OSBORNE et Ariel RUBINSTEIN. *A course in Game Theory*. T. 63. Jan. 1994. DOI : 10.2307/2554642 (cf. p. 139).

[Pab12] Chloe PABERZ. « Le jeu vidéo comme sport en Corée du Sud ? » In : *Hermès* n° 62 (déc. 2012), , [p.] DOI : 10.4267/2042/48276 (cf. p. 248).

[Pap+03] Dimitris PAPADIAS, Yufei TAO, Greg FU et al. « An Optimal and Progressive Algorithm for Skyline Queries ». In : *SIGMOD Conference*. Sous la dir. d'Alon Y. HALEVY, Zachary G. IVES et AnHai DOAN. ACM, 2003, p. 467-478. ISBN : 1-58113-634-X (cf. p. 80, 81).

[Pap+05] Dimitris PAPADIAS, Yufei TAO, Greg FU et al. « Progressive skyline computation in database systems ». In : *ACM Trans. Database Syst.* 30.1 (2005), p. 41-82 (cf. p. 81, 213).

[Par96] V PARETO. *Cours d'économie politique*. Sous la dir. de Lausanne F. ROUGE. 1896 (cf. p. 75).

[Par99] Pierre PARLEBAS. *Jeux, sports et sociétés : Lexique de praxéologie motrice*. Jan. 1999. ISBN : 9782865801039. DOI : 10.4000/books.insep.1067 (cf. p. 247).

[Pas82] Moritz PASCH. *Vorlesungen uber neuere Geometrie*. 201 p. Leipzig : Teubner, 1882, 201 p. URL : http://hdl.handle.net/2027/wu.89043163294 (cf. p. 29).

[Pas+99] Nicolas PASQUIER, Yves BASTIDE, Rafik TAOUIL et al. « Efficient Mining of Association Rules Using Closed Itemset Lattices ». In : *Inf. Syst.* 24.1 (1999), p. 25-46. DOI : 10.1016/S0306-4379(99)00003-4. URL : https://doi.org/10.1016/S0306-4379(99)00003-4 (cf. p. 56).

[Pea89] G PEANO. *Augustae Taurinorum Torino : Fratres Bocca*. 1889 (cf. p. 29).

[Pei+07] Jian PEI, Ada Wai-Chee FU, Xuemin LIN et al. « Computing Compressed Multidimensional Skyline Cubes Efficiently ». In : *ICDE*. Sous la dir. d'Asuman DOGAC, Tamer OZSU et Timos SELLIS. IEEE, 2007, p. 96-105 (cf. p. 205).

[PHL04] Jian PEI, Jiawei HAN et Laks V. S. LAKSHMANAN. « Pushing Convertible Constraints in Frequent Itemset Mining ». In : *Data Min. Knowl. Discov.* 8.3 (2004), p. 227-252 (cf. p. 88, 209).

[PHM00] Jian PEI, Jiawei HAN et Runying MAO. « CLOSET : An Efficient Algorithm for Mining Frequent Closed Itemsets ». In : *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. 2000, p. 21-30 (cf. p. 53).

[Pei+05] Jian PEI, Wen JIN, Martin ESTER et al. « Catching the Best Views of Skyline : A Semantic Approach Based on Decisive Subspaces ». In : *VLDB*. 2005, p. 253-264 (cf. p. 75, 81, 204).

[Pei+06] Jian PEI, Yidong YUAN, Xuemin LIN et al. « Towards multidimensional subspace skyline analysis ». In : *ACM Trans. Database Syst.* 31.4 (2006), p. 1335-1381 (cf. p. 18, 77, 83, 86, 188, 204, 205, 261).

[Per06] Bernard PERRON. « The Heuristic Circle of Gameplay : the Case of Survival Horror ». In : 2006 (cf. p. 135).

[Pha16] Viet PHANLUONG. « A Simple Data Cube Representation for Efficient Computing and Updating ». In : *International Journal on Advances in Intelligent Systems* Vol 9 no 3 & 4 (jan. 2016) (cf. p. 99).

[Pia45] J PIAGET. *La Formation Du Symbole Chez L'enfant : Initiaton Jeu Et Rêve Image Et Représentation. Paris : Delachaux & Niestlé*. 1945 (cf. p. 122).

[PE15] Timotheus PREISINGER et Markus ENDRES. « Looking for the best, but not too many of them : multi-level and top-k skylines ». In : *International Journal on Advances in Software* (jan. 2015) (cf. p. 22, 227, 261).

[PS85] Franco P. PREPARATA et Michael Ian SHAMOS. *Computational Geometry - An Introduction*. Springer, 1985. ISBN : 3-540-96131-3 (cf. p. 78-80).

[Rab+17] Gaylor RABU, Morgane REVERCHON-BILLOT, Mickaël MARTIN NEVOT et al. « Glossaire ». In : *Les enjeux juridiques de l'e-sport*. Sous la dir. de Presses Universitaires D'AIX-MARSEILLE. Aix-en-Provence ; Aix-Marseille, sept. 2017, p. 591-615. ISBN : 9782731410709. URL : https://hal.archives-ouvertes.fr/hal-03081749 (cf. p. 3, 23, 250).

[RK01] Luc De RAEDT et Stefan KRAMER. « The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding ». In : *IJCAI*. Sous la dir. de Bernhard NEBEL. Morgan Kaufmann, 2001, p. 853-862. ISBN : 1-55860-777-3 (cf. p. 19, 86, 95).

[RD00a] Erhard RAHM et Hong Hai DO. « Data Cleaning : Problems and Current Approaches ». In : *IEEE Data Eng. Bull.* 23.4 (2000), p. 3-13. URL : http://sites.computer.org/debull/A00DEC-CD.pdf (cf. p. 23, 238).

[Raï12] Chedy RAÏSSI. « Multidimensional skylines ». In : *Actes des 8èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA 2012, Bordeaux, France, Juin 2012*. Sous la dir. de Sofian MAABOUT. T. B-8. RNTI. Hermann, 2012, p. 82. URL : http://editions-rnti.fr/?inprocid=1001217 (cf. p. 84).

[RPK10] Chedy RAÏSSI, Jian PEI et Thomas KISTER. « Computing Closed Skycubes ». In : *Proc. VLDB Endow.* 3.1 (2010), p. 838-847. DOI : 10.14778/1920841.1920948. URL : http://www.vldb.org/pvldb/vldb2010/pvldb%5C_vol3/R75.pdf (cf. p. 84).

[Rei+19] Jason REITMAN, Maria ANDERSON-COTO, Minerva WU et al. « Esports Research : A Literature Review ». In : *Games and Culture* 15 (avr. 2019), p. 155541201984089. DOI : 10.1177/1555412019840892 (cf. p. 23, 260).

[RR07] Scott RIGBY et Richard RYAN. « Rethinking Carrots : A New Method For Measuring What Players Find Most Rewarding and Motivating About Your Game ». 2007. URL : https://www.gamasutra.com/view/feature/1738/rethinking_carrots_a_new_method_.php (cf. p. 137).

[Rog14] Scott ROGERS. *Level Up ! : The Guide to Great Video Game Design*. Wiley, 2014 (cf. p. 127).

[RA03] Andrew ROLLINGS et Ernest ADAMS. *Andrew Rollings and Ernest Adams on Game Design*. Jan. 2003. ISBN : 1592730019 (cf. p. 127, 130).

[RS97] Kenneth A. ROSS et Divesh SRIVASTAVA. « Fast Computation of Sparse Datacubes ». In : *VLDB*. Sous la dir. de Matthias JARKE, Michael J. CAREY, Klaus R. DITTRICH et al. Morgan Kaufmann, 1997, p. 116-125. ISBN : 1-55860-470-7 (cf. p. 18, 29, 44, 55, 68, 72, 173).

[RKV95] Nick ROUSSOPOULOS, Stephen KELLEY et Frédéic VINCENT. « Nearest Neighbor Queries ». In : *SIGMOD Conference*. Sous la dir. de Michael J. CAREY et Donovan A. SCHNEIDER. ACM Press, 1995, p. 71-79 (cf. p. 81).

[RB03] M ROUSTANT et Manuel BOUTET. « Des jeux d'adultes ? Corporéïtés et sociabilités dans les cyberespaces ludiques ». In : *La pratique du jeu vidéo : réalité ou virtualité ?* Paris : L'Harmattan, 2003, p. 99-111 (cf. p. 23, 250).

[Roy19] Brice ROY. « Pour un dépassement des théories du game et du playGoing beyond Theories of Game and Play ». In : *Sciences du jeu* 11 (avr. 2019). DOI : 10.4000/sdj.1709 (cf. p. 122).

[RD00b] Richard RYAN et Edward DECI. « Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being ». In : *The American psychologist* 55 (fév. 2000), p. 68-78. DOI : 10.1037/0003-066X.55.1.68 (cf. p. 20, 136, 140).

[SZ04] Katie SALEN et Eric ZIMMERMAN. *Rules of play : game design fundamentals*. MIT Press, 2004. ISBN : 9780262240451. URL : http://www.worldcat.org/oclc/56961648 (cf. p. 19, 122-124, 130, 140, 262).

[Sch85] J. David SCHAFFER. « Multiple Objective Optimization with Vector Evaluated Genetic Algorithms ». In : *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985*. Sous la dir. de John J. GREFENSTETTE. Lawrence Erlbaum Associates, 1985, p. 93-100 (cf. p. 76).

[Sch14] Jesse SCHELL. *The Art of Game Design : A Book of Lenses*. CRC Press, 2014, p. 600. ISBN : 9781466598645. URL : http://www.worldcat.org/oclc/1064390850 (cf. p. 19, 121, 122, 138, 140, 141).

[SRF87] Timos K. SELLIS, Nick ROUSSOPOULOS et Christos FALOUTSOS. « The R+-Tree : A Dynamic Index for Multi-Dimensional Objects ». In : *VLDB*. Sous la dir. de Peter M. STOCKER, William KENT et Peter HAMMERSLEY. Morgan Kaufmann, 1987, p. 507-518. ISBN : 0-934613-46-X (cf. p. 81).

[SDN98] Amit SHUKLA, Prasad DESHPANDE et Jeffrey F. NAUGHTON. « Materialized View Selection for Multidimensional Datasets ». In : *VLDB*. Sous la dir. d'Ashish GUPTA, Oded SHMUELI et Jennifer WIDOM. Morgan Kaufmann, 1998, p. 488-499. ISBN : 1-55860-566-5 (cf. p. 29).

[SVC22] Surabhi SOLANKI, Seema VERMA et Kishore CHAHAR. « A Comprehensive Study of Page-Rank Algorithm ». In : avr. 2022, p. 1-10. DOI : 10.1007/978-981-16-6616-2_1 (cf. p. 262).

[Sol15] Julien SOLER. « Orion, A Generic Model for Data Mining : Application to Video Games. (Orion, un modèle générique pour la fouille de données : application aux jeux vidéo) ». Thèse de doct. University of Western Brittany, Brest, France, 2015. URL : https://tel.archives-ouvertes.fr/tel-01201960 (cf. p. 262).

[Spr+06] Pieter SPRONCK, Marc PONSEN, Ida SPRINKHUIZEN-KUYPER et al. « Adaptive game AI with dynamic scripting ». In : *Machine Learning* 63 (juin 2006), p. 217-248. DOI : 10.1007/s10994-006-6205-6 (cf. p. 135).

[Spy87] Nicolas SPYRATOS. « The Partition Model : A Deductive Database Model ». In : *ACM Trans. Database Syst.* 12.1 (1987), p. 1-37 (cf. p. 74, 188, 191).

[SA96] Ramakrishnan SRIKANT et Rakesh AGRAWAL. « Mining Quantitative Association Rules in Large Relational Tables ». In : *SIGMOD Conference*. Sous la dir. de H. V. JAGADISH et Inderpal Singh MUMICK. ACM Press, 1996, p. 1-12 (cf. p. 56, 68).

[Stu+02] Gerd STUMME, Rafik TAOUIL, Yves BASTIDE et al. « Computing iceberg concept lattices with Titanic ». In : *Data Knowl. Eng.* 42.2 (2002), p. 189-222 (cf. p. 112).

[SW05] Penelope SWEETSER et Peta WYETH. « GameFlow : a model for evaluating player enjoyment in games ». In : *Comput. Entertain.* 3.3 (2005), p. 3. DOI : 10.1145/1077246.1077253. URL : https://doi.org/10.1145/1077246.1077253 (cf. p. 132).

[Szw15] Wioleta SZWOCH. « Model of emotions for game players ». In : (juill. 2015), p. 285-290. DOI : 10.1109/HSI.2015.7170681 (cf. p. 125).

[TEO01] Kian-Lee TAN, Pin-Kwang ENG et Beng Chin OOI. « Efficient Progressive Skyline Computation ». In : *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. Sous la dir. de Peter M. G. APERS, Paolo ATZENI, Stefano CERI et al. Morgan Kaufmann, 2001, p. 301-310. URL : http://www.vldb.org/conf/2001/P301.pdf (cf. p. 80).

[Tré09] T Mythologie Des Jeux TRÉMEL LAURENT FORTIN. *Cavalier bleu*. Paris, 2009 (cf. p. 248).

[Ulu94] J ULUNGU EL TEGHEM. « Multi-objective combinatorial optimization problems : a survey ». In : *Journal of multi-criteria decision analysis* (1994), p. 83-104 (cf. p. 75).

[VPG14] George VALKANAS, Apostolos N. PAPADOPOULOS et Dimitrios GUNOPULOS. « Skyline Ranking à la IR ». In : *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014*. Sous la dir. de K. Selçuk CANDAN, Sihem AMER-YAHIA, Nicole SCHWEIKARDT et al. T. 1133. CEUR Workshop Proceedings. CEUR-WS.org, 2014, p. 182-187. URL : https://ceur-ws.org/Vol-1133/paper-31.pdf (cf. p. 22, 219, 220).

[Veb04] Oswald VEBLEN. « A system of axioms for geometry ». en. In : *Trans. Am. Math. Soc.* 5.3 (1904), p. 343-384 (cf. p. 29).

[Vel93] M. Van de VEL. *Theory of Convex Structures*. North-Holland Mathematical Library 50. North-Holland, Amsterdam, 1993 (cf. p. 66).

[Vel+14] John VELEZ, Tobias GREITEMEYER, Jodi WHITAKER et al. « Violent Video Games and Reciprocity ». In : *Communication Research* online (sept. 2014). DOI : 10.1177/0093650214552519 (cf. p. 259).

[Vij+96] T. M. VIJAYARAMAN, Alejandro P. BUCHMANN, C. MOHAN et al., éd. *Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*. Morgan Kaufmann, 1996. ISBN : 1-55860-382-4.

[VV10] A. VLACHOU et M. VAZIRGIANNIS. « Ranking the sky : Discovering the importance of SKYLINE points through subspace dominance relationships ». Anglais. In : *Data and Knowledge Engineering* 69 (2010), p. 943-964 (cf. p. 219).

[Vog95] H VOGT. *Leçons Sur La Résolution Algébrique Des Équations*. Paris : Nony, 1895 (cf. p. 30).

[WHL07] Jianyong WANG, Jiawei HAN et Chun LI. « Frequent Closed Sequence Mining without Candidate Maintenance ». In : *IEEE Trans. Knowl. Data Eng.* 19.8 (2007), p. 1042-1056 (cf. p. 53).

[WH12] Liz WANG et Daniel HSIAO. « Antecedents of flow in retail store shopping ». In : *Journal of Retailing and Consumer Services* 19 (juill. 2012), p. 381-389. DOI : 10.1016/j.jretconser.2012.03.002 (cf. p. 132).

[Wan+21] Quankun WANG, Jinguo YOU, Benyuan ZOU et al. « Reduced Quotient Cube : Maximize Query Answering Capacity in OLAP ». In : *IEEE Access* 9 (2021), p. 141524-141535. DOI : 10.1109/ACCESS.2021.3120278. URL : https://doi.org/10.1109/ACCESS.2021.3120278 (cf. p. 72).

[Wan+19] Wenlu WANG, Ji ZHANG, Min-Te SUN et al. « A scalable spatial skyline evaluation system utilizing parallel independent region groups ». In : *VLDB J.* 28.1 (2019), p. 73-98. DOI : 10.1007/s00778-018-0519-4. URL : https://doi.org/10.1007/s00778-018-0519-4 (cf. p. 204).

[Whi59] Robert WHITE. « Motivation Reconsidered : The Concept Of Competence ». In : *Psychological review* 66 (oct. 1959), p. 297-333. DOI : 10.1037/h0040934 (cf. p. 136).

[XZ06] Tian XIA et Donghui ZHANG. « Refreshing the sky : the compressed skycube with efficient support for frequent updates ». In : *SIGMOD Conference*. Sous la dir. de Surajit CHAUDHURI, Vagelis HRISTIDIS et Neoklis POLYZOTIS. ACM, 2006, p. 491-502. ISBN : 1-59593-256-9 (cf. p. 204).

[Xin+07] Dong XIN, Jiawei HAN, Xiaolei LI et al. « Computing Iceberg Cubes by Top-Down and Bottom-Up Integration : The StarCubing Approach ». In : *IEEE Trans. Knowl. Data Eng.* 19.1 (2007), p. 111-126 (cf. p. 53).

[Xin+06] Dong XIN, Zheng SHAO, Jiawei HAN et al. « C-Cubing : Efficient Computation of Closed Cubes by Aggregation-Based Checking ». In : *ICDE*. Sous la dir. de Ling LIU, Andreas REUTER, Kyu-Young WHANG et al. IEEE Computer Society, 2006, p. 4 (cf. p. 53).

[YH07] Georgios YANNAKAKIS et John HALLAM. « Game and Player Feature Selection for Entertainment Capture ». In : mai 2007, p. 244-251. ISBN : 1-4244-0709-5. DOI : 10.1109/CIG.2007.368105 (cf. p. 135).

[Yee06] Nick YEE. « Motivations for Play in Online Games ». In : *Cyberpsy., Behavior, and Soc. Networking* 9.6 (2006), p. 772-775. DOI : 10.1089/cpb.2006.9.772. URL : https://doi.org/10.1089/cpb.2006.9.772 (cf. p. 20, 126).

[YM07] M. YIU et N. MAMOULIS. « Efficient Processing of Top-k Dominating Queries on Multidimensional Data ». Anglais. In : *Proceedings of the VLDB conference*. 2007, p. 483-494 (cf. p. 219).

[Yua+05] Yidong YUAN, Xuemin LIN, Qing LIU et al. « Efficient Computation of the Skyline Cube ». In : *VLDB*. 2005, p. 241-252 (cf. p. 75, 81, 83, 198, 204).

[YTH10] Chang YUN, Philip TREVINO et William HOLTKAMP. « PADS : Enhancing Gaming Experience Using Profile-Based Adaptive Difficulty System ». In : juill. 2010. DOI : 10.1145/1836135.1836140 (cf. p. 135).

[ZH02] Mohammed Javeed ZAKI et Ching-Jiu HSIAO. « CHARM : An Efficient Algorithm for Closed Itemset Mining ». In : *SDM*. Sous la dir. de Robert L. GROSSMAN, Jiawei HAN, Vipin KUMAR et al. SIAM, 2002. ISBN : 0-89871-517-2 (cf. p. 56).

[Zek17] Lougmiri ZEKRI. « A New Progressive Method for Computing Skyline Queries ». In : *J. Inf. Technol. Res.* 10.3 (2017), p. 1-21. DOI : 10.4018/JITR.2017070101. URL : https://doi.org/10.4018/JITR.2017070101 (cf. p. 204).

[Zer12] Ernst ZERMELO. « Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels ». In : *Proceedings of the Fifth International Congress of Mathematicians* II (jan. 1912) (cf. p. 139, 141).

Index

A
Agôn, 127
Alea, 127
Approche atomique, 128
Atome, 61
Atome hiérarchique, 163
Attribut dimension, 154
Autodétermination, 136

B
Bordure L, 30
Bordure U, 30
Bordures [L; U], 93
Boucle OCR, 136
Brique gameplay, 128

C
Classe d'équivalence d'un tuple, 191
Classement Elo, 255
Classification de Bartle, 125
Classification de Yee, 126
Clef cubique, 70
Co-atome, 61
Co-atome hiérarchique, 163
Concept Accord, 192
Concept Skyline, 195
Condition de non accord, 198
Conjonction de contraintes antimonotones, 66
Conjonction de contraintes hybrides, 66
Conjonction de contraintes monotones, 66
Connexion cubique, 69
Contrainte antimonotone, 65
Contrainte monotone, 65
CoSky, 222
Cube de données émergent, 90

D
DeepSky, 227

Difficulté, 135
Digramme de Hasse, 30
Dimension, 154
Domaine d'un niveau d'une dimension, 155
Domaine d'une dimension, 155
Dominance de Pareto, 76
Dominance sous condition de non accord, 198
Dp-idp, 219
Défi, 133

E

e-sport, 246
Elément maximal d'une hiérarchie, 155
Empreinte dimensionnelle, 155
Ensemble en accord, 107, 190
Ensemble en accord d'une relation, 190
Equilibre de Nash, 139
Espace convexe, 65
Espace multidimensionnel, 57
Espace multidimensionnel hiérarchique, 156

F

Fermeture cubique, 69, 110, 112
Fermeture cubique hiérarchique, 169
Fermeture d'une partition IDEA, 117
Flow, 132
Fonction Attribute, 60
Fonction Attribute dimensionnelle, 158
Fonction de mesure, 88
Fonction mesure compatible avec la fermeture cubique, 112
Forme normale d'un jeu, 139
Fun, 130

G

Gameplay, 130
Genre de jeux vidéo, 127
Généralisation, 59

H

Hiérarchie, 154
Hiérarchie de dominance, 220
Hiérarchie multiple, 153
Hiérarchie non-stricte, 153
Hiérarchie simple, 153
Hiérarchie stricte, 153

I
IDEA-successeur, 107
IDEA-successeur maximal, 107
Illinx, 127
Incertitude, 133
Individual value, 178
IV, 178

J
jeu, 122
jeu vidéo, 122
Join, 30
jouet, 122

L
L'opérateur de dominance $\succ$, 78
L-cube de données fermé émergent, 114
L-Skycube fermé émergent, 209
Ludus, 123, 127

M
Match making rating, 257
MDA, 123
Meet, 30
Meilleure réponse, 139
Mesure invariante, 207
Mimicry, 127
MinRatio, 90
Motivation extrinsèque, 136
Motivation intrinsèque, 136
Métabrique, 129

N
Niveau d'une hiérarchie, 155

O
Optimum de Pareto, 75
Opérateur Π-S, 195
Opérateur de fermeture, 31
Opérateur de fermeture cubique, 69, 110, 112
Opérateur de fermeture cubique hiérarchique, 169
Opérateur Différence, 61
Opérateur max, 59
Opérateur max dimensionnel, 158
Opérateur max généralisé sur les treillis cube, 159
Opérateur min, 59

Opérateur min dimensionnel, 158
Opérateur min généralisé sur le treillis cube, 159
Opérateur Produit, 60
Opérateur Produit dimensionnel, 160
Opérateur Produit généralisé sur le treillis cube, 161
Opérateur Semi-Produit, 60
Opérateur Semi-Produit dimensionnel, 161
Opérateur Semi-produit généralisé sur le treillis cube, 162
Opérateur Skyline, 78
Opérateur Somme, 59
Opérateur Somme dimensionnelle, 159
Opérateur Somme généralisé sur le treillis cube, 160
Opérateurs de fermeture de concepts Accords, 192
Ordre de géneralisation, 59
Ordre de spécialisation intradimensionnelle, 156
Ordre de spécialisation multidimensionnelle, 156
Ordre partiel, 29

P

Paidia, 123, 127
Partition d'un ensemble, 188
Partitions égales, 194
Plongement d'ordre, 62
Plongement d'ordre hiérarchique, 162
Produit de partitions, 189
Profondeur d'une hiérarchie, 155
Préférence Skyline, 78

R

Rang d'un tuple, 62, 163
Relation d'ordre, 29
Relation d'ordre entre partitions, 188
Relation de couverture, 30

S

SDT, 136
Skycube, 81
Skycube fermé émergent, 209
Skycube émergent, 205
Skyline, 79
Somme de partitions, 189
Sous-espace Skyline, 81
Spécialisation, 59
Stratégie dominante, 140
Structure d'une dimension, 154

Système de fermeture cubique, 70
Système de fermeture cubique hiérarchique, 171

T
Table de dimension, 155
TAGED, 231
Taille d'un treillis cube hiérarchique, 164
Taux d'emergence, 89
Taux d'emergence mesuré, 207
Technique anthropologiquement constitutive et constituante, 122
Test de fermeture, 117
Théorie des jeux, 139
Tier stratégique, 178
Top-k, 227
Treillis, 30
Treillis cube atomique, 61
Treillis cube co-atomique, 61, 162
Treillis cube de données, 62
Treillis cube de données hiérarchique, 163
Treillis cube fermé, 72
Treillis cube gradué, 62
Treillis cube hiérarchique atomique, 162
Treillis cube hiérarchique fermé, 171
Treillis cube hiérarchique gradué, 163
Treillis des concepts Accords, 192
Treillis des concepts Skylines, 196
Treillis des parties des attributs binaires, 62
Treillis des partitions, 189
Treillis hiérarchique des parties des attributs binaires, 162
Tuple dimensionnel, 155
Tuple fermé candidat, 117
Tuple fermé émergent, 112
Tuple émergent, 89, 205
Tuple émergent mesuré, 206
Tuple émergent non-fermé, 116
Typologie de la théorie des jeux, 127

V
Virtuosité, 135

A. Annexes

Sommaire

A. Réalisations autour des jeux 292
A.1. Genius 292
A.2. Unplugged 296
A.2.1. Likeadislike 299
A.2.2. Fantasy out there! 312
A.2.3. Iterative clues 317
A.2.4. Form Factor 325
A.2.5. Plastic Continent 339
A.2.6. Good ways 348
A.2.7. Dr Hanoï 358
A.2.8. Reroute better world 368

A. Réalisations autour des jeux

A.1. Genius

Genius est un projet de jeu vidéo sérieux [1] de l'Institut universitaire de technologie (IUT) de l'Université d'Aix-Marseille. Son objectif double est d'une part d'apporter une aide à l'orientation post-baccalauréat, notamment au sein des formations des IUT, et en particulier de celui de l'Université d'Aix-Marseille, qui dispose d'une offre de formations couvrant la plupart des secteurs d'activité. Son objectif est également d'autre part de proposer un outil d'accompagnement et de projection vers les métiers préparés par les IUT ainsi que leurs poursuites d'études.

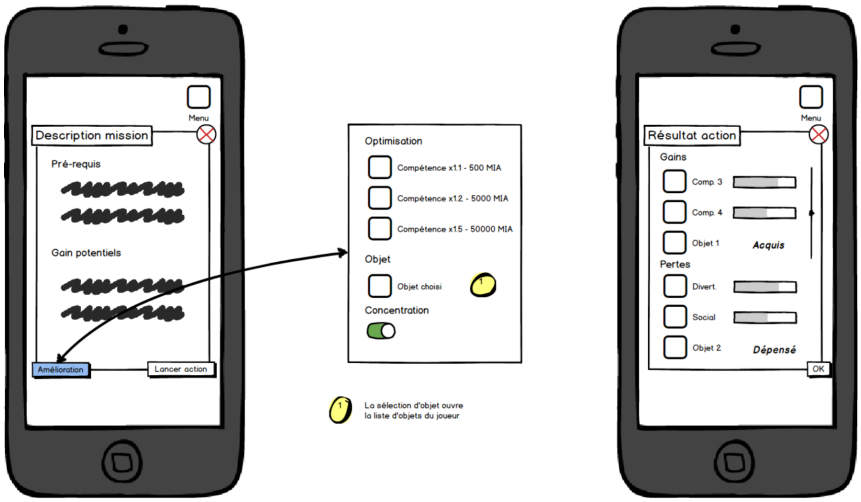


FIGURE A.1. – Genius : maquette

Le projet Genius est la suite du Test d'accompagnement métiers (TAM) [2] en y ajoutant une approche ludique et en incorporant l'approche par compétence incitée au niveau national. Partant du constat que ses formations préparent à un grand nombre de métiers, le TAM était un test à destination des étudiants qui leur permettait de se projeter sur des métiers correspondant à leur personnalité. A partir d'un questionnaire qui porte sur les goûts, les capacités et le caractère, les futurs étudiants peuvent amorcer une réflexion sur leur avenir professionnel et les parcours d'étude qui lui sont liés.

Genius est un jeu vidéo solo (mono-joueur) de gestion et d'aventure sous forme d'une application mobile disponible sur *smartphone* et tablette Android développée avec le moteur de jeu multi-plateforme Unity et disponible sur Google Play ou depuis son site Web vitrine.

1. Un jeu sérieux (ou *serious game*) est un dispositif qui combine une intention « sérieuse », de type pédagogique, informative, etc., avec des ressorts ludiques. De manière synthétique, un jeu vidéo sérieux englobe tous les jeux vidéo qui s'écartent du seul divertissement.
2. http://tufaisquoiapreslebac.fr/.

FIGURE A.2. – Genius : écran de chargement

FIGURE A.3. – Genius : écran de ville zonée en quartiers

L'histoire de Genius se déroule à Daedalus, une ville moderne découpée en quartiers thématiques où de grandes entreprises se sont implantées. Le joueur incarne un personnage agent de l'Orca, une agence de services secrets, menant des investigations et effectuant des missions de recherche des agents corrompus de l'Orca à l'aide d'un *smartphone* doté de l'intelligence artificielle (IA) Genius. Afin de mener à bien sans se faire repérer ses missions, allant de simples sujets de travaux dirigés ou pratiques à celles liées à des fonctions d'ampleur en passant par un éventail de tâches intermédiaires, et de faire toute la lumière sur les mystères de Genius, l'agent est amené à assimiler et améliorer des compétences lors de missions d'infiltration au sein des entreprises de Daedalus ainsi qu'en obtenant des diplômes à l'IUT au quartier universitaire.

FIGURE A.4. – Genius : tutoriel

Le cadre du projet Genius sert de prétexte aux joueurs de découvrir de manière ludique les compétences métier qui leur plaisent et les métiers qu'ils souhaitent faire plus tard, tout en faisant découvrir l'IUT en le définissant comme organisme de formation présent dans la plupart des domaines du monde du travail.

Compte tenu d'un nombre important de formations technologiques françaises de premier cycle universitaire, et de la bonne réception critique du précédent projet TAM porté par l'IUT de l'Université d'Aix-Marseille, l'offre d'une expérience mobile ludique dédiée à l'orientation des étudiants en études supérieures est apparue comme pertinente.

Outre la responsabilité de conception d'un jeu, c'est bien sa gestion et son encadrement qui ont été une grande source de motivation et d'enrichissement.

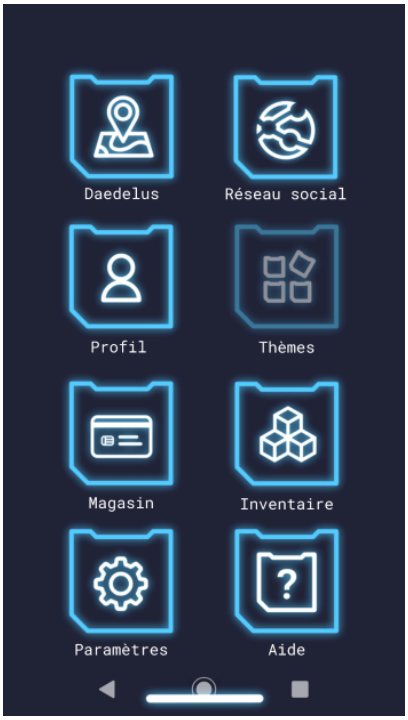


FIGURE A.5. – Genius : écran des options

FIGURE A.6. – Genius : mission

L'extraction des compétences métier s'est étalée sur plusieurs mois et s'est effectuée avec des entretiens et grâce à un formulaire de renseignement devant être complété par les responsables de formation. Au total, c'est 59 formations universitaires auditées sur les 64 que comptent l'organisme de formation, 5 compétences métier spécifique ou non par formation, et de 13 à 21 missions (et métiers correspondants) réparties en 4 rangs distincts. Au total, 272 compétences métier ont été répertoriées pour plus de 1000 missions et déclinaisons de métiers.

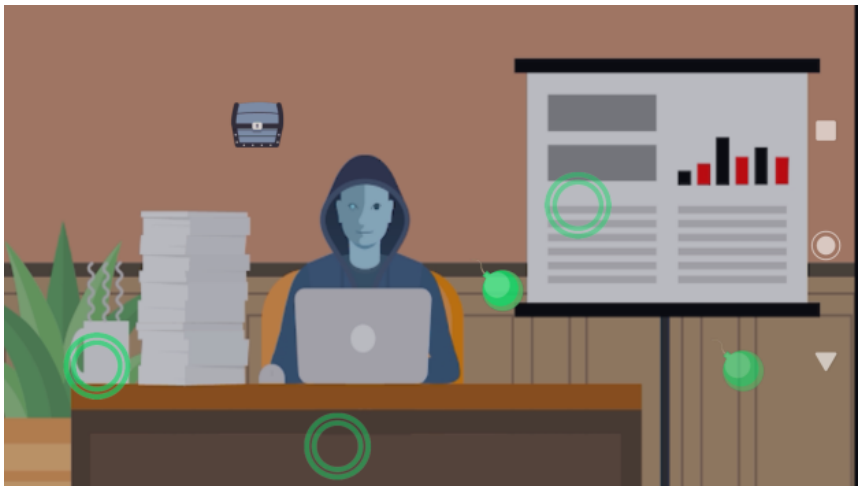


FIGURE A.7. – Genius : mini-jeu

Genius, bien que développé alors que la solution TAGED n'était pas encore achevée, est typiquement le type de projet de jeu vidéo sérieux qui pourrait en tirer profit tellement le nombre d'éléments à équilibrer, plus de 1200, est bien trop important pour être réalisable en étant dépourvu de dominance stratégique (cf. section 4.6.2.4 p. 140) sans assistance d'un outil ou bien d'une importante communauté de testeurs. Nous projetons un gain de plusieurs mois de travail homme si Genius avait été réalisé avec une implémentation de TAGED.

A.2. Unplugged

Unplugged [3] est une approche débranchée de la pensée computationnelle pour les enfants permettant de développer la créativité et la pensée critique.

Le projet Unplugged, coordonné par l'Université d'Aix-Marseille, a reçu le financement du programme ERASMUS+ de l'Union européenne sous la convention de subvention n°2020-1-FR01-KA227-SCH-095528 dans le cadre des appels extraordinaires lancés en 2020 pour soutenir la préparation à l'éducation numérique et les compétences créatives dans le cadre de la crise sanitaire COVID-19. Sa mise en œuvre réunie un consortium multidisciplinaire de professionnels, dont une majorité d'enseignants et de chercheurs, et se déroule au sein de cinq pays européens, la France,

3. https://www.unplugged-metaverse.eu/fr.

la Belgique, l'Italie, l'Autriche et la Bulgarie. Le projet Unplugged a été conçu pour renforcer les relations entre les écosystèmes d'apprentissage formel, comme les écoles, et informel, comme les musées, les bibliothèques, les associations, les familles…. Pour ce faire, les porteurs du projet sont à l'initiave de développements de pratiques créatives pour soutenir le positionnement des élèves en tant que penseurs critiques et citoyens actifs de la société du XXI^{e} siècle.

Autour d'une quête centrale, le projet s'articule comme un jeu unique complet composé de plusieurs défis ludiques, sous forme de jeux complets à part entière, réunis dans un univers original proposant un parcours cohérent dans le développement de comportements citoyens permettant aux enfants de reconnaître les valeurs nécessaires à la vie en communauté. L'ensemble des jeux Unplugged peut également être utilisé comme un guichet ouvert d'activité pouvant être appliquées à la fois en classe, à l'extérieur et à la maison, toutes liées à plusieurs objectifs pédagogiques clairement exprimés et constituant des objectifs de développement durable [4].

Dans le cadre de ce travail, nous sommes intervenu comme concepteur principal de jeux et auteur, et à ce titre, nous avons entièrement conçu 8 jeux et équilibré (cf. section 4.6 p. 137) 17 autres. Outre un public cible, enfant vivant en Europe de 6 à 12 ans, connu pour être délicat pour un concepteur de jeu, la prise en compte des objectifs de développement durable constituent, à eux seuls, un défi. Pour autant, la principale particularité de cette création est de rendre les dispositifs ludiques imprimables. Ainsi, idéalement, un jeu Unplugged doit être entièrement constructible par la famille d'un enfant disposant seulement d'un périphérique d'impression traditionnel, du manuel au plateau de jeu en pensant par les pions, les cartes…. De même, le cahier des charges donné pour consigne que les mécaniques, dynamiques et esthétiques de jeux (cf. section 4.3 p. 123) se doivent de permettre une distanciation sociale permanente sans en impacter négativement le *fun* (cf. section 4.5.2 p. 130). Ce travail a été mené à bien en parallèle de nos recherches afin, notamment, de tirer partie de l'expérience de conception de jeux et d'équilibrage sous fortes contraintes qu'il représente ainsi que pour le potentiel de synergies recélées entres les enseignants, les chercheurs et les membres du consortium, dont l'effort commun est à l'origine de plusieurs publications effectives ou escomptées, dans des domaines techniques, pédagogiques et sociaux. À l'échéance de la période d'exploitation d'Unplugged et de ses dispositifs ludiques, une implémentation de la solution TAGED pourra être utilisée pour générer des données décisionnelles permettant des choix améliorés de conception de jeux à la pédagogie et aux contraintes proches de ceux d'Unplugged.

Les jeux que nous avons créé sont listés succinctement ci-dessous, et sont accessibles en totalité, imprimables, et tels qu'ils sont mis à disposition du public, à la suite de cette section :

— Likeadislike : dépasser les idées préconçues en classe avec les élèves [5] ;
— Fantasy out there ! : développer l'imagination, l'ouverture d'esprit et la créati-

4. Les objectifs de développement durable (ODD, ou encore *sustainable development goals*, ou SDG en anglais) sont, au nombre de 17, les objectifs établis par les états membres des nations unies pour l'Agenda 2030, adopté par l'ONU en 2015.

5. https://www.unplugged-metaverse.eu/fr/likadislike

vité [6] ;
— Iterative clues : sensibilisation aux lois et règles liées aux nouvelles technologies [7] ;
— Form Factor : ; confronter les enfants au besoin d'inclusion [8] ;
— Plastic Continent : travailler sur les actions de l'homme sur la terre et les océans [9] ;
— Good ways : sensibiliser au climat et à la consommation responsable [10] ;
— Dr Hanoï : développer l'esprit logique avec un mini-jeu [11] ;
— Reroute better world : comprendre l'impact des actions personnelles sur les autres [12].

FIGURE A.8. – Jeux Unplugged conçus et développés [a]

a. De gauche à doite : Likeadislike, Fantasy out there !, Iterative clues, Form Factor, Plastic Continent, Good ways, Dr Hanoï, Reroute better world

6. https://www.unplugged-metaverse.eu/fr/fantasy
7. https://www.unplugged-metaverse.eu/fr/iterative-clues
8. https://www.unplugged-metaverse.eu/fr/form-factor
9. https://www.unplugged-metaverse.eu/fr/plastic-continent
10. https://www.unplugged-metaverse.eu/fr/good-ways
11. https://www.unplugged-metaverse.eu/fr/hanoi
12. https://www.unplugged-metaverse.eu/fr/reroute-better-world

Likadislike

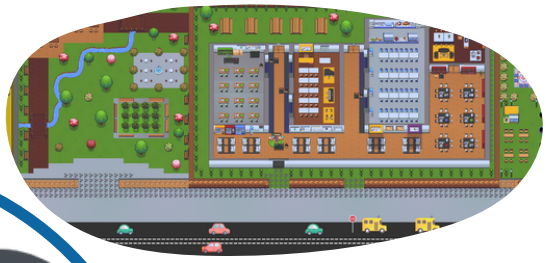
unplugged QUEST

Les joueurs piochent une carte décrivant une situation. Ensuite, chaque enfant exprime face cachée avec une carte emoji s'il aime, n'aime pas ou aime la situation décrite. Tous les joueurs retournent les cartes emoji en même temps et peuvent réagir sans jugement aux sentiments des autres.

Ressources imprimables fournies : cartes de situation x 45, cartes de réaction x 12, cartes de groupe (majority card x 1, minority card x 1, median card x 1)

Matériel additionnel nécessaire : paire de ciseaux

Bienvenue dans l'Univers Unplugged



Durée totale : 30 min.

Nombre de joueurs : 2 - 8+

ODDs liés

Objectifs d'apprentissage

- Mettre en mots des émotions et des sentiments
- Communiquer
- Apprendre à vivre en communauté
- Comprendre et s'exprimer à l'oral
- Discuter et remettre en question des choix moraux

Modalités de jeu

8 - 12 ans	En groupe
intérieur	Extérieur

Cette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Apprendre à se connaître et à mieux se connaître : offre un dispositif brise-glace favorisant le rapprochement des élèves en les confrontant à des situations de toutes sortes, amusantes, cocasses ou ennuyeuses, de leur quotidien. Le jeu permet aux enfants de partager librement des expériences personnelles, de découvrir que certains pensent comme eux et de comprendre d'autres points de vue. Il prépare les jeunes à partager leurs opinions en toute confiance et à savoir le faire de manière respectueuse. Il n'y a pas de carte "ni j'aime, ni je n'aime pas" car le jeu vise à ce que les enfants prennent réellement position.

Avoir moins peur des autres et de ce que pensent les autres : comme pour tout autre groupe sociétal, les élèves ne se connaissent pas toujours très bien. Ils peuvent avoir des idées préconçues, ce qui peut les amener à avoir peur des autres et de ce qu'ils pensent, et donc parfois aussi à les intimider ou à les rejeter. Les relations sociales peuvent permettre d'éviter cela et favoriser l'inclusion.

S'exprimer : donner des prétextes pour exprimer par des mots, à l'oral, ses émotions et ses sensations. Likedislike favorise l'expression des pensées et des sentiments et provoque des discussions sur les habitudes et les émotions qui peuvent souvent rester en dehors de la zone de confort des enfants. Il développe des compétences de communication saines chez les plus jeunes.

Développer sa sensibilité : étendre sa sensibilité par l'émotionnel et comprendre les règles de vie communes ainsi que respecter ce que pensent les autres.

Se poser des questions : encourage la curiosité et aide à adopter un comportement éthique et responsable.

Règles du jeu

But du jeu : Avec Likadislike, brisez la glace en pariant sur le groupe d'opinion auquel vous appartenez lors de mises en situations criantes de vérité ou drôle.

Rôle de l'enseignant et organisation du jeu : Nous appellerons les écoliers, **joueurs**, et l'adulte en charge des écoliers, **arbitre**.
Lorsque les joueurs jouent à tour de rôle, c'est l'arbitre qui décide de l'ordre.

Installation : Mélangez le paquet de **cartes situations** et placez-le au centre des joueurs. Donnez à chaque joueur une **carte de réaction** de chaque type : **je n'aime pas,** **j'aime** **et** **j'adore.**

Disposez les cartes de groupes : **majorité**, **médiane** et **minorité** au centre des joueurs.

Chaque tour de jeu est constitué de trois phases consécutives :

- phase du joueur **actif**
- phase de **réaction**
- phase de **révélation**

Une séance de jeu s'arrête lorsque tous les joueurs ont été joueur actif le même nombre de fois. **Ceux qui ont le plus de points de victoire (cartes situations) l'emportent !**

Phases de jeu

Phase 1 - Phase du joueur actif

À chaque nouvelle phase de situation, le joueur actif change. Tous les joueurs seront, à tour de rôle, joueur actif.
Le joueur actif effectue les actions suivantes en ne dévoilant aucune des cartes :

1. piocher **une carte situation** et en prend connaissance seul
2. choisir **sa carte réaction** décrivant au mieux son ressenti réel face à la situation décrite
3. choisir **la carte de groupe** correspondante, s'il pense (*) qu'il fera partie de la majorité des joueurs à réagir ainsi (carte majorité), de la minorité (carte minorité), ou aucune des deux (carte médiane)
4. Décidez si le vote portera sur le "sentiment à l'égard" ou sur le "comportement à adopter dans" la situation.

*Pour tenter de parier juste, il est conseillé, notamment, de prendre en compte la situation décrite et les autres joueurs.

Phase 2 - Phase de réaction

Au début de la phase de réactions, le joueur actif révèle, uniquement, la **carte situation** piochée à la phase précédente.
Tous les autres joueurs prennent connaissance de la situation, et, à leur tour, choisissent, et réservent face cachée, leur carte réaction décrivant au mieux leur ressenti réel face à la situation décrite.

Phase 3 - Phase de révélation

Durant la phase de comptabilisation, tous les joueurs retournent ensuite leurs cartes de réaction en même temps.
Les joueurs sont alors libres de débattre calmement, respectueusement et sans jugement, des réactions de chacun.
L'arbitre aide ensuite les joueurs à comptabiliser le nombre de réactions de chaque type pour déterminer si le joueur actif remporte son pari. Si oui, il récupère la carte situation en tant que point de victoire.

Sujet 1 - Exprimer une opinion

Pour développer davantage une communication saine et l'expression d'opinions, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Brainstorming, The Perfect City, Live in Harmony, etc.***

Pour en savoir plus sur le raisonnement logique, vous pouvez consulter le site suivant (en) : https://www.rolemodels.me/news/express-opinions-confidently

Sujet 2 - Exprimer ses sentiments

Pour favoriser l'expression des sentiments, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Clay and Sculpture, Poverty-free game collection, Live in Harmony, etc.***

Pour en savoir plus sur la manière de développer l'intelligence émotionnelle, vous pouvez consulter les sites suivants (en) https://ideas.ted.com/how-to-raise-emotionally-intelligent-kids/ ou https://www.understood.org/en/articles/the-importance-of-emotional-intelligence-for-kids-with-learning-and-thinking-differences

Sujet 3 - Inclusivité

Pour aider les enfants à développer leurs sentiments envers vous, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Poverty-free game collection, Live in Harmony, Cookies and Peace, Equal, etc.***

Pour avoir un regard global sur l'inclusion, vous pouvez vous référer à (en) https://www.twinkl.bg/teaching-wiki/inclusion

Imprimer les déclarations

Mâcher les stylos qu'on me prête	Me mettre toujours au premier rang	Emprunter les affaires des autres mais oublier de les rendre
Se mettre toujours au fond de la classe près du radiateur	Amener toujours des goûters à la récréation	Être tout le temps sur les réseaux sociaux à la récréation
Être toujours à jour de la dernière tendance	Dire que je n'ai pas révisé quand j'ai une bonne note	Aller souvent aux toilettes
Avoir tout le temps envie de faire la sieste	Avoir toujours faim	Avoir un nouveau cartable à chaque rentrée
Avoir toujours une nouvelle idée d'activité à faire à la récréation	Regarder toujours ce que font les autres	Se mettre toujours seul dans le bus

Imprimer les déclarations

Raconter tout le temps ma vie	Raconter toujours les aventures extraordinaires qui me seraient arrivées	Faire profiter de notre belle voix tout le monde
Être rassuré d'avoir toutes mes affaires dans mon cartable	Faire des acrobaties avec mes doigts	Faire des bonjours qui n'en finissent pas
Ne rien retenir du premier coup	Sculpter des gommes	Reprendre ceux qui font des fautes de langue
Rentrer à la maison avec des vêtements qui ne m'appartiennent pas	Toujours envie d'aider les autres	Tenter tout le temps d'esquiver les punitions
Reprendre la question ou la réponse qui vient d'être donnée	Renifler au lieu de se moucher	Passer au tableau

Imprimer les déclarations

Regarder des films en classe	Ne pas être invité à jouer à la récréation	Les sorties à la piscine
Les élections de délégués	La nourriture de la cantine	Connaître les tables de multiplication
Avoir toujours un commentaire à faire	La rentrée des classes	Faire ses devoirs
Les problèmes de maths	Les dictées	Les jours où il n'y a pas école
Les problèmes de maths	Les dictées	Les jours où il n'y a pas école

Imprimer les emojis

Reproduire autant que nécessaire en fonction du nombre d'élèves.

Imprimer les emojis

Reproduire autant que nécessaire en fonction du nombre d'élèves.

Imprimer les emojis

unplugged QUEST

Reproduire autant que nécessaire en fonction du nombre d'élèves.

Cartes groupe

unplugged QUEST

je pense que ni une majorité ni une minorité de joueurs réagirait comme moi

je pense qu'une minorité de joueurs réagirait comme moi

je pense qu'une majorité de joueurs réagirait comme moi

Fantasy out there!

A.2.2

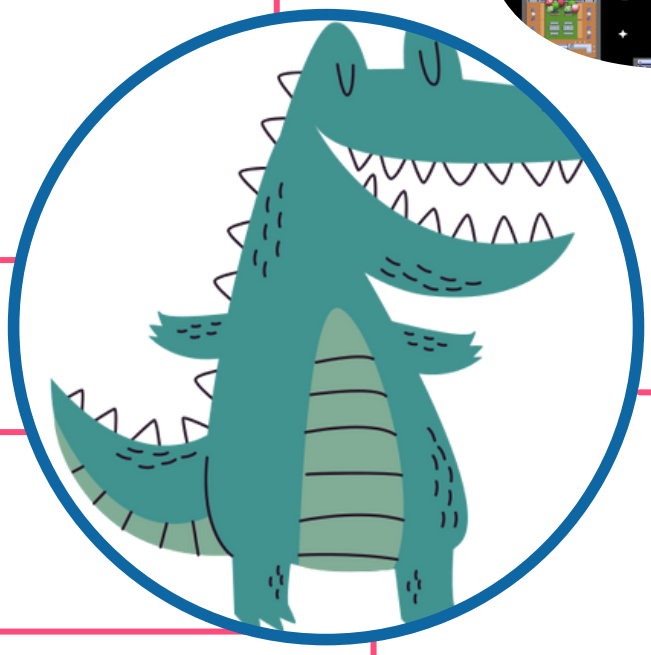
unplugged QUEST

Par la fenêtre de la classe, vous pouvez voir le ciel. Mais si vous accrochez la feuille plastique sur laquelle vous avez dessiné des dinosaures, des superhéros ou des OVNIs, cela ressemblera à une époque perdue, une bande dessinée devenue réelle ou une invasion extraterrestre ! Transformez la fenêtre de la classe en vitre de Jeep de Jurassic Park, une vue depuis la Tour Stark ou un niveau de Space Invaders !

Ressources imprimables fournies : aucune.

Matériel additionnel nécessaire: feuille plastique transparente, ruban adhésif transparent, peinture glycédo (à l'huile) sous forme de feutre (façon Posca) ou avec un pinceau, paire de ciseaux, montre

Territoire 4 - L'usine fantastique



Durée totale : 60 min.

Nombre de joueurs : 2 - 12

ODDs liés

Objectifs d'apprentissage

- Imaginer et réaliser
- Être sensible à l'art
- Comprendre les relations entre objet et espace
- Mettre en œuvre un projet artistique
- Développer un esprit d'autonomie, de coopération et de responsabilité dans la coopération et le partage

Modalités de jeu

8 - 12 ans	intérieur
En classe	En groupe

Cette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Histoire à créer et à retenir : ensemble, les élèves créent et donne corps à un conte fantastique développant leur créativité, leur imaginaire, leur mémorisation et leur coopération. La narration est un outil efficace pour l'apprentissage et le développement des enfants. Selon des recherches récentes, les contes suscitent l'empathie chez les enfants, favorisent le développement de leur cerveau et jettent les bases du développement de compétences sociales, communicationnelles et interpersonnelles. En les introduisant dans le monde des contes, on encourage leur pensée créative et leur capacité à résoudre des problèmes. Elle laisse place à leur imagination, développe le langage et les émotions, et renforce les relations. Le jeu Fantasy out there! se joue en équipe, où les enfants ont l'occasion de coopérer et de créer ensemble des histoires fantastiques. Ce jeu développe simultanément leurs capacités de travail en équipe et de communication et les initie au processus créatif.

Décoration esthétique : apporte un intérêt décoratif de la salle de classe, visible par les vitres, de dedans et de dehors. Fantasy out there! contribue à l'éducation des enfants au service de la communauté en s'impliquant dans des activités telles que la décoration de l'environnement scolaire. Elle démontre aux enfants le processus artistique et développe leur sens esthétique en matière de design.

Comprendre les problèmes liés à la création d'une attraction, jouer avec les perspectives et faire de la réalité augmentée déconnectée : fait travailler l'imaginaire mais aussi la logique et permet de créer des scènes et des photographies originales. La réalité augmentée est une caractéristique indispensable du monde numérique actuel et de la façon dont la société moderne interprète l'information. La réalité augmentée est une expérience dans laquelle les concepteurs améliorent certaines parties du monde physique des utilisateurs à l'aide de données générées par ordinateur. La réalité augmentée soutient l'apprentissage des enfants en démontrant les caractéristiques des objets étudiés dans des conditions presque réelles, par exemple les événements naturels, l'univers, etc. Il s'agit d'un parcours professionnel attrayant. Dans Fantasy out there!, les enfants reçoivent leur première impression de ce que peut être la réalité augmentée et de son impact sur les autres.

Règles du jeu

Récit du jeu : Le but de ce jeu est de permettre aux enfants de faire ce qu'ils ne sont pas censés faire : dessiner sur le verre de la classe ! Fantasy out There! est un jeu coopératif où les joueurs dessinent et créent une histoire ensemble.

Rôle de l'enseignant et organisation du jeu : Nous appellerons les enfants, les joueurs, et l'adulte responsable, un maitre de jeu.
Pendant que les joueurs se relaient, c'est le maître de jeu qui décide de l'ordre des tours et de la durée des phases de jeu.

Règles du jeu : Chaque tour est composé de trois phases consécutives :

- phase de création artistique
- phase de composition
- phase de conte

Une partie s'arrête au bout d'une heure maximum, ou lorsque tous les joueurs ont été conteurs au moins une fois durant la phase de conte. Le jeu pouvait aussi se jouer pendant plusieurs jours, semaines et même mois, reprenant les contes là où ils s'étaient arrêtés la dernière fois.

Phases du jeu

Phase 1 - Création artistique

Durant la phase de création artistique, tous les joueurs dessinent, peignent, élaborent comme bon leur semble, en même temps, tout ce qu'ils souhaitent sur une feuille transparente. Cette phase dure un temps connu et fixé à l'avance,

Phase 2 - Composition

Durant la phase de composition, tous les joueurs doivent ajouter, à tour de rôle, leur réalisation à la scène crée par l'ensemble des créations, en la collant suffisamment proche ce celles des autres sur les vitres de la salle de classe. Cela peut être fait avec du ruban adhésif double face par exemple.

S'il est parfois possible pour un joueur de garder l'idée de départ qu'il avait en tête lors de la réalisation de sa création, il lui faut le plus souvent mener une réflexion sur la manière de combiner et remanier sa proposition avec celles des autres pour faire évoluer la scène qu'elles composent ensemble.

Les feuilles peuvent être superposées à loisir, leur transparence permettant de changer la précédente proposition d'une scène. Les feuilles peuvent même être découpées en plusieurs parties avant leur intégration à la scène pour encore mieux coller à la volonté du joueur.

Phase 3 - Récit

Durant la phase de récit, tous les joueurs deviennent, à tour de rôle, dans l'ordre de la phase de composition, le conteur.

Ainsi, le conte prend vie petit à petit, il est répété du début à chaque fois, par chaque nouveau conteur, qui y ajoute sa contribution à la fin en rapport avec sa création et sa composition. Cette contribution prend la forme d'un mot ou d'une courte phrase de quelques mots seulement.

Si un joueur n'arrive pas à se rappeler de tout le conte, il peut y être aidé par les autres joueurs ou le maître du jeu.

Sujet 1 - Compétences en matière de narration

Pour développer davantage les compétences en matière de narration, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Brain Twister, The Perfect City, Pop-up City in the Future, etc.***

Pour en savoir plus sur l'importance du développement des compétences en matière de narration, vous pouvez consulter la ressource suivante (en) : https://www.playskipy.com/blog/art-of-storytelling-for-kids/.

Sujet 2 - Sens esthétique

Pour développer davantage le sens esthétique et la créativité, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***The Perfect City, Pop-up City in the Future, etc.***

Pour en savoir plus sur l'importance du développement du sens esthétique, vous pouvez consulter (en) :

- https://bit.ly/3LWBsmq
- https://www.scirp.org/html/2-6302236_51211.htm

Sujet 3 - Réalité augmentée

Pour en savoir plus sur les avantages de l'utilisation de la réalité augmentée dans l'apprentissage, vous pouvez consulter le site suivant (en) : https://saferkidsonline.eset.com/uk/article/get-to-know-augmented-reality

Iterative clues

unplugged QUEST

Découvrez le monde d'aujourd'hui, le monde de demain et les nouvelles technologies tout en vous amusant grâce à Iterative Clues ! La quête améliore les connaissances des enfants sur les nouvelles technologies et le monde du digital, en introduisant l'association d'idée pour stimuler la mémorisation ainsi que le lien avec la vie réelle.

Ressources imprimables fournies : cartes concepts x 30, point x 74 (1 point x 64, 5 points x 10)

Matériel additionnel nécessaire : paire de ciseaux
Ardoises (optionnel)

Territoire 4 - L'usine fantastique

Durée totale : 30 min.

Nombre de joueurs : 2 - 12+

ODDs liés

Objectifs d'apprentissage

- Comprendre et s'exprimer oralement
- Rechercher
- Lire
- Raisonner
- S'informer dans le monde numérique et mobiliser les outils numériques

Modalités de jeu

8 - 12 ans	Soutien d'un adulte
intérieur	Faisable à la maison
En classe	Travailler seul

Cette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Rester informé : en apprendre plus sur les nouvelles technologies et le monde numérique. Le jeu introduit de manière ludique des concepts et des définitions d'outils numérique. En jouant avec des associations d'idées, les élèves se familiarisent avec les composants numériques de base et leur interrelation.

Faire preuve de rapidité : répondre le premier pour l'emporter. L'élément compétitif du jeu prépare les enfants à être performants dans un environnement dynamique.

Associer les idées : combiner les indices entre eux pour trouver le concept caché. La pensée associative et logique est développée par la série d'indices divulgués aux enfants.

Règles du jeu

But du jeu

Comme son nom l'indique, avec Iterative clues, soyez le premier à trouvez le concept caché par une suite d'indice de plus en plus précis.

Nous appellerons les écoliers, **joueurs**, et l'adulte en charge des écoliers, **arbitre**.

Installation

Mélangez le **paquet de cartes** concepts et placez-le au centre des joueurs.
Disposez les **points**, 1 point et 5 points, au centre des joueurs.

1 point

5 points

L'arbitre est responsable des cartes concepts et des points.

Étapes de jeu

Déroulement d'un tour de jeu

L'arbitre **tire la carte concept du** dessus de la pioche, et tout en la gardant secrète, énonce l'indice n°1, noté 1 en haut de la carte. Tous les joueurs ont le droit de **répondre en même temps, mais ne peuvent donner qu'une seule réponse.**

Si le concept est trouvé, **l'arbitre garde toujours la carte secrète, le tour de jeu prend fin, et le joueur ayant trouvé le concept tente un défi.** Sinon**, l'indice n°2, noté 2 en haut de la carte,** est lu, et ainsi de suite jusqu'à l'indice n°4.

Si aucun joueur n'a trouvé le concept après que l'indice n°4 ait été donné, l'arbitre énonce la réponse et commence un nouveau tour de jeu.

Défis

Le joueur ayant trouvé le concept peut tenter un des deux défis suivant de son choix :

- **épeler correctement le concept trouvé**
- **citer les deux précédents concepts (ce défi ne peut être choisi qu'à partir du troisième tours)**

Si le joueur échoue au défi qu'il a choisi, il n'a pas le droit d'essayer l'autre.

Comptabilisation des points et fin de partie

Le premier joueur a donner la bonne réponse gagne (5 - niveau du dernier indice) points, soit de 1 à 4 points comme ceci :

- concept trouvé avec l'indice n°1 : **4 points**
- concept trouvé avec l'indice n°2 : **3 points**
- concept trouvé avec l'indice n°3 : **2 points**
- concept trouvé avec l'indice n°4 : **1 point**
- S'il a réussi le défi, le joueur gagne **1 point de plus**.

N'importe quel joueur peut, à tout moment, demander à l'arbitre de faire la monnaie de ses points.

Le premier joueur à obtenir 15 points ou plus remporte la partie.

Étapes de jeu

Régle alternative : pour animer le jeu dans une classe en gardant le calme et l'attention des enfants.

Les enfants sont répartis par groupe (ou tout seul si peu nombreux). Chaque groupe commence avec **6 pièces d'or** et chaque groupe a une ardoise.

Le meneur de jeu écrit le premier indice au tableau. Les élèves se concertent en groupe et choisissent de jouer ou non. S'ils jouent ils peuvent soumettre un mot en l'écrivant sur l'ardoise. Cette action leur coutera une pièce d'or :

- Si le concept est trouvé, chaque groupe qui a trouvé le bon concept reçoit **quatre pièces d'or supplémentaire**. Le tour s'arrête et le meneur de jeu choisit un nouveau mot.
- Si le concept n'est pas trouvé, le meneur de jeu inscrit l'indice suivant au tableau et ainsi de suite jusqu'à ce que l'un ou plusieurs groupes trouvent le concept ou que les quatres indices soient énoncés.

Dans cette version, le nombre d'indices donnés n'impacte pas le nombre de points gagnés qui **est toujours 4.**

Si un groupe se retrouve sans pièces d'or à la fin d'un tour, le meneur du jeu le renfloue de deux pièces d'or pour ne pas pénaliser l'enthousiasme et la participation.

A la fin du jeu, le groupe avec le nombre le plus élevé de points l'emporte.

Sujet 1 - Sensibilisation à la numérisation

Pour aller plus loin dans l'apprentissage des faits et des termes relatifs au monde numérique, vous pouvez vous rendre dans d'autres quêtes Unplugged, parmi lesquelles : ***Memory, etc***.

En outre, n'hésitez pas à consulter un document de recherche de l'OCDE sur l'éducation à l'ère numérique pour des enfants heureux et en bonne santé (en) : https://www.oecd.org/education/education-in-the-digital-age-1209166a-en.htm

Sujet 2 - Compétitivité et rapidité de réaction

Pour aller plus loin dans la rapidité de réponse et la compétitivité, vous pouvez vous référer aux autres quêtes Unplugged, parmi lesquelles : ***Brain Twister, Plastic Continent, Form factor, etc.***

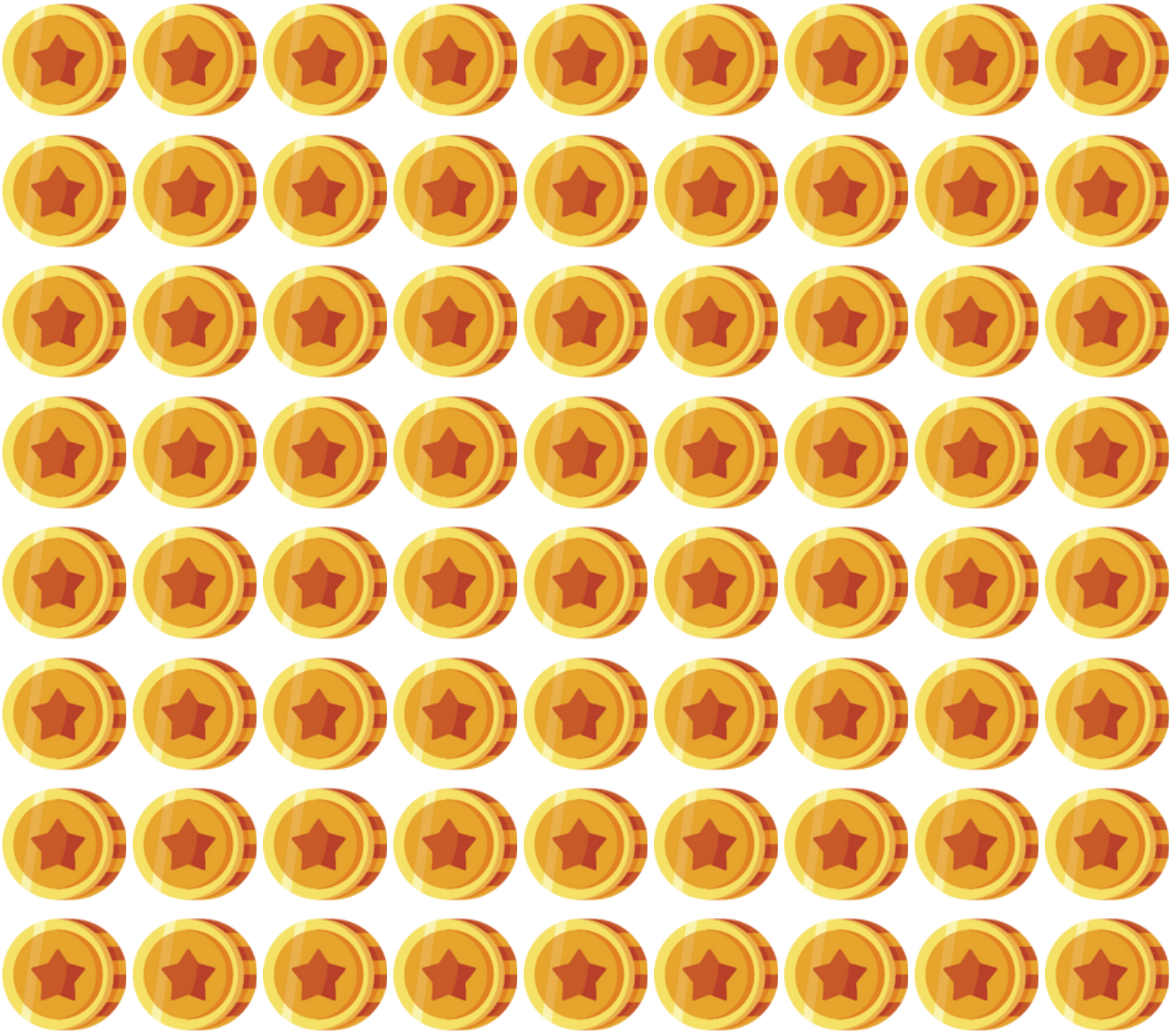
Sujet 3 - La pensée associative

Pour en savoir plus sur les avantages du développement de la pensée associative et de la créativité chez les élèves, reportez-vous à (en) : https://www.psychologytoday.com/us/blog/prime-your-gray-cells/201802/study-reveals-thought-processes-foster-creativity

Pour développer davantage la pensée associative, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Brain Twister, Fantasy out there !***

Matériaux

- **Points & Pièces**



Cartes à imprimer

Matériaux – Cartes à imprimer

1. Encre 2. Feuille 3. Copie 4. Périphérique **IMPRIMANTE**	1. Écran tactile 2. Applications 3. Poche 4. Téléphone **SMARTPHONE**	1. Outil numérique 2. Multimédia 3. Tactile 4. Applications **TABLETTE**
1. Logiciel 2. Téléchargement 3. Store 4. Smartphone **APPLICATION MOBILE**	1. World Wide Web 2. Site web 3. Domaines 4. Wifi **INTERNET**	1. Virus 2. Cyber attaque 3. Anonymous 4. Ordinateur **HACKEUR**
1. Metaverse 2. Virtuel 3. Serveur 4. Univers **MONDE PERSISTANT**	1. Messagerie directe 2. Amis 3. Stories 4. Likes **RÉSEAUX SOCIAUX**	1. Personnalité 2. Followers 3. Communauté 4. Placement de produit **INFLUENCEUR**
1. E-sports 2. Twitch 3. Temps réel 4. Live **STREAMING**	1. Films et séries 2. Abonnement 3. Catalogue 4. Streaming **VIDÉO À LA DEMANDE**	1. 10eme Art 2. Virtuel 3. Amusant 4. Manette **JEUX VIDEO**
1. Immersion 2. Simulation 3. 3D 4. Casque **RÉALITÉ VIRTUELLE**	1. Écran 2. Effet spéciaux 3. Réalité 4. En direct **RÉALITÉ AUGMENTÉE**	1. Véhicule 2. Électricité 3. Deux roues 4. Freins **TROTINETTE ÉLECTRIQUE**

Cartes à imprimer

1. Vol 2. Transport 3. Propulsion 4. Sac à dos **JETPACK**	1. Ciel 2. Télécommandé 3. Vidéo 4. Quatres hélices **DRONE**	1. Connexion 2. Accessoire 3. Réseau local 4. Sans fil **BLUETOOTH**
1. Connexion 2. Ordinateur 3. Réseau 4. Sans-fil **WIFI**	1. Internet 2. Réseau 3. Cable 4. Lumière **FIBRE OPTIQUE**	1. Touches 2. Azerty 3. Écrire 4. Ctrl **CLAVIER**
1. Carré 2. Code barre 3. Scan 4. Information **QR CODE**	1. Smartphone 2. Accessoire 3. Heure 4. Poignet **MONTRE CONNECTÉE**	1. FabLab 2. Plastique 3. Fondu 4. Objet **IMPRIMANTE 3D**
1. Téléphone 2. Accessoire 3. Bluetooth 4. Musique **ECOUTEURS SANS FIL**	1. Carré 2. Résolution 3. Écran 4. Px **PIXEL**	1. Découpe 2. Rayons 3. Sabre 4. Lumière **LASER**
1. Robot 2. Cerveau 3. Algorithme 4. AI **INTELLIGENCE ARTIFICIELLE**	1. Miner 2. Coin 3. Paiement 4. Monnaie **CRYPTO MONNAIES**	1. Crypto monnaies 2. Token 3. Non-Fongible 4. Art **NFT**

Form factor

unplugged QUEST

Form Factor est un jeu de modélisation de formes complexes avec des formes de base qui met l'accent sur la communication et la coopération avec les équipes. Le but est d'essayer de remplir au mieux une forme complexe sans dépasser les jantes avec les formes simples dont nous disposons. Un vote collectif à la fin décide des gagnants.

Ressources imprimables fournies : formes complexes x 8, formes simples x 60.

Matériel additionnel nécessaire : paire de ciseaux, montre

Territoire 1 - L'île pacifique

Durée totale : 15 min.

Nombre de joueurs : 4 équipes

ODDs liés

Objectifs d'apprentissage

Modéliser

Comprendre les représentations du monde et de l'activité humaine

Représenter

Comprendre les relations entre objet et espace

Modalités de jeu

6 - 8 ans | Intérieur

En classe | En groupe

TCette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Développer sa capacité d'analyse : reconnaître, projeter mentalement, et faire correspondre, des formes et des motifs. Les compétences informatiques et la pensée critique reposent toutes deux sur le développement de la capacité à analyser les informations. Form factor renforce les capacités d'analyse et d'abstraction, en permettant aux enfants d'élaborer des stratégies pour créer une figure.

Pensée visuelle et géométrique : La mémoire visuelle est améliorée en encourageant la pensée visuelle/spatiale des enfants. Elle prépare les enfants à traiter des connaissances diverses, en favorisant un traitement efficace des informations complexes ou potentiellement déroutantes.

Augmenter sa précision : être à l'aise en précision et en dextérité, ou "motricité fine", des doigts. Le terme "motricité fine" fait référence aux petits muscles du corps, tels que les doigts, les mains, la langue, les orteils et les yeux. La motricité fine est la capacité à contrôler ces muscles fins - en particulier dans les mains et les yeux - pour accomplir des mouvements et des tâches. Au cours de la petite enfance, les enfants doivent se développer de manière holistique, ce qui inclut le développement physique, intellectuel (cognitif), social et émotionnel. Le développement de la motricité fine et de la motricité globale est important, car la réussite scolaire dépend de la bonne maîtrise de ses muscles par l'enfant. Il a besoin d'un tronc solide et d'une bonne posture pour s'asseoir à un bureau sans se fatiguer, de muscles oculaires capables de suivre la lecture, d'une bonne prise du crayon et d'un bon contrôle des doigts pour former les lettres lorsqu'il écrit, et bien d'autres choses encore. En essayant d'adapter les formes simples proposées dans Form factor à une figure complexe, les enfants développent leur motricité fine et leur précision, tout en apprenant à être rapides.

Augmenter sa vitesse : être plus rapide en réflexion et en exécution. Form factor est un jeu compétitif et suggère du temps dans sa réalisation au sein des équipes. L'amélioration de la vitesse dans les tâches effectuées augmente l'efficacité de l'allocation des ressources attentionnelles chez les jeunes adultes. La vitesse de traitement est l'un des principaux éléments du processus cognitif, ce qui explique qu'elle soit l'une des aptitudes les plus importantes dans l'apprentissage, les performances scolaires, le développement intellectuel, le raisonnement et l'expérience. La vitesse de traitement est une capacité cognitive qui peut être définie comme le temps qu'il faut à une personne pour effectuer une tâche mentale. Elle est liée à la vitesse à laquelle une personne peut comprendre et réagir aux informations qu'elle reçoit, qu'elles soient visuelles (lettres et chiffres), auditives (langage) ou gestuelles. En d'autres termes, la vitesse de traitement est le temps qui s'écoule entre la réception et la réaction à un stimulus. En outre, la plupart des enfants n'ont pas les capacités cognitives nécessaires pour organiser leur emploi du temps de manière autonome avant le collège. L'amélioration de la vitesse de traitement est étroitement liée à l'adoption précoce de compétences en matière de gestion du temps. Apprendre aux enfants à planifier et à hiérarchiser leur temps à un stade précoce les préparera à être plus efficaces dans leur développement professionnel et à mieux profiter de leurs activités de loisirs.

Coopérer : travailler de concert avec son binôme ou son équipe pour faire la plus belle œuvre ! Le travail en équipe dans le cadre dans Form factor présente aux enfants les opportunités et les défis de la coopération - exprimer des idées, accepter et rejeter de manière justifiée les autres idées, et travailler sous une contrainte de temps.

Règles du jeu

But du jeu

Le but de ce jeu est de compléter une forme complexe avec des formes simples plus petites de la manière le plus créative qui soit !

Form factor est un jeu coopératif dans lequel les joueurs réalisent le plus beau remplissage ensemble.

Rôle de l'enseignant et organisation du jeu

Nous appellerons les écoliers, joueurs, et l'adulte en charge des écoliers, arbitre.

C'est l'arbitre qui décide du temps de jeu.

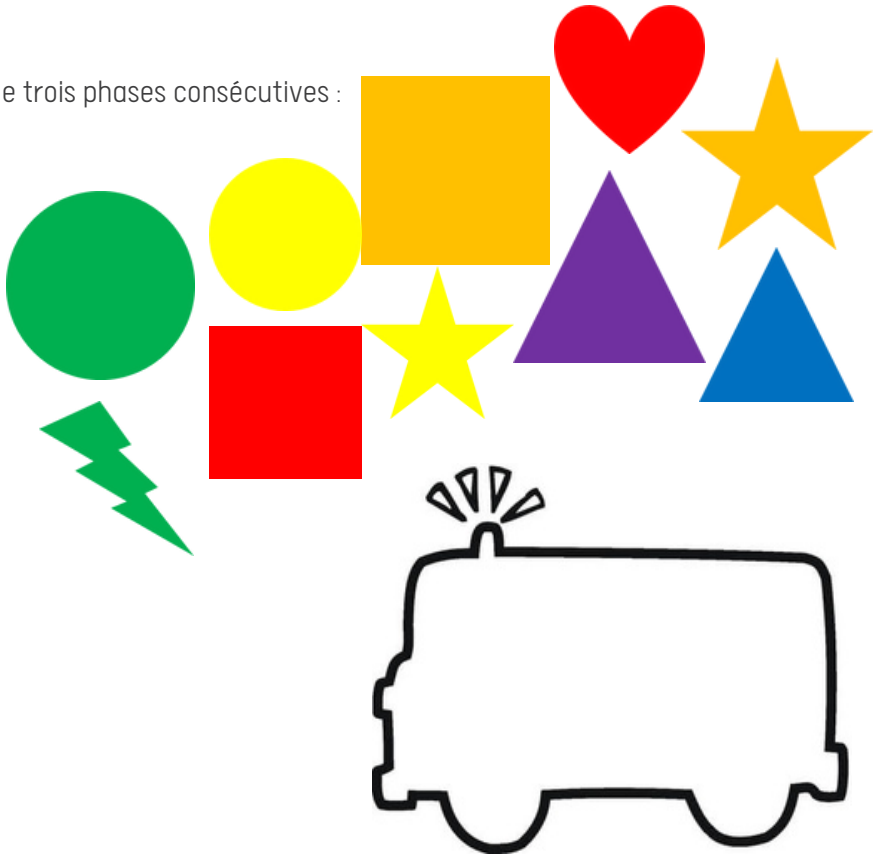
Installation :

Disposez les formes simples au centre des joueurs.

L'arbitre choisi une forme complexe à réaliser,
éventuellement après avoir consulté l'avis des joueurs sur le sujet.

Règles du jeu

- Chaque tour de jeu est constitué de trois phases consécutives :
- phase de constitution d'équipes
- phase de création artistique
- phase de concours



Phases de jeu

Phase 1 - Constitution d'équipes

Durant la phase de constitution d'équipes, les joueurs forment, avec l'aide de l'arbitre, au plus **4 équipes, avec si possible** un nombre équivalent de joueurs dans chaque équipe.
L'arbitre remet ensuite, à chaque équipe, **la forme complexe qu'il a choisie** (la même pour chaque équipe).

Phase 2 - Création artistique

Durant la phase de création artistique, tous les joueurs de toutes les équipes **remplissent, en même temps, la forme complexe de leur page.** Les joueurs sont libres **d'élaborer le remplissage de leur choix**, du moment qu'ils le font uniquement en superposant la forme complexe avec les formes simples mises à leur disposition. Ils peuvent pour cela laisser libre court à leur créativité et, notamment, **empiler autant de formes simples qu'ils le souhaitent afin de créer de nouveaux motifs de formes et de couleurs**. Cette phase dure un temps connu et fixé à l'avance,

Phase 3 - Concours

Durant la phase de concours**, un vote collectif est organisé par l'arbitre.** Chaque joueur vote en son âme et conscience pour l'équipe qui a selon lui :

- le mieux rempli la forme complexe en **dépassant le moins possible**
- fait le remplissage le plus **créatif** et le plus **artistique**

Les votes sont individuels. Les équipes n'existent plus à la phase de concours que pour éventuellement faire emporter les joueurs qui compose l'équipe victorieuse.

Sujet 1 - Compétences analytiques et visuelles

Pour entraîner d'autres enfants à développer leur pensée analytique et visuelle, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels: ***Memory, Good ways, Farm in the City, Plastic continent, etc.***

Pour en savoir plus sur l'importance d'améliorer les capacités de réflexion analytique et visuelle, vous pouvez consulter le site suivant (en) : http://www.xplaner.com/visual-thinking-school/

Sujet 2 - Dextérité et motricité fine

Pour entraîner d'autres enfants à développer leur dextérité et leur motricité fine, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Good ways, Pop-up city in the Future, Dr. Hanoi, etc.***

Pour en savoir plus sur l'importance du développement de la motricité fine, vous pouvez consulter les sites suivants (en) : https://empoweredparents.co/why-are-fine-motor-skills-important/

Sujet 3 - Vitesse de traitement et coopération

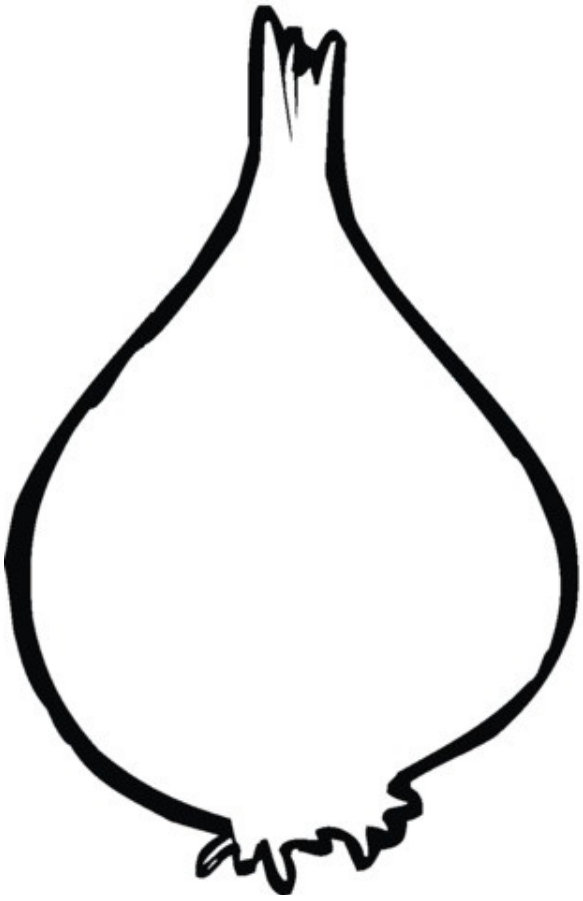
Pour entraîner d'autres enfants à développer leur vitesse de traitement et leur coopération, vous pouvez vous référer à d'autres jeux Unplugged, dont les suivants : ***Brainstorming, Fantasy out there!, Poverty-free game collection, etc.***

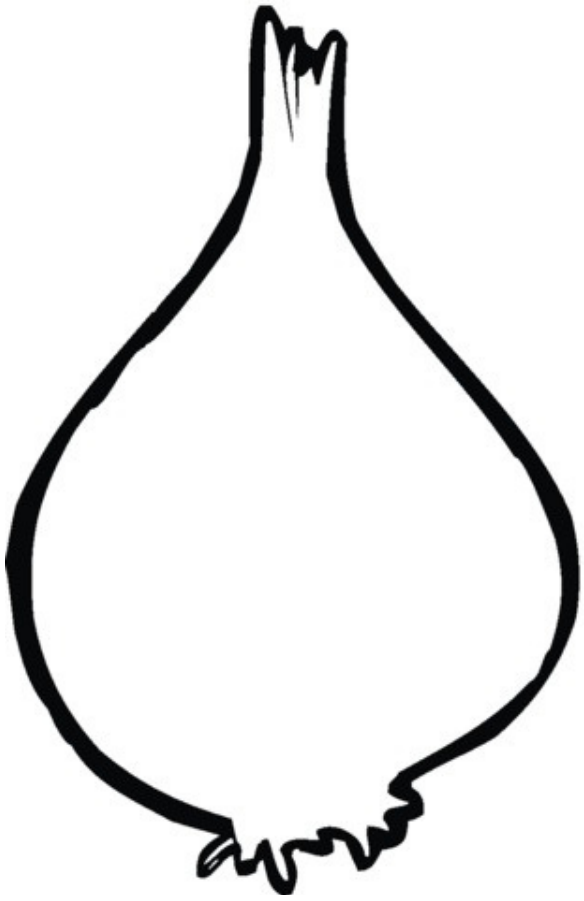
Pour en savoir plus sur la vitesse de traitement et sa relation avec le développement personnel, vous pouvez consulter les sites suivants (en) : https://www.cognifit.com/science/cognitive-skills/processing-speed

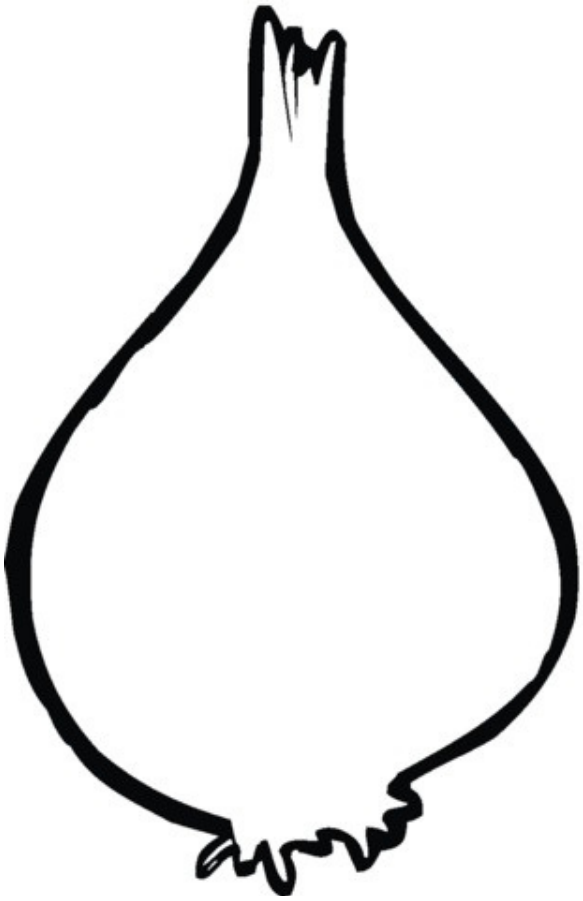
Pour en savoir plus sur la gestion du temps pour les enfants et sa relation avec le développement personnel, vous pouvez consulter les sites suivants (en) ; https://www.scholastic.com/parents/family-life/parent-child/teach-kids-to-manage-time.html

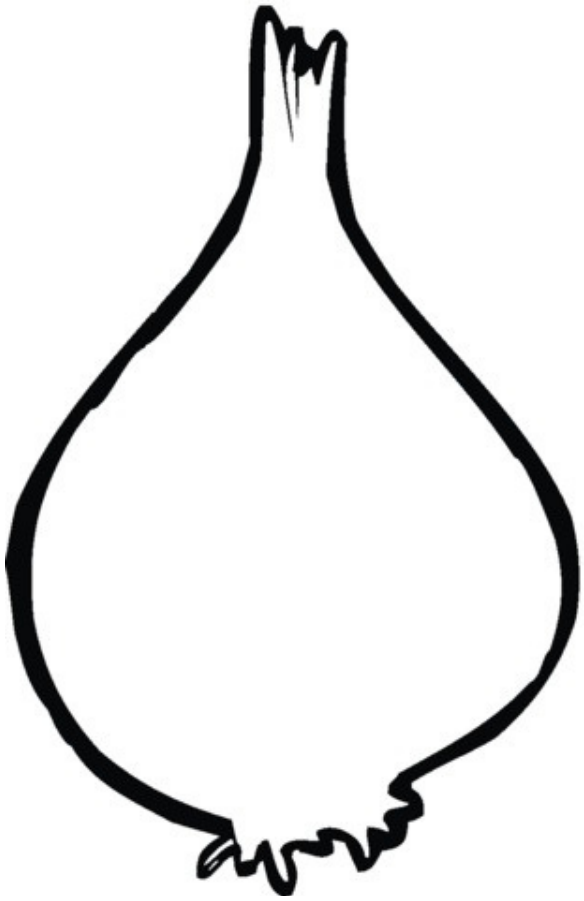
Matériaux

Figures complexes

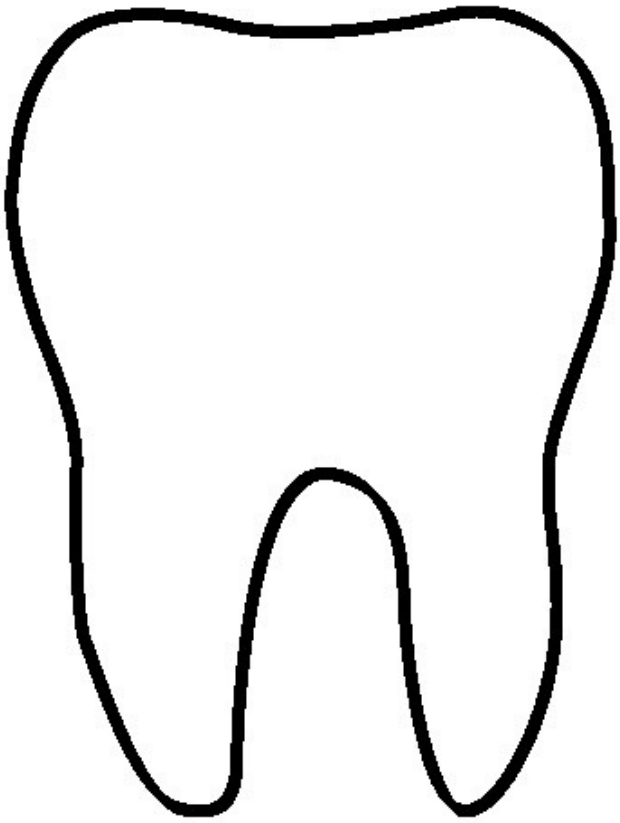


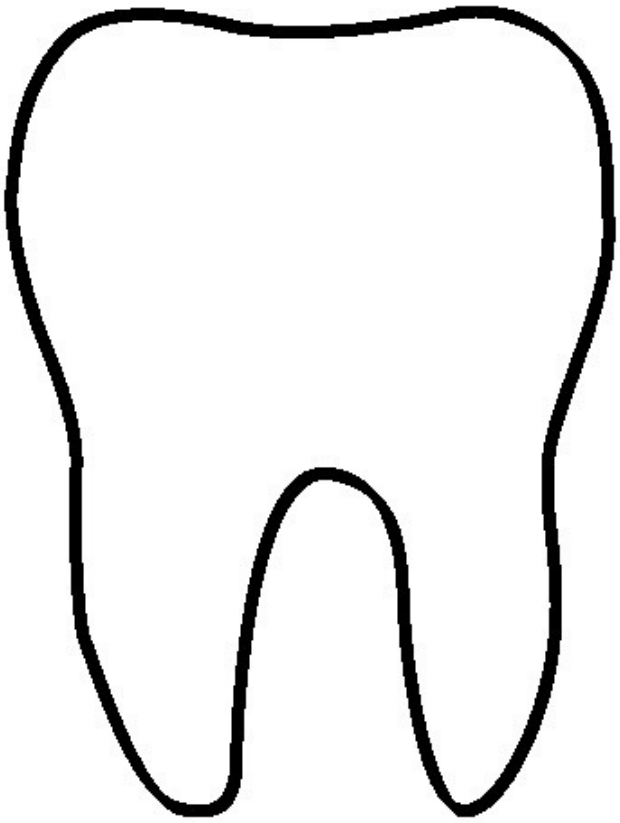


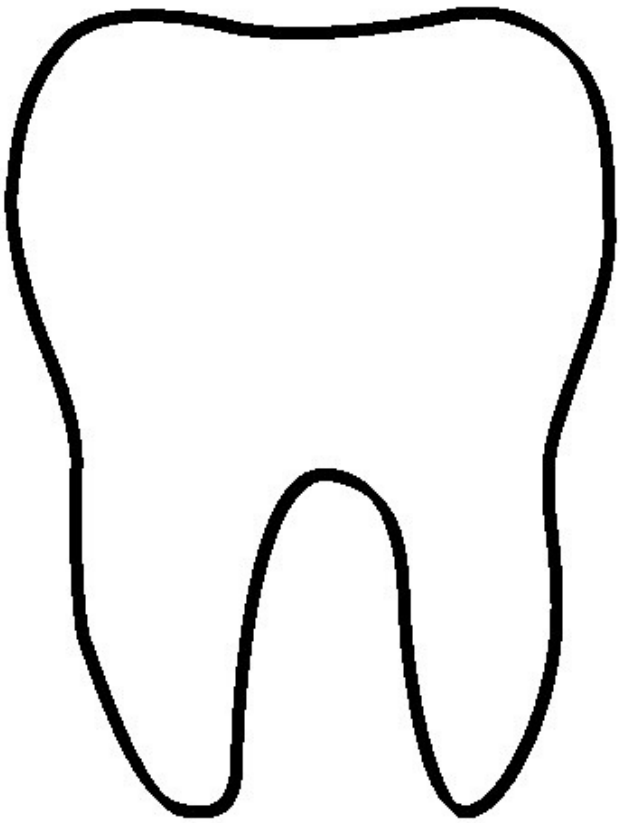


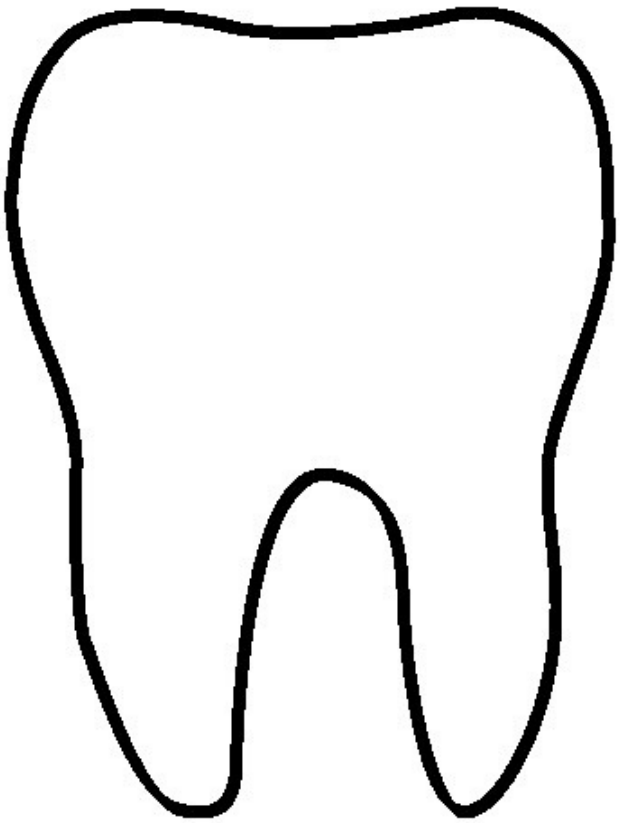


Figures complexes

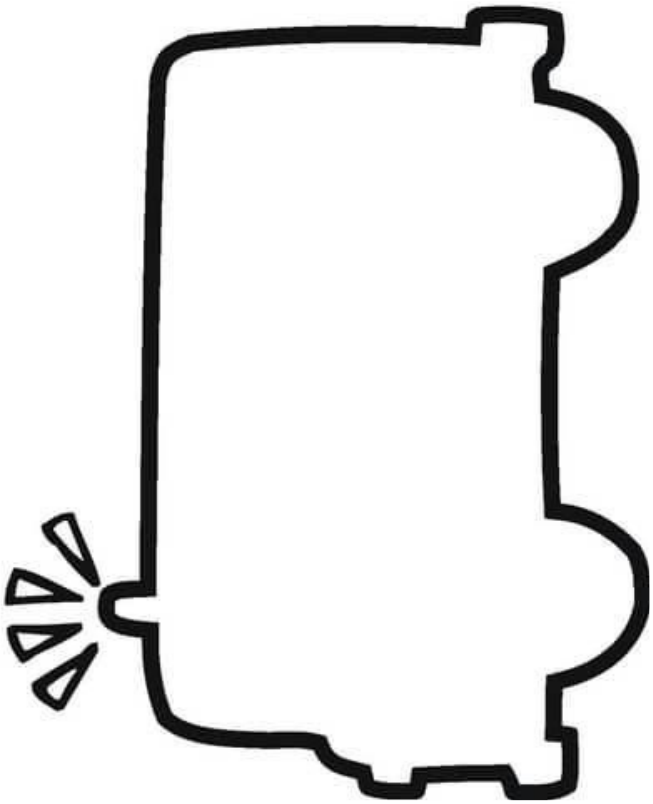


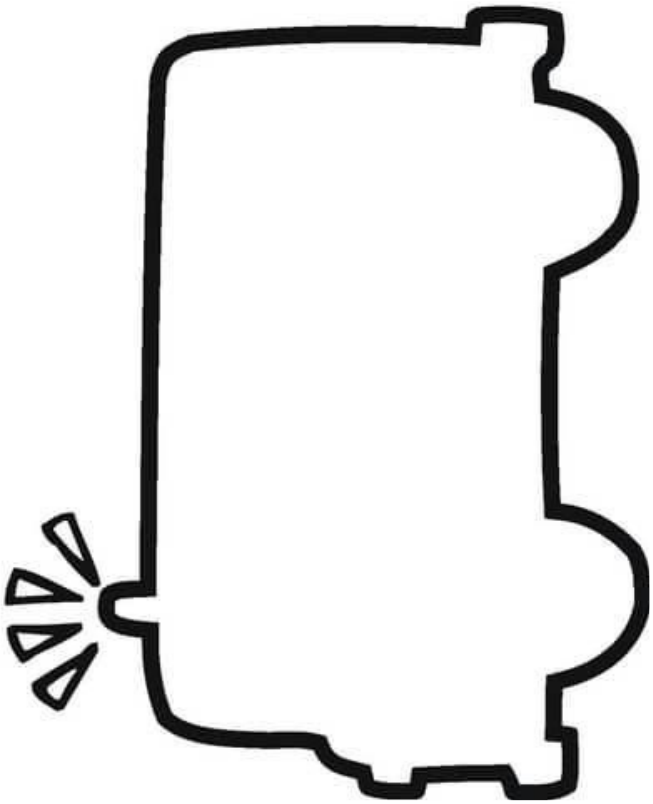


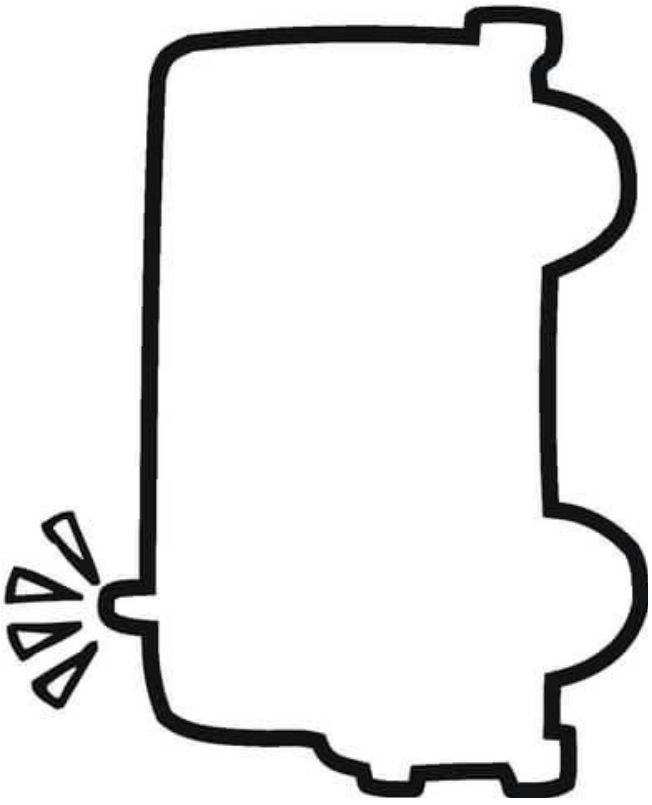


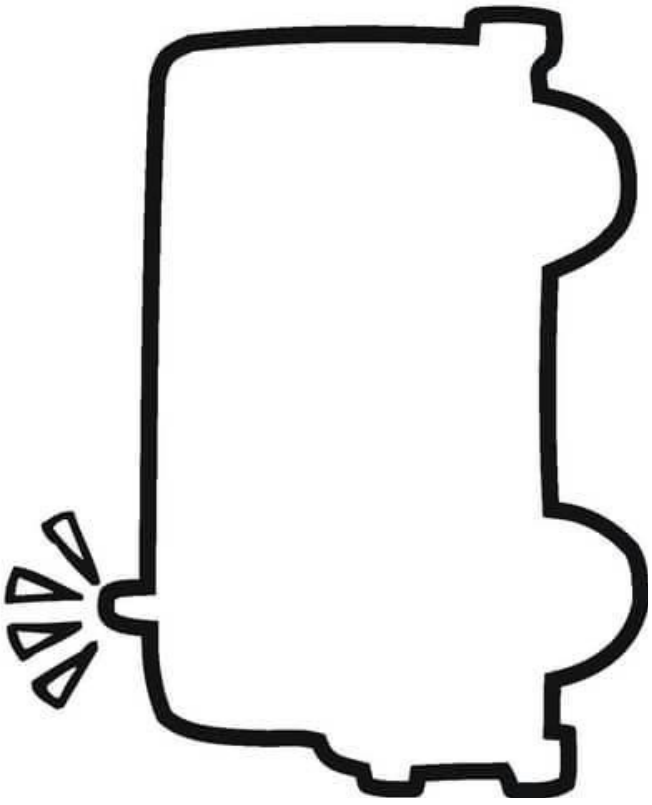


Figures complexes



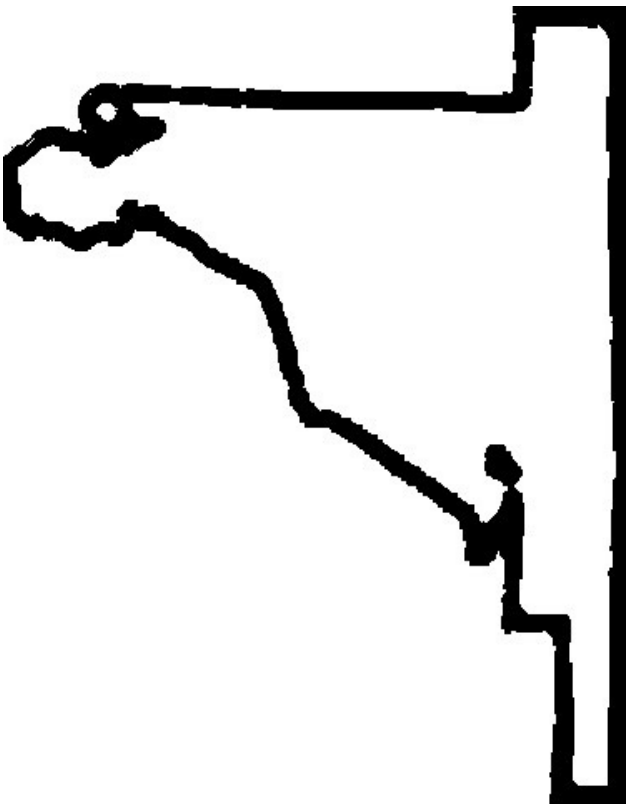


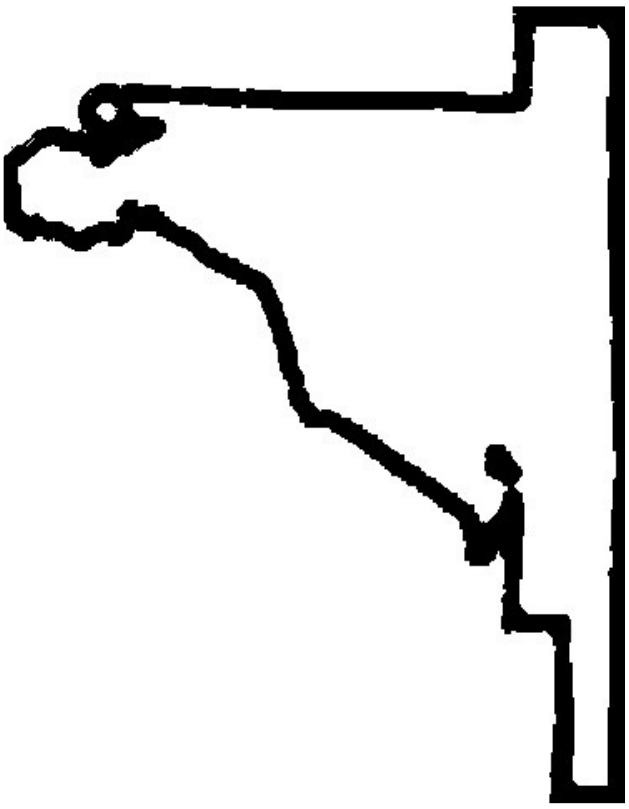


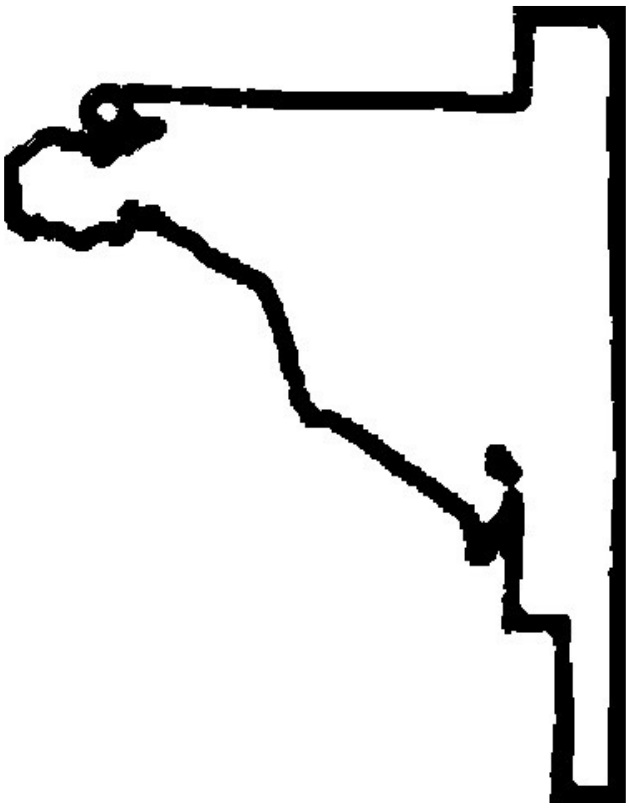


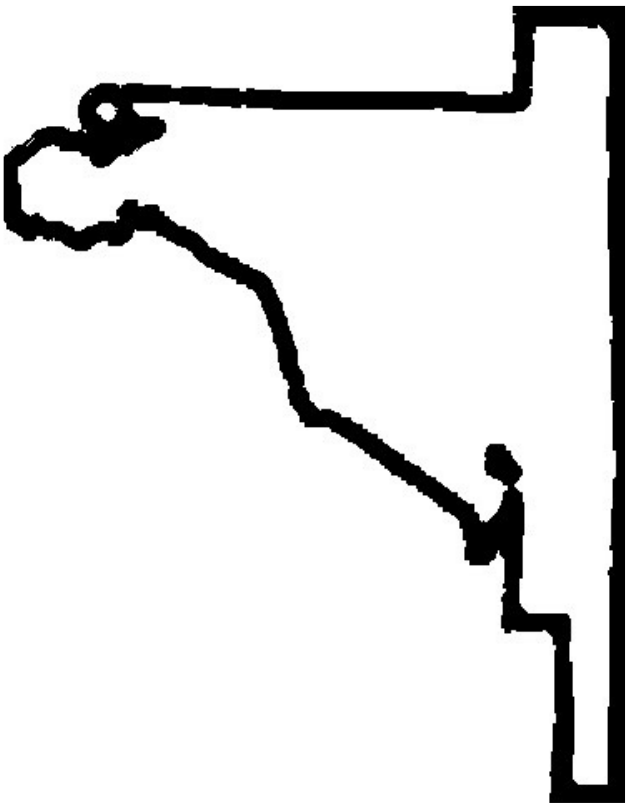
Figures complexes

Figures complexes

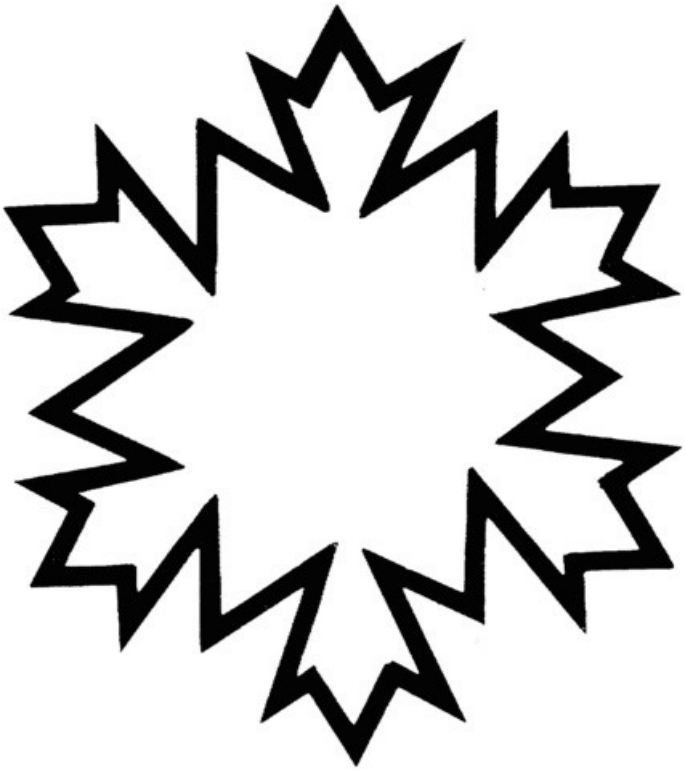


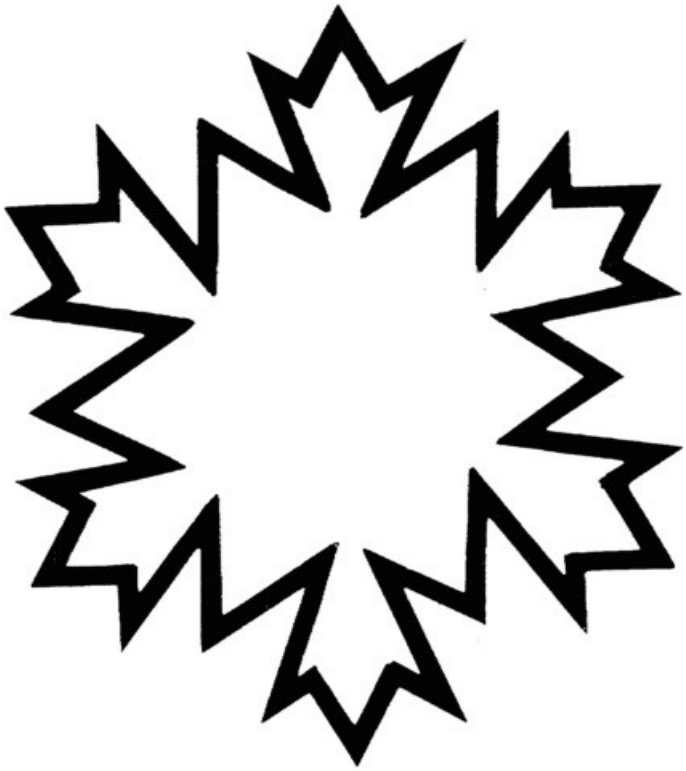


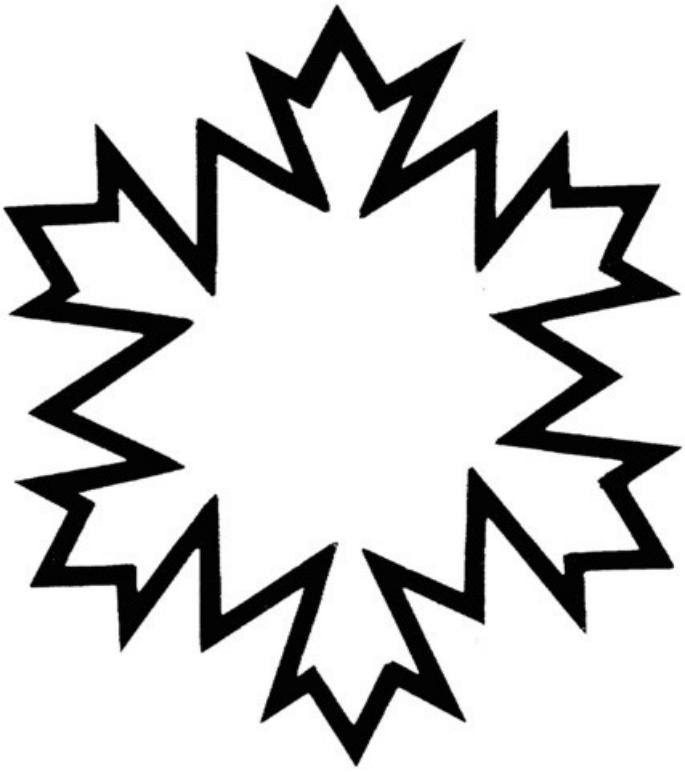


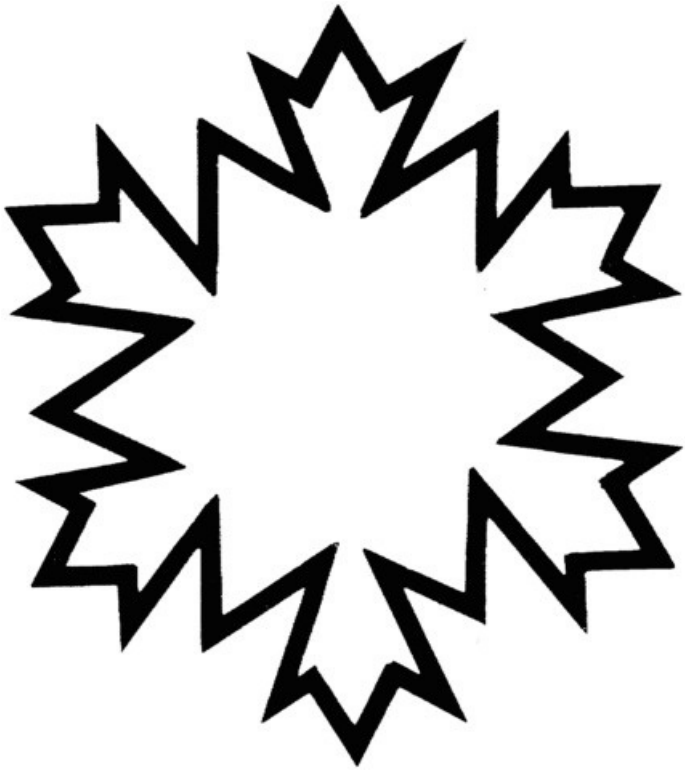


Figures complexes

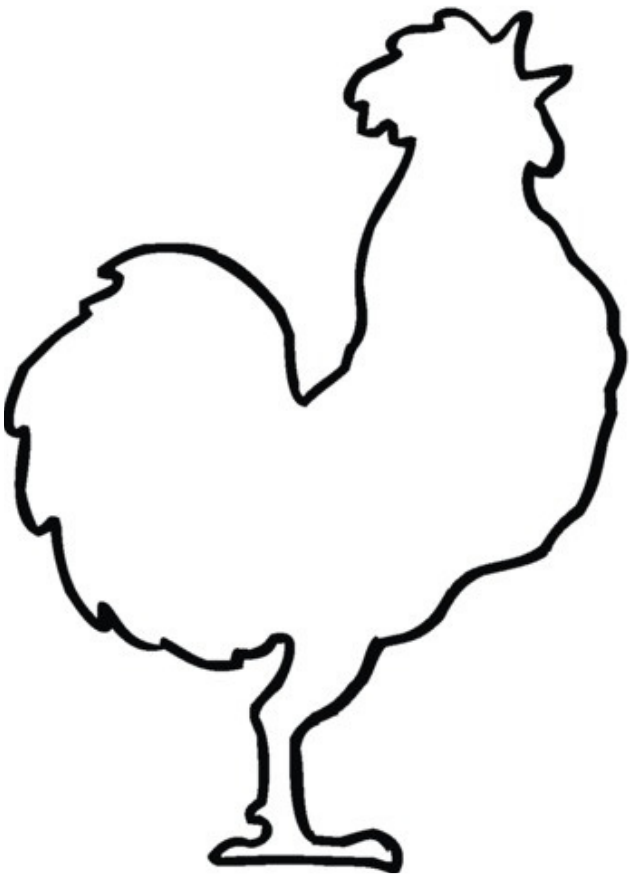


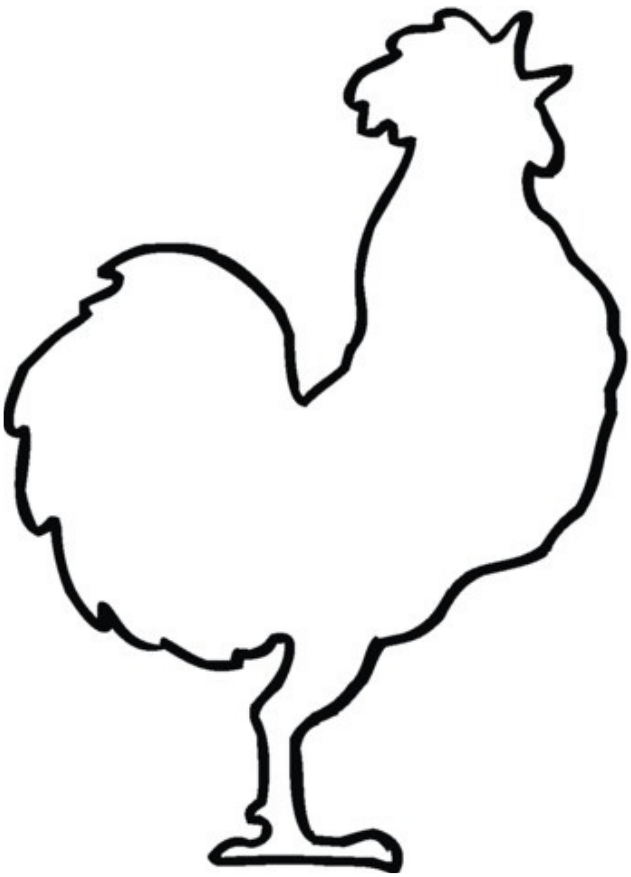


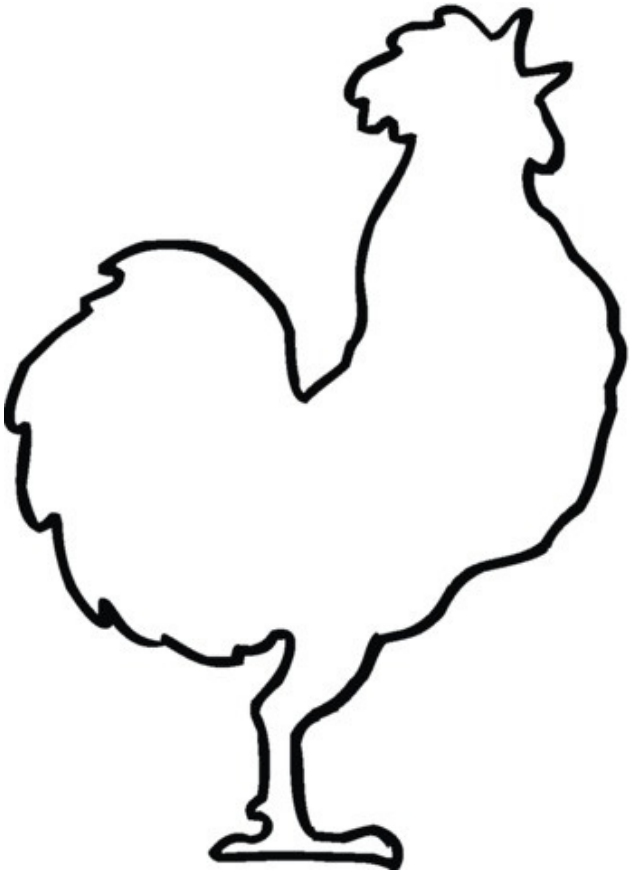


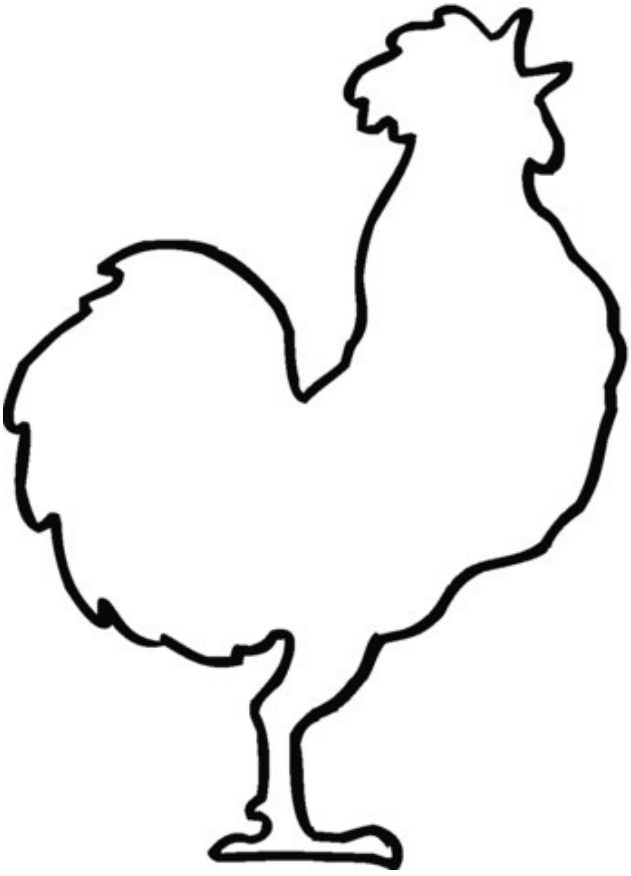


Figures complexes

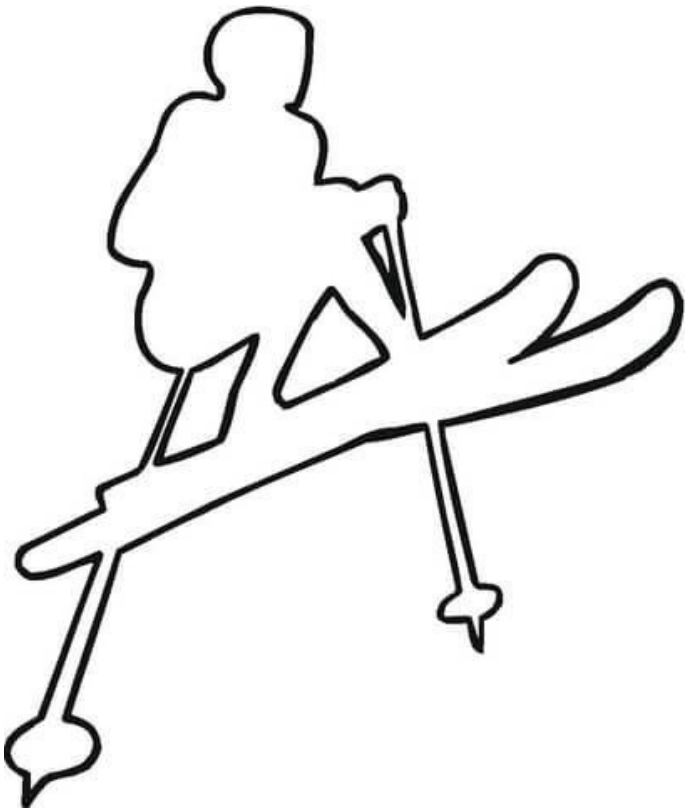


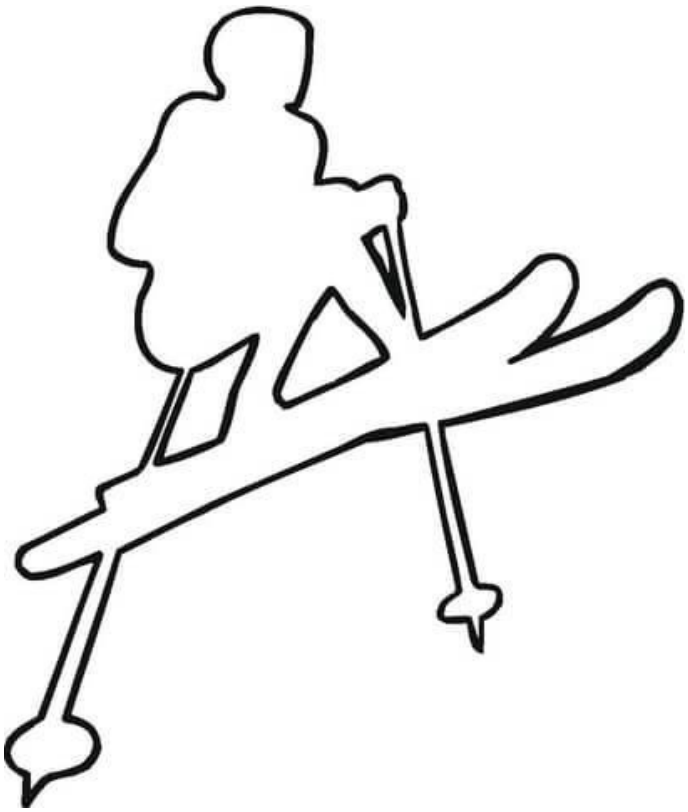




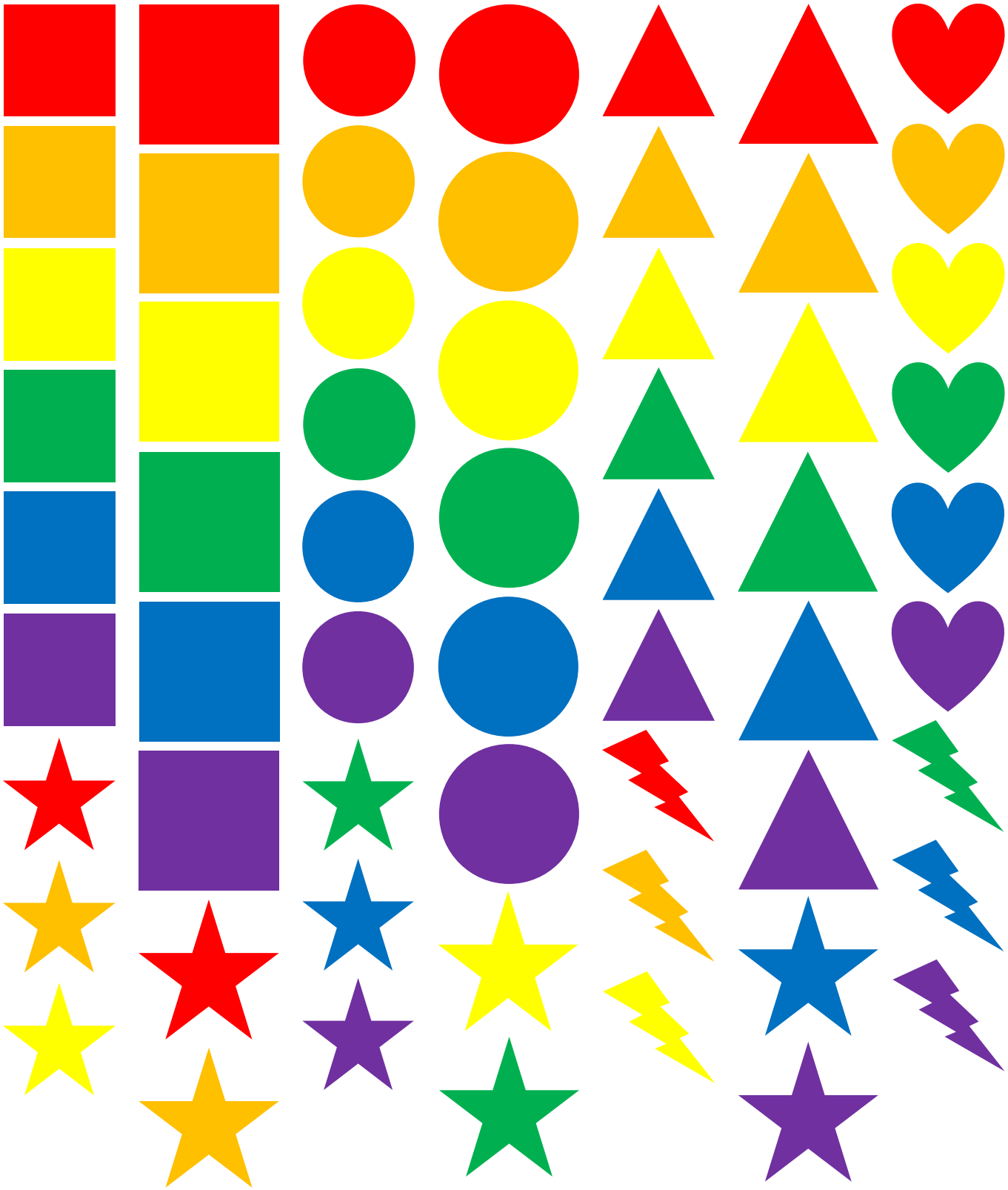


Figures complexes





Figures complexes



Plastic continent

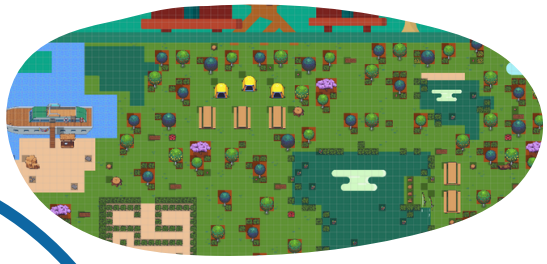
unplugged QUEST

L'action se déroule au milieu d'un continent de plastique créé par la négligence humaine. Un dauphin y est emprisonné et doit se frayer un chemin à travers les tas de déchets pour sortir ! Le jeu sensibilise les enfants à la pollution plastique et aux conséquences négatives de l'activité humaine, améliorant par la même occasion leur réflexion logique et spatiale.

Ressources imprimables fournies : plateau x 1, niveau x 8, pièces x 37 (dauphin x1, déchet de 1 case x 13, déchet de 2 cases x 8, déchet de 3 cases x 7, déchet de 4 cases x 4, déchet de 5 cases x 4)

Matériel additionnel nécessaire : paire de ciseaux

Territoire 3 - Folie dans le Bayou



Durée totale : 10 min.

Nombre de joueurs : 1

ODDs liés

Objectifs d'apprentissage

Comprendre les systèmes naturels et les systèmes techniques

Se repérer dans le temps et l'espace

Raisonner

Comprendre les représentations du monde et de l'activité humaine

Adopter un comportement éthique et responsable

Modalités de jeu

8 - 12 ans	intérieur
En classe	seul

TCette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Développer sa logique : comprendre les mécanismes de construction des niveaux. Le raisonnement logique est la capacité à "prévoir" les implications au-delà des décisions. L'affinement de cette compétence favorise l'apprentissage actif, l'enfant étant encouragé à poser de bonnes questions qui pourraient l'aider dans son propre apprentissage. Dans Plastic Continent, le raisonnement logique est développé en mettant les enfants au défi de trouver un moyen pour le dauphin de sortir d'une pile d'ordures, ou en traçant le chemin d'un objet comportant plusieurs obstacles.

Se repérer dans l'espace : visualiser et prévoir suffisamment à l'avance les coups nécessaires à la réussite du niveau. La conscience spatiale est une compétence cognitive complexe qui nous indique notre position par rapport aux objets qui nous entourent, ainsi que la position relative des objets entre eux. Elle implique la compréhension des concepts de direction, de distance et d'emplacement, et elle est au cœur de l'apprentissage et de l'exécution d'innombrables tâches.
La pensée spatiale est également appelée pensée visuelle. La pensée spatiale vous permet de comprendre l'emplacement et la dimension des objets, ainsi que les relations entre différents objets. Elle vous permet également de visualiser et de manipuler des objets et des formes dans votre tête. Non seulement la pensée spatiale est très importante pour les tâches quotidiennes, mais de nouvelles recherches montrent également qu'elle est essentielle à l'apprentissage des mathématiques. Elle prépare les enfants à traiter des connaissances diverses, en favorisant un traitement efficace d'informations complexes ou potentiellement déroutantes.

Se poser des questions / développer sa curiosité à l'égard du développement durable et de la consommation et de la production durables : s'interroger sur l'impact négatif de l'Homme sur la nature, et sur la pollution des océans. Le jeu aborde le sujet de la pollution plastique dans les mers et océans du monde, en touchant à plusieurs ODD, qui pourraient être discutés en classe : 6 Eau propre et assainissement, 13 Action climatique, 12 Consommation et production responsables, 14 Vie sous l'eau. La représentation d'un océan de plastique pollué où un dauphin suffoque dans des déchets plastiques suscite l'empathie des enfants et les incite à reconsidérer leurs modes de consommation pour adopter un comportement plus durable.

Règles du jeu

But du jeu

Dans Plastic continent, déplacez les tas de déchets sur un échiquier afin de libérer un dauphin pris au piège.

Rôle de l'enseignant et organisation du jeu

Un seul écolier peut jouer à ce jeu à la fois, nous l'appellerons **joueur**. L'adulte en charge des écoliers n'a aucun rôle particulier si ce n'est peut-être aidé à la compréhension des règles.

Installation

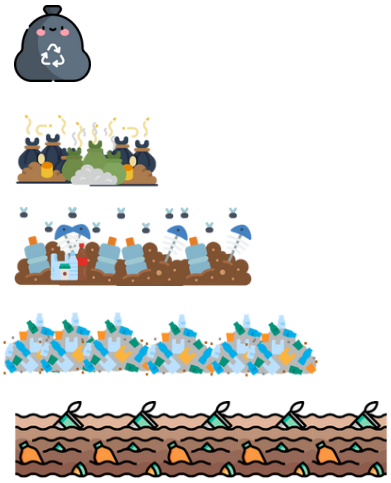
Placez le plateau en forme d'**échiquier** au centre de la table.

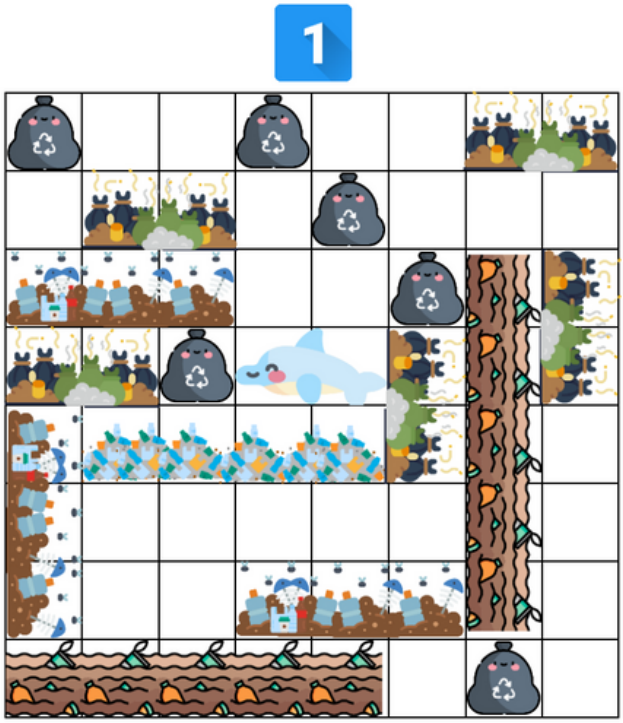
Choisissez un **niveau**.

Ajoutez sur le plateau les pièces, le **dauphin** et les **déchets**, comme indiqué sur l'illustration du niveau.

Il existe cinq types de déchets, définis en fonction du nombre de cases qu'ils couvrent :

- déchet de 1 case
- déchet de 2 cases
- déchet de 3 cases
- déchet de 4 cases
- déchet de 5 cases





Mode de jeu

Règles de déplacement

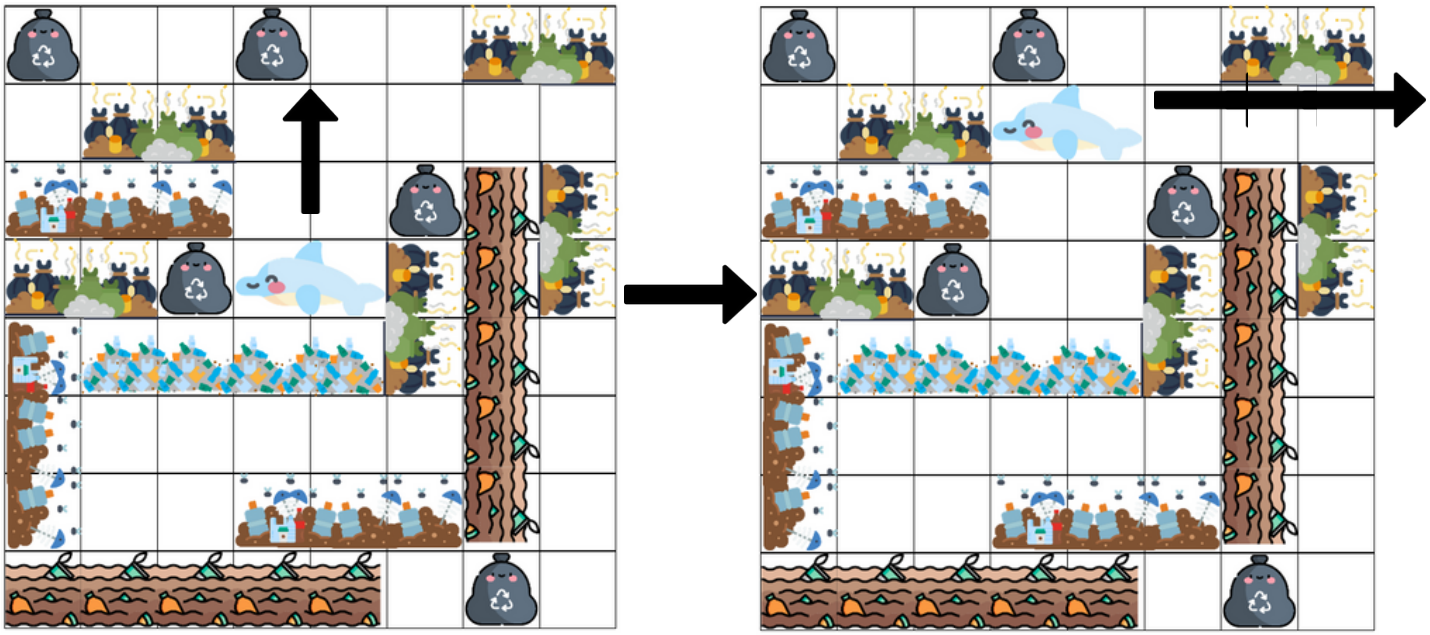
Le déplacement de pièce se fait **en les faisant glisser sur le plateau**. Les pièces ne sont **jamais soulevées**, jamais retournées, et ne chevauchent, même partiellement, jamais aucune autre pièce.

Le joueur peut déplacer n'importe quelle pièce le permettant . Il ne **déplace qu'une seule pièce à la fois**.

Un déplacement peut être fait **vers le haut, vers le bas, vers la gauche ou vers la droite.**

Un déplacement est effectué **en ligne droite jusqu'à être arrêté par un bord du plateau ou une autre pièce**. Un déplacement ne peut pas être interrompu volontairement sans que son mouvement ait été stoppé par un de ces moyens.

Si la pièce déplacée est le dauphin et que le mouvement est **arrêté par un bord du plateau, le joueur emporte la partie**.



Mouvements

Les règles sont les même en difficile à l'exception que les pièces de **déchet de 1 case ne sont pas déplaçables.**

Sujet 1 - Raisonnement logique

Pour développer davantage le raisonnement logique, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Memory, Good ways, Farm in the City, Peace Magic Grid, PotLuck March, Reroute a better world, Dr. Hanoi, etc.***

Pour en savoir plus sur le raisonnement logique, vous pouvez voir (en) : https://criticalthinkingsecrets.com/what-is-logical-thinking/

Sujet 2 - Orientation spatiale, réflexion spatiale

Pour développer davantage le raisonnement logique, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Good ways, Farm in the City, PotLuck March, Reroute a better world, etc.***

Pour en savoir plus sur l'importance de l'amélioration des compétences en matière de réflexion spatiale/visuelle, vous pouvez consulter différentes ressources telles que (en) :

- http://www.xplaner.com/visual-thinking-school/
- https://kids.frontiersin.org/articles/10.3389/frym.2020.00050
- https://babysparks.com/2019/02/20/what-is-spatial-awareness-how-does-it-develop/

Sujet 3 - Objectifs de développement sociétal

Pour se familiariser avec les objectifs de développement social et les actions que vous pourriez souhaiter voir : https://bit.ly/3llzknl

Pour voir d'autres jeux présentant les ODD, vous pouvez ouvrir :

- https://go-goals.org/
- https://culpeer-for-change.eu/quiz/en

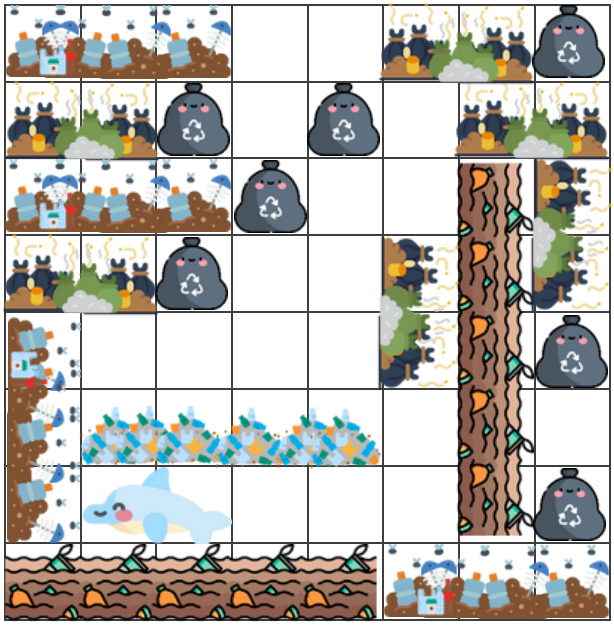
Matériaux

Grilles

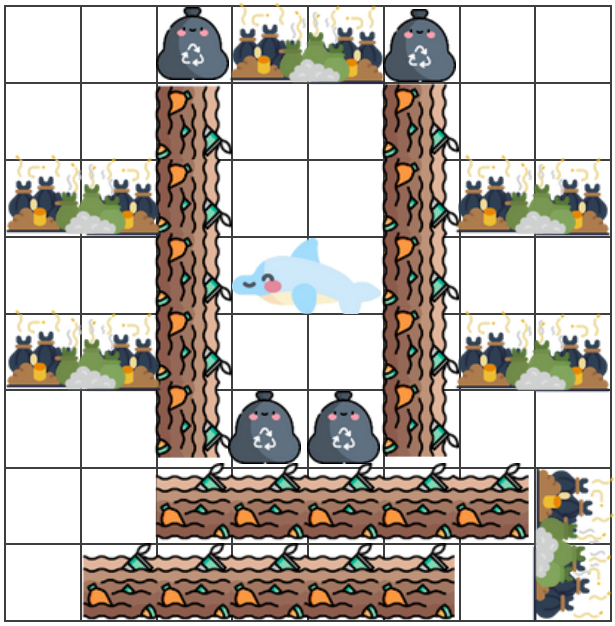
Matériaux – Exemples d'arrangements

1

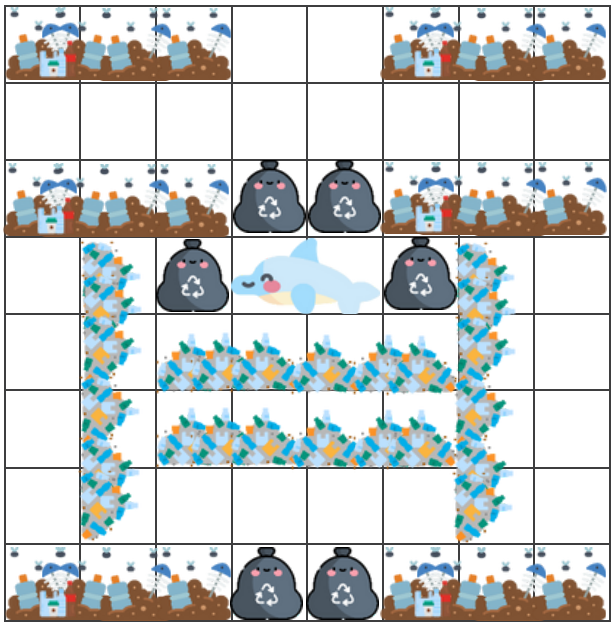
3



2

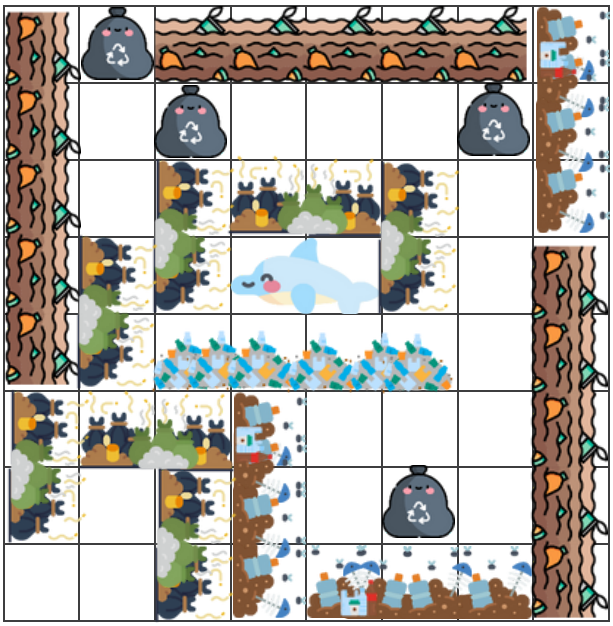


4

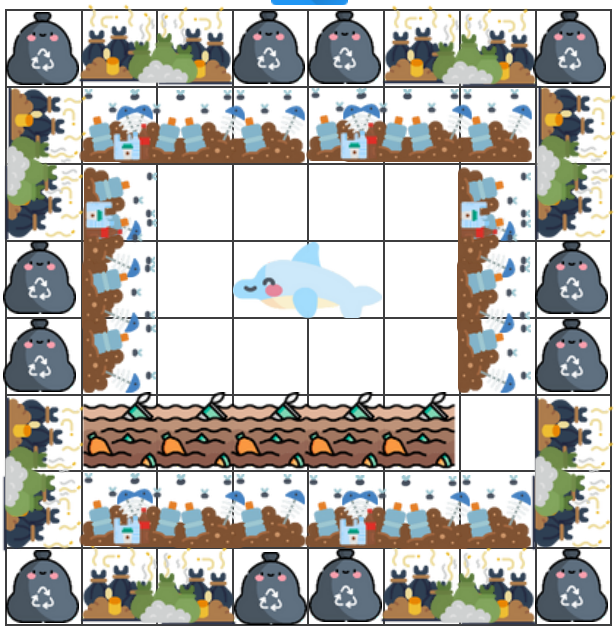


Matériaux – Exemples d'arrangements

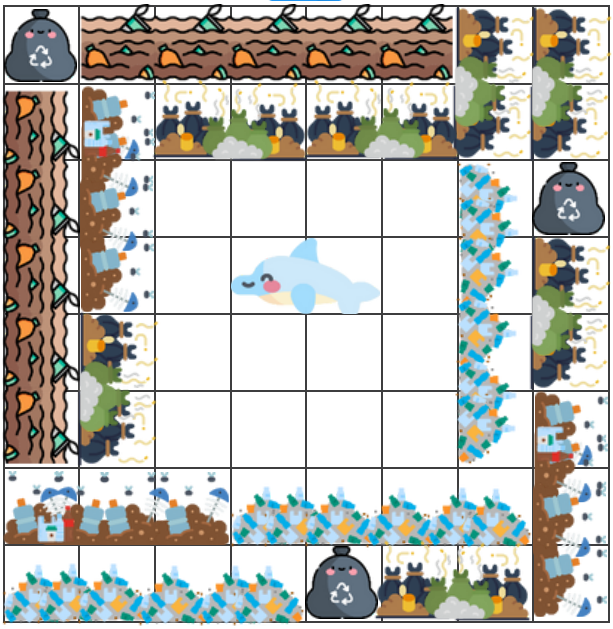
5



7

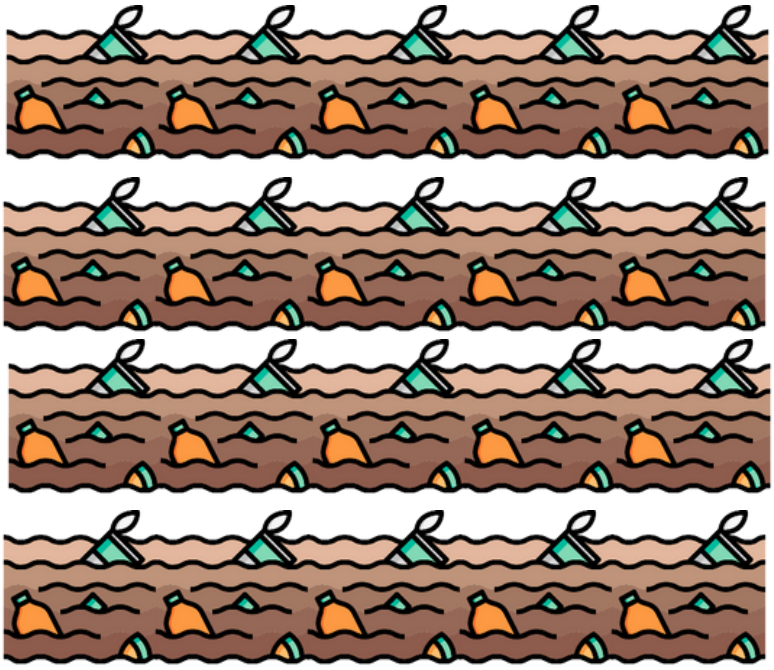


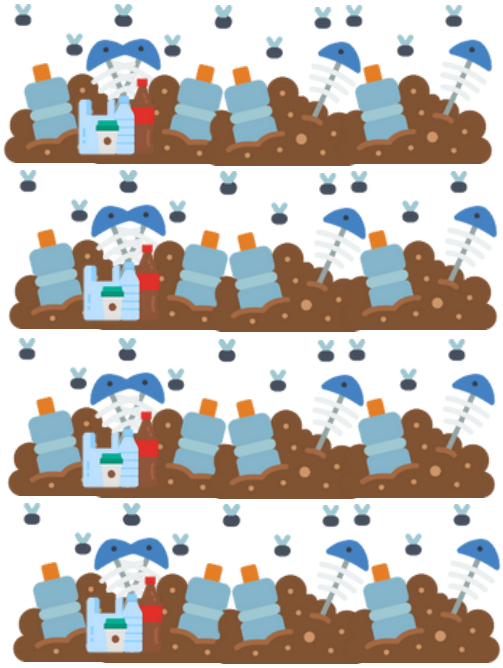
6

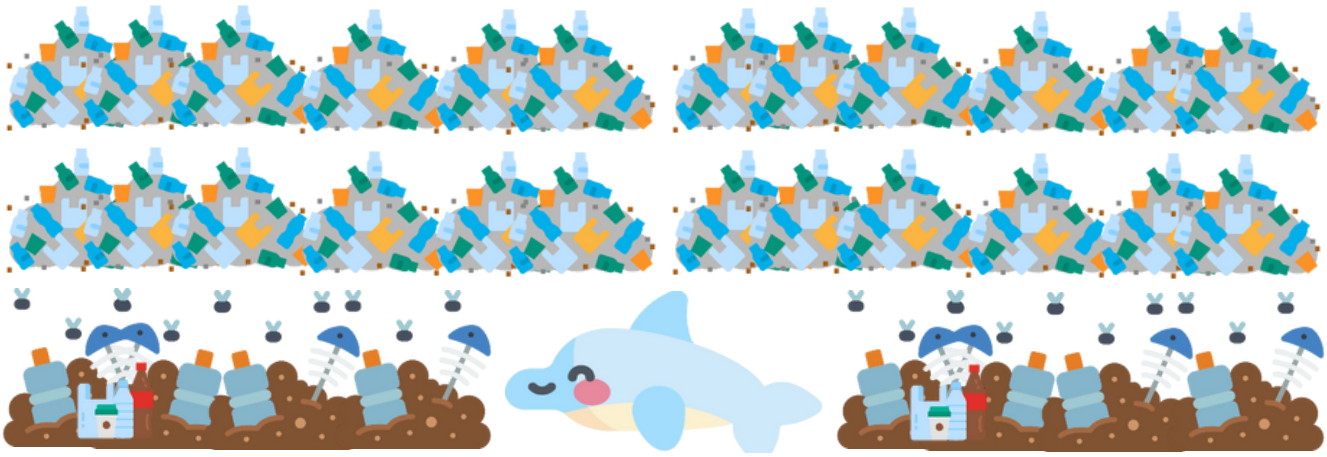


8

Matériaux – Carreaux d'ordures





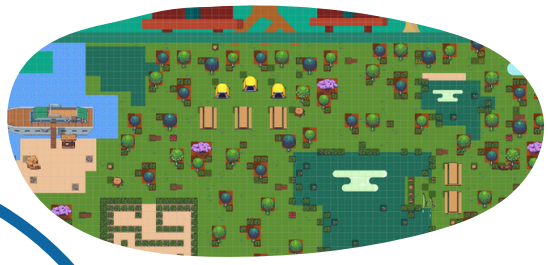


Good ways

unplugged QUEST

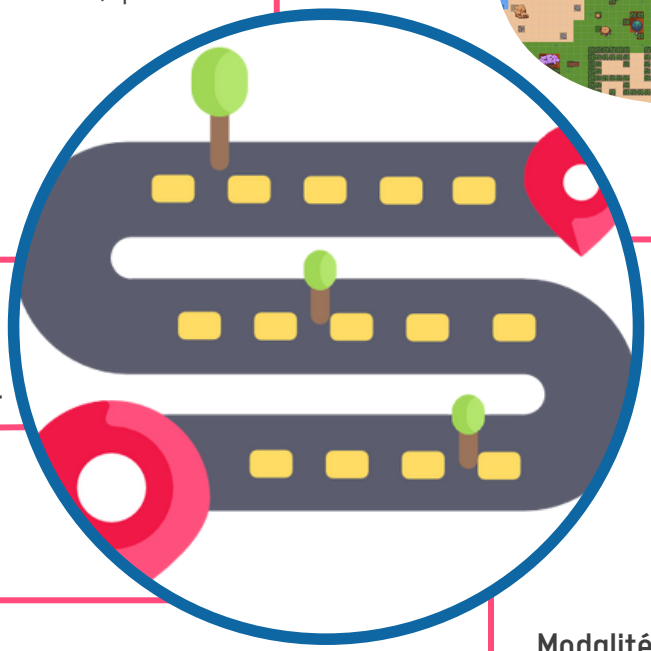
Good ways est un jeu d'observation dans lequel les joueurs tentent de se situer sur des cartes et sur des chemins avant d'essayer de tracer une route continue dans le but d'atteindre les bonus et d'éviter les malus ainsi que les obstacles sur une feuille transparente ne recouvrant pas les cartes.

Territoire 3 - Folie dans le Bayou



Ressources imprimables fournies : carte x 3, point x 74 (1 point x 64, 5 points x 10)

Matériel additionnel nécessaire : feuilles A4 transparentes, marqueur effaçable x 6, paire de ciseaux



Durée totale : 20 min.

Nombre de joueurs : 2-6+

ODDs liés

Objectifs d'apprentissage

Comprendre les systèmes naturels et les systèmes techniques

Raisonner

Comprendre les relations entre objet et espace

Se repérer dans le temps et l'espace

Comprendre les représentations du monde et de l'activité humaine

Modalités de jeu

- 6 - 12 ans
- En classe
- Intérieur
- à la maison
- Seul

TCette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Orientation : aide à se situer dans l'espace et à trouver ses repères. Ce jeu permet d'apprendre l'orientation spatiale. En reproduisant des itinéraires, les enfants apprennent à préparer un plan de transport (moyens de transport, meilleurs itinéraires et points d'arrêt, obstacles, etc.)

Etre sensibilisé à l'engagement sociétal : découvrir ou mieux connaître les objectifs de développement durable, ou ODD (en anglais : *sustainable development goals*, ou SDG), établis par les Etats membres des nations unies. Toutes les cartes du jeu contiennent un certain nombre d'ODD liés au développement durable positif et un certain nombre d'ODD difficiles.

Concentration : faire le meilleur tracé possible sans lever son marqueur. La dextérité et la motricité fine sont améliorées. La motricité fine est la capacité à contrôler les muscles fins - en particulier dans les mains et les yeux - pour accomplir des mouvements et des tâches. Au cours de la petite enfance, les enfants doivent se développer de manière holistique, ce qui inclut le développement physique, intellectuel (cognitif), social et émotionnel. Le développement de la motricité fine et de la motricité globale est important car la réussite scolaire dépend de la bonne maîtrise des muscles par l'enfant. Ils ont besoin d'un tronc solide et d'une bonne posture pour s'asseoir à un bureau sans se fatiguer, de muscles oculaires capables de suivre la lecture, d'une bonne prise du crayon et d'un bon contrôle des doigts pour former les lettres lorsqu'ils écrivent, et bien plus encore.

Règles du jeu

But du jeu

Avec good ways, tracez le meilleur chemin possible à main levée en évitant les embuches sur une feuille transparente avant qu'elle ne soit superposée à la carte !

Rôle de l'enseignant et organisation du jeu

Nous appellerons les écoliers, **joueurs**, et l'adulte en charge des écoliers, **arbitre**.

Lorsque les joueurs jouent à tour de rôle, c'est l'arbitre qui décide de l'ordre.

Installation

Placez les **cartes** au centre de la table. Donnez, à chaque joueur, une **feuille transparente** et un **marqueur**. Disposez les points, 1 point et 5 points, au centre des joueurs.

1 point

5 points

Phase de jeu

Installation

Chaque tour de jeu concerne une carte et est constitué de deux phases consécutives :

- phase de tracé
- phase de résolution

Phase 1 - Phase de tracé

Tous les joueurs effectuent, en même temps, un tracé continu à main levé, sans soulever leur marqueur, sur leur feuille transparente.

Les joueurs doivent essayer de suivre les chemins, toucher les bonus, le plus souvent sous formes d'objectifs de développement durable (SDG) ou de chemins, et éviter de toucher les malus, représentés le plus souvent par des éléments opposés aux SDG.
Voici, en surimpression, le détail des bonus et malus des cartes. Tous les éléments en vert représentent un bonus, y compris les éléments des cartes elles-mêmes comme les herbes ou les feuilles des arbres. De même, tous les éléments en rouge représentent un malus, y compris les éléments des cartes elles-mêmes comme les champignons venimeux ou les crabes. Les éléments des autres couleurs ne sont à considérer ni comme des bonus ni comme des malus.

14
Bonus
Malus
7
10
6
5

Bonus
Malus
11
4
12
9
16
8

Phase 2 - Phase de résolution

Chaque joueur place sa feuille transparente sur la carte et comptabilise ensuite les points qu'il a gagné ce tour-ci de la manière suivante :

- chaque bonus touché fait gagner 1 point (les bonus sont délimités par des bordures, les éléments autres que les SDG, comme les chemins, ne sont comptabilisés qu'une seule fois en tout et pour tout)
- chaque malus touché faire perdre 1 point (les malus sont délimités par des bordures)
- aucun malus touché fait gagner 2 points
- si le nombre de point comptabilisé par un joueur lors d'un tour est négatif, il ne gagne simplement pas de point (il n'en rend pas d'éventuels qu'il aurait gagné sur d'autres cartes)
-

***

Par exemple, sur la carte suivante :

Il est possible de gagner au maximum 9 points :

- *5 points pour chacun des 5 SDG, qui en remporte 1 chacun*
- *1 point pour le chemin*
- *1 point pour n'importe laquelle des étoiles de mer vertes*
- *2 points pour n'avoir touché aucun malus*

***

N'importe quel joueur peut, à tout moment, demander à l'arbitre de faire la monnaie de ses points.
Le joueur ayant obtenu le plus de points à l'issue de toutes les cartes remporte la partie.

Sujet 1 - Orientation

Pour développer davantage l'orientation et la pensée spatiale, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels: ***Farm in the City, Plastic Continent, Reroute a better world etc.***

Pour en savoir plus sur l'importance des capacités de réflexion visuelle, vous pouvez consulter le site suivant (en) : http://www.xplaner.com/visual-thinking-school/

Sujet 2 - Découvrez l'engagement social

Pour vous familiariser avec les objectifs et actions de développement social, vous pouvez consulter le site suivant (en) : https://bit.ly/3llzknl.

Pour voir d'autres jeux présentant les ODD, vous pouvez ouvrir : https://go-goals.org/ ou https://culpeer-for-change.eu/quiz/en

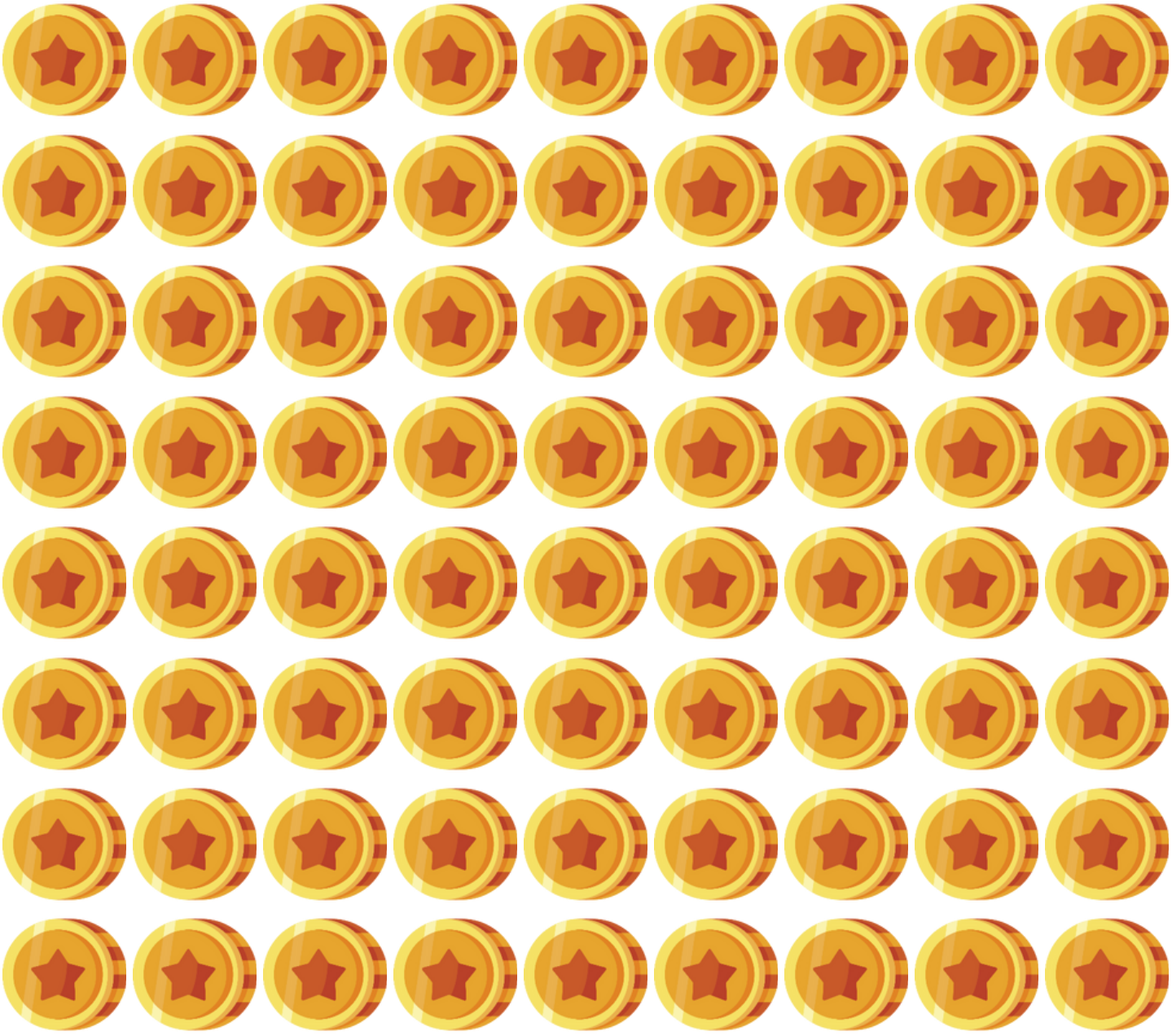
Sujet 3 - Concentration et dextérité

Pour entraîner d'autres enfants à développer leur dextérité et leur motricité fine, vous pouvez vous référer aux autres jeux Unplugged, parmi lesquels: ***Good ways, Pop-up city in the Future, Dr. Hanoi, etc.***

Pour en savoir plus sur l'importance du développement de la motricité fine, vous pouvez consulter les sites suivants (en) : https://empoweredparents.co/why-are-fine-motor-skills-important/

Matériaux

Points & Pièces



15 LIFE ON LAND
13 CLIMATE ACTION
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
1 NO POVERTY

14 LIFE BELOW WATER
7 AFFORDABLE AND CLEAN ENERGY
10 REDUCED INEQUALITIES
6 CLEAN WATER AND SANITATION
5 GENDER EQUALITY

11 SUSTAINABLE CITIES AND COMMUNITIES
4 QUALITY EDUCATION
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
8 DECENT WORK AND ECONOMIC GROWTH

Dr. Hanoï

unplugged QUEST

Le jeu se déroule dans un laboratoire de chimie, avec des tubes à essais, des formules et des éléments chimiques. Les joueurs doivent trouver un vaccin avec la formule chimique correcte en transférant les éléments depuis un tube vers un autre!

Ressources imprimables fournies : cartes objectif x 30, tube à essai x 4, petit élément x 18 (6 de chacun des 3 types), gros élément x 6 (3 de chacun des 2 types)

Matériel additionnel nécessaire : paire de ciseaux

Territoire 2 - La Communauté

Durée totale : 10 min.

Nombre de joueurs : 1 - 3

ODDs liés

Objectifs d'apprentissage

Expérimenter des approches scientifiques

Calculer

Raisonner

Appréhender les quantités et les mesures

Modéliser

Modalités de jeu

8 - 12 ans	intérieur
En classe	Seul

TCette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

Cette œuvre est placée sous licence Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), ce qui permet une utilisation, une distribution et une reproduction sans restriction sur n'importe quel support, à condition de citer de manière appropriée le ou les auteurs originaux et la source, de fournir un lien vers la licence Creative Commons, d'indiquer si des modifications ont été apportées et de partager de la même manière.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Aiguiser sa réflexion : trouver la logique de construction derrière chaque formule scientifique et calculer les étapes pour la reproduire. Les enfants améliorent leurs capacités combinatoires en trouvant les meilleures solutions pour déplacer les pièces d'un tube à essai à un autre. Le Dr Hanoi prépare également à la réflexion stratégique.

Affûter sa dextérité : faire attention de ne pas faire tomber d'élément en les transférant d'un tube à essai à un autre. La dextérité est liée à l'intégration transparente des mains dans le système cognitif, ce qui fait des aptitudes manuelles un élément important de l'interaction des personnes avec l'environnement et de leurs capacités à ressentir, explorer, agir, planifier et apprendre.

Comprendre le fonctionnement basique des tours de Hanoï : aller à la découverte de ce formidable mais néanmoins difficile jeu de réflexion mathématique. Il s'agit d'un puzzle composé de 3 tiges et d'un certain nombre de disques de différents diamètres. Au début, les disques empilés sur la première tige par taille décroissante. L'objectif est de déplacer la pile entière sur la dernière tige. On ne peut déplacer plus d'un disque à la fois, et on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

Règles du jeu

But du jeu

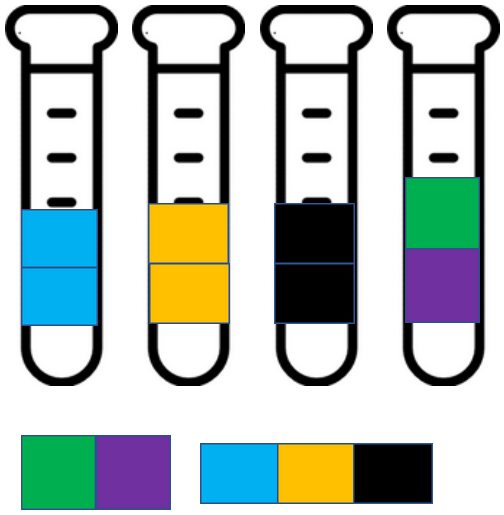
Chaque joueur doit effectuer les bons remplissages de tubes à essai décrits par la formule scientifique en faisant le plus petit nombre de transferts d'un tube à un autre !

Rôle de l'enseignant et organisation du jeu

Nous appellerons les écoliers, **joueurs**, et l'adulte en charge des écoliers ou un écolier choisi par les autres, **arbitre**. Lorsque les joueurs jouent à tour de rôle, c'est l'arbitre qui décide de l'ordre.

Installation

Mélangez le paquet de **cartes objectifs** et placez-le au centre des joueurs. Disposez les **éléments** -**petits et grands**- au centre des joueurs. Donnez à chaque joueur **quatre tubes à essai**. Respectez la composition initiale des tubes à essai donnée ci-après :



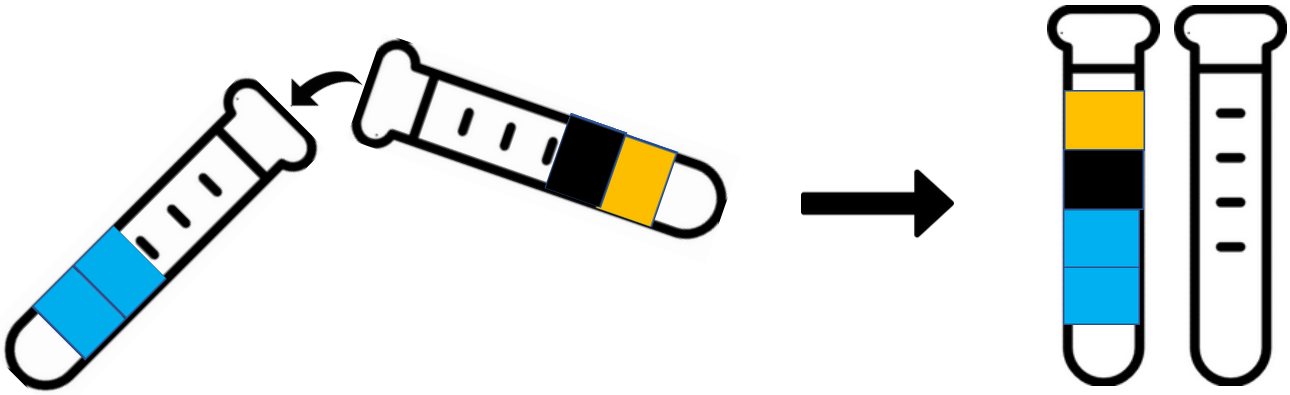
Début du jeu

L'arbitre pioche et dévoile une carte objectif décrivant une formule scientifique composés de petits éléments et de gros éléments dans 3 tubes à essai. Chaque joueur doit réaliser les formules dans le sens indiqué par les tubes à essai de la carte.

Phases de jeu

Comment jouer ?

A tour de rôle, chaque joueur effectue **un transfert**, partiel ou complet, d'un tube à essai à un autre.



Un élément ne peut être transféré qu'au dessus d'un élément de taille égale ou supérieure, ou au bas d'un tube vide. Ainsi, les gros éléments doivent toujours être au bas des tubes à essai, et ne peuvent pas être au-dessus de petits éléments.

Attention, un transfert a pour effet d'inverser l'ordre des éléments à l'intérieur des tubes à essai !

Règles diverses

Tous les joueurs doivent respecter les règles suivantes :

- il est possible de permuter ses propres tubes devant soi, cela **ne compte pas comme un transfert**.
- à tout moment, chaque tube à essai **ne peut contenir que 4 éléments**, petits ou gros, à la fois.
- si un joueur fait tomber un élément en dehors d'un tube à essai, **il est éliminé.**
- un joueur le désirant peut abandonner à tout moment, il est alors éliminé.

Fin de partie

Le ou les joueurs complétant en premier la formule scientifique avec leurs tubes à essai, c'est-à-dire ceux l'ayant fait avec le plus petit nombre de transferts, emportent la partie.
Si tous les joueurs sont éliminés, il n'y a pas de gagnant.

Bonus

En début où fin de partie, l'enseignant/éducateur peut utiliser les tests de logiques à faire par écrit disponible dans la section matériaux/bonus pour diversifier le jeu.

Sujet 1 - Réflexion logique et stratégique

Pour développer davantage le raisonnement logique, tu peux te référer à d'autres jeux Unplugged, parmi lesquels : ***Memory, Good ways, Farm in the City, Peace Magic Grid, PotLuck March, Reroute a better world, Plastic continent, etc.***
Pour en savoir plus sur le raisonnement logique, vous pouvez voir https://criticalthinkingsecrets.com/what-is-logical-thinking/.

Pour en savoir plus sur la combinatoire et sa relation avec l'informatique, vous pouvez lire les documents suivants https://www.carthrottle.com/post/2zm2ele/

Sujet 2 - Dextérité

Pour développer davantage la dextérité, vous pouvez vous référer à d'autres jeux Unplugged, parmi lesquels : ***Good ways, Farm in the City, Pop-up city of the Future, Reroute a better world, Plastic continent, Brain Twister, etc.***

Pour en savoir plus sur l'importance du développement de la dextérité et du contrôle des mains, vous pouvez lire ici (en) : https://childdevelopment.com.au/areas-of-concern/fine-motor-skills/hand-control/

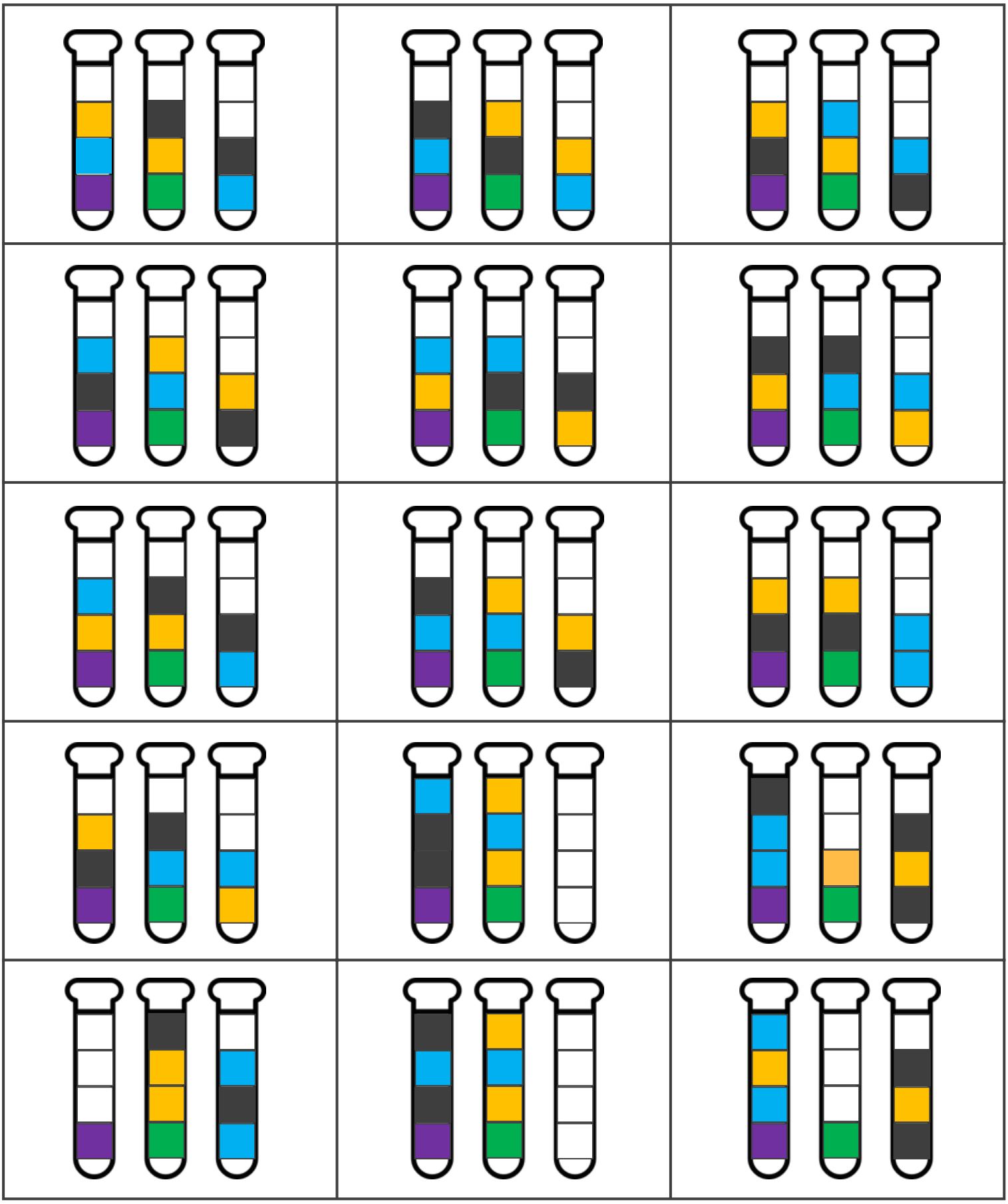
Pour vous inspirer d'autres idées pour entraîner la dextérité, vous pouvez vous référer à (en) : https://www.teachearlyyears.com/images/uploads/article/Developing_fine_motor_skills.pdf

Sujet 3 - Tour de Hanoi

Vous souhaiterez peut-être découvrir le célèbre jeu mathématique Tour de Hanoï : https://fr.wikipedia.org/wiki/Tours_de_Hano%C3%AF ou jouez-la ici: https://www.mathsisfun.com/games/towerofhanoi.html

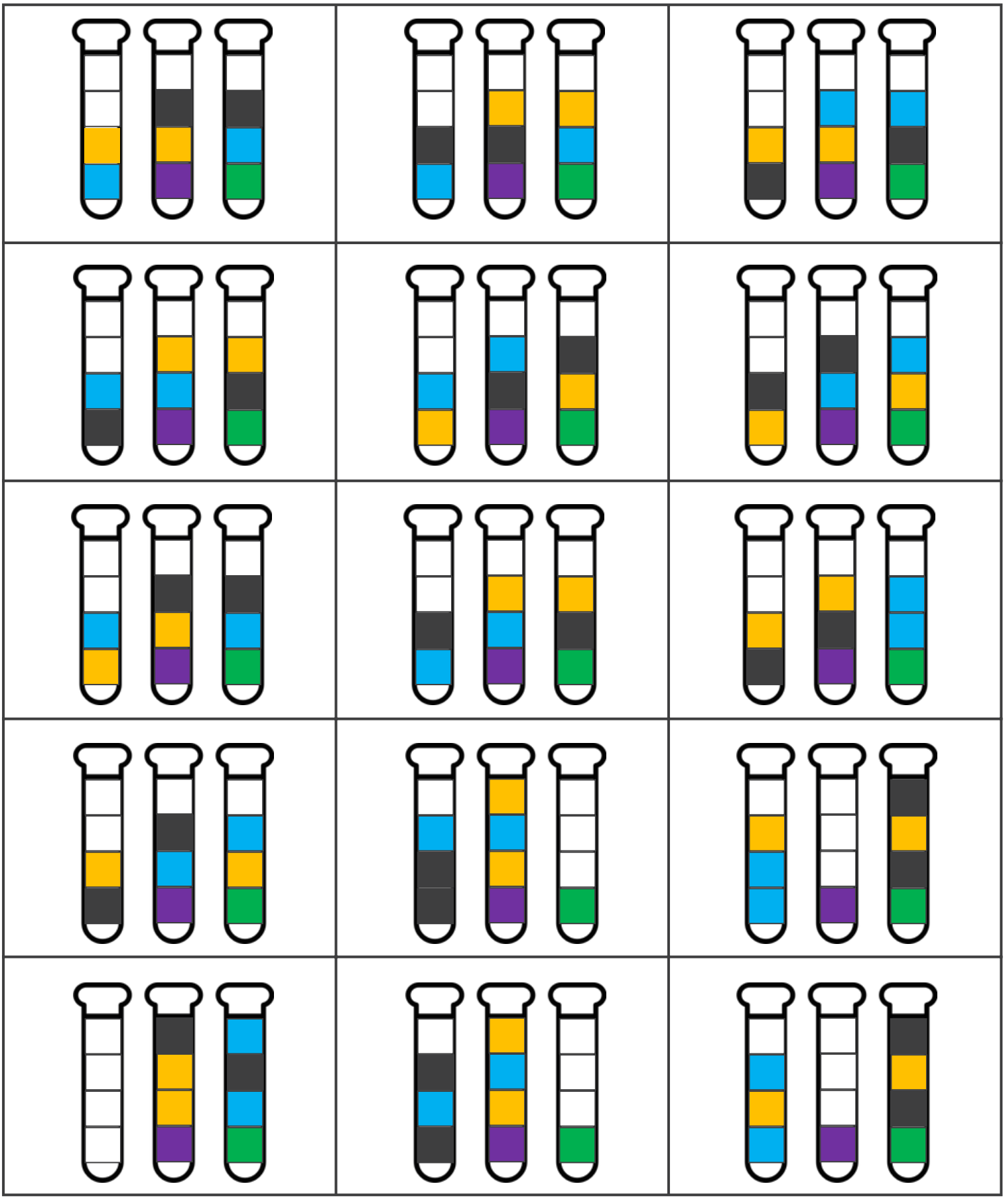
Matériaux

Cartes de but

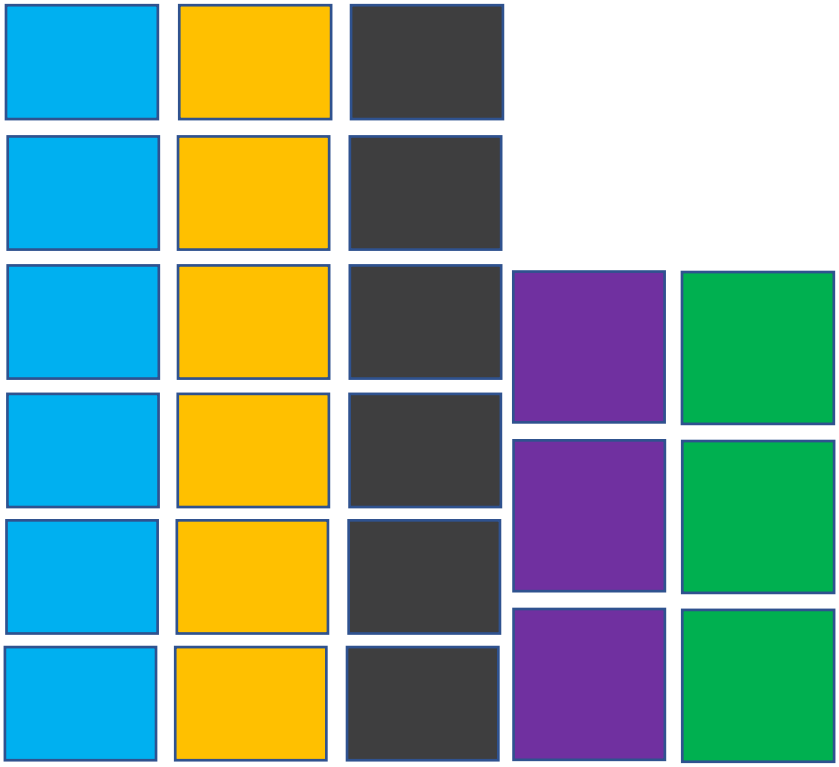


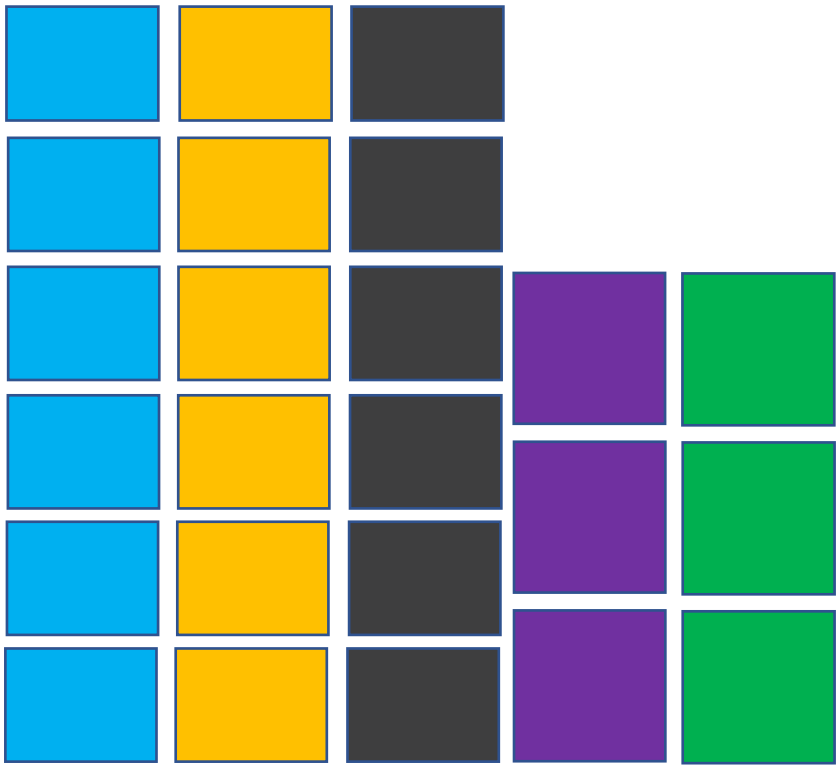
Matériaux

Cartes de but

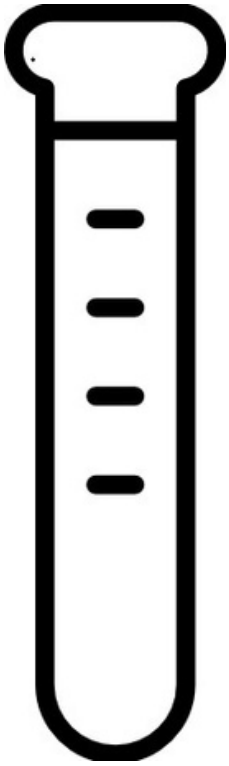
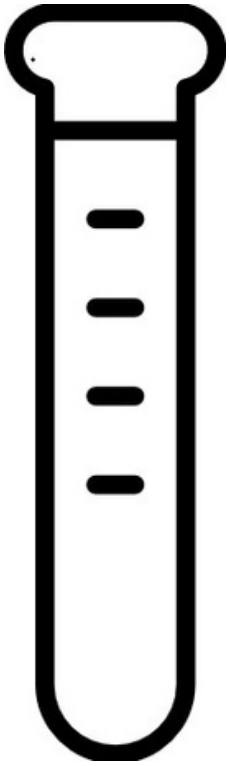
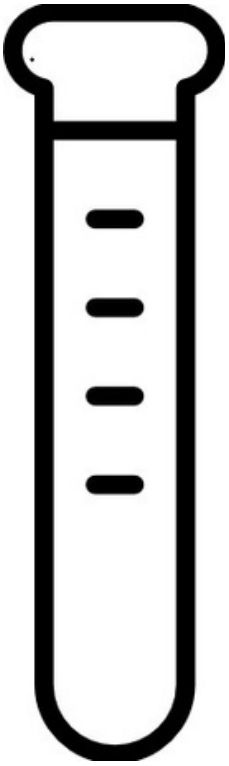
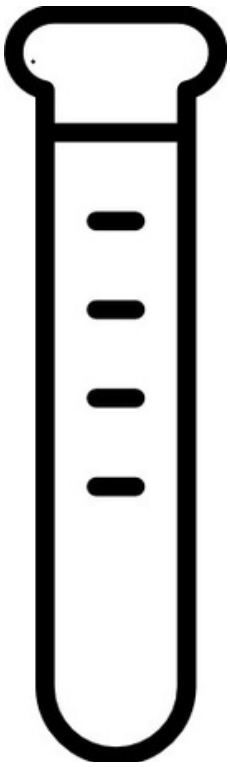
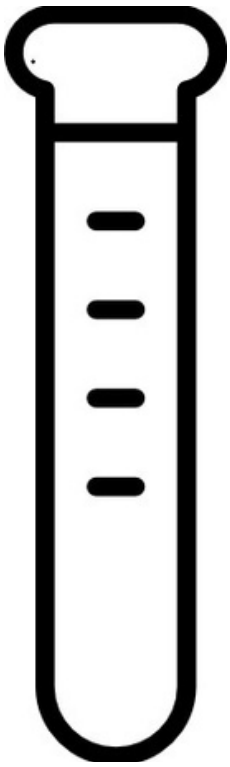
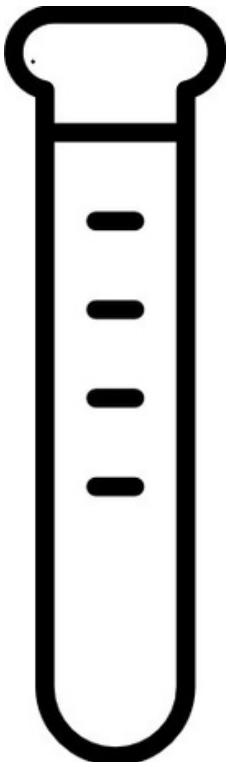


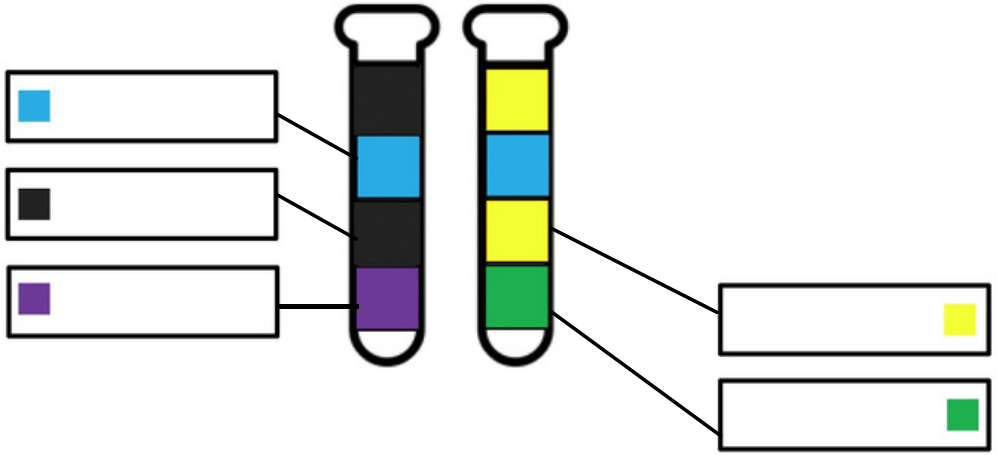
Matériaux - Petites et grandes pièces





Matériaux – Tubes à essai



Consigne : Placez le nom de chacun de ces liquides dans la bonne case :
Eau déminéralisée, Brome, Mercure, Formol, Lithium

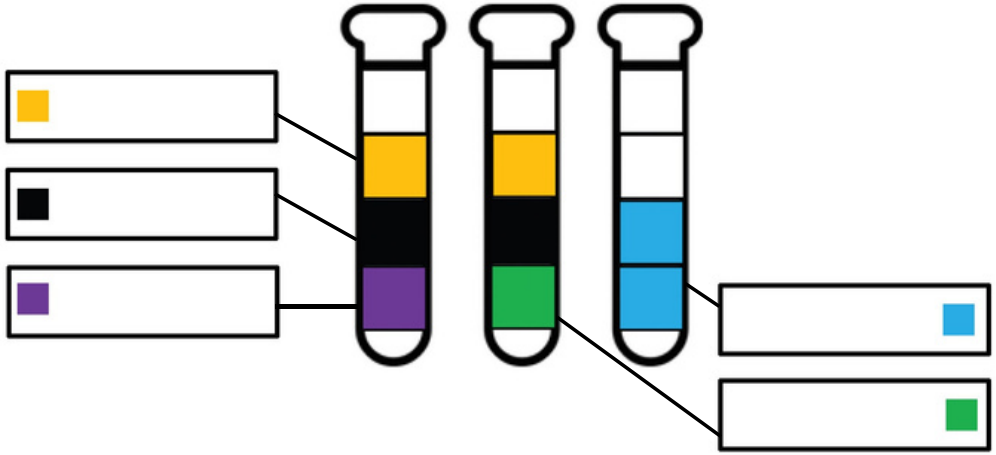
Quelques indices :

Le brome et **le mercure** sont les liquides les plus lourds.

Le brome est dans la fiole à droite de celle qui contient **le mercure.**

Le lithium ne doit pas être mélangé avec **le mercure** sinon il explose.

Chaque flacon contient de **l'eau déminéralisée.**



Consigne : Placez le nom de chacun de ces liquides dans la bonne case :
Eau déminéralisée, Brome, Mercure, Huile, Alcool

Quelques indices :

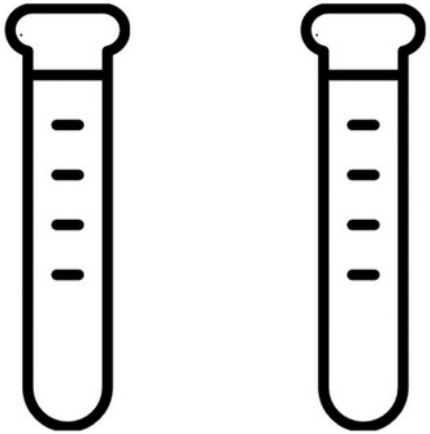
L'eau déminéralisée n'est pas mélangée aux autres liquides.

L'huile est le liquide le plus léger.

Le brome et **le mercure** sont les liquides les plus lourds.

Le brome est dans la fiole à droite de celle qui contient **le mercure.**

Réponse



Consigne : Trouver la composition de chaque flacon.

Huile x 2	Eau déminéralisée	Lithium
Mercure	Brome	

Le brome et **le mercure** sont les liquides les plus lourds.

Le brome est dans la fiole à droite de celle qui contient **le mercure.**

Le lithium ne doit pas être mélangé avec **le mercure.**

L'huile est toujours le liquide le plus léger.

Reroute better world

unplugged QUEST

L'action se déroule au sein des territoires de l'Union Européenne. Choisissez votre type de transport et organisez votre plan de transport afin d'aller aussi loin que possible par voie terrestre, rail, voie maritime ou fluviale et voie aérienne, mais attention aux autres transporteurs qui pourront emprunter les mêmes routes que vous et vous gêner dans votre cheminement en tentant d'arriver, eux aussi, à leur fin.

Ressources imprimables fournies : plateau x 1, pion de transport x 12, carte de ville de départ x 12, carte de transport x 33, point x 74 (1 point x 64, 5 points x 10)

Matériel additionnel nécessaire : paire de ciseaux

Territoire 2 - La Communauté

Durée totale : 30 min.

Nombre de joueurs : 2-12

ODDs liés

Objectifs d'apprentissage

- Justifier une démarche et des choix effectués
- Comprendre les relations entre objet et espace
- Recueillir des références géographiques, historiques et culturelles
- Modéliser
- Raisonner

Modalités de jeu

- 8 - 12 ans
- A la maison
- En classe
- En groupe
- Intérieur

Cette production fait partie du matériel produit par le projet Unplugged qui a reçu un financement du programme ERASMUS + de l'Union européenne sous la convention de subvention n° 2020-1-FR01-KA227-SCH-095528. Cette publication n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient.

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (http://creativecommons.org/licenses/by-sa/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made and share alike.

Cofinancé par le programme Erasmus+ de l'Union européenne

Intérêt pédagogique et sujets abordés

Développer sa pensée computationnelle : résoudre des problèmes de programmation de routes logistiques. Un certain nombre de conditions préalables préparent les enfants à appliquer des compétences de raisonnement et de logique afin de préparer leurs stratégies pour gagner. Il apprend également à adapter les stratégies en cours de route car il propose un conflit de stratégies entre les joueurs.

Se repérer dans l'espace : visualiser et prévoir suffisamment à l'avance les étapes nécessaires pour aller le plus loin possible, et si possible sur une autre ville de départ. Les enfants améliorent leurs connaissances géographiques des grandes villes de l'Europe. L'orientation spatiale et les moyens de transport sont abordés de manière ludique.

Cohabiter avec les autres : essayer de frayer une route au milieu des autres transporteurs en essayant de se gêner le moins possible. Les concepts de base du vivre ensemble sont pensés face aux intérêts croisés et aux compromis.

Règles du jeu

But du jeu

Avec Reroute a better world, élaborez des plans de transport pour amener votre marchandise le plus loin possible. Les autres transporteurs ont le même but que vous, et il faudra tenter de prévoir leur programmation afin d'éviter de croiser leur route !

Rôle de l'enseignant et organisation du jeu

Nous appellerons les écoliers, **joueurs**, et l'adulte en charge des écoliers, **arbitre**.

Lorsque les joueurs jouent à tour de rôle, c'est l'arbitre qui décide de l'ordre.

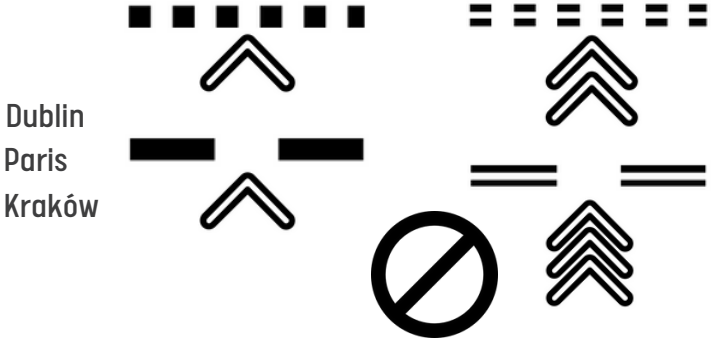
Installation

Placez le **plateau de jeu** au centre de la table.

Donnez le **pion de transport** de son choix à chaque joueur.

Mélangez le paquet de **cartes de villes de départ** et placez-le au centre des joueurs. Faites de même avec le paquet de **cartes de transport**.

A tour de rôle, les joueurs piochent une carte de ville de départ, qu'ils gardent, et pose leur pion de transport sur la ville correspondante (en orange) du plateau de jeu.



Phases de jeu

Phase 1 - Installation

Disposez les **points**, 1 point et 5 points, au centre des joueurs.

1 point

5 points

Chaque tour de jeu est constitué de trois phases consécutives : phase de programmation, phase de transport, phase de comptabilisation.

Phase 2 - Phase de programmation

Mélangez le paquet de cartes de transport. Lors d'une phase de programmation, chaque joueur **:**

- **pioche 4 cartes de transport**, et **en prend connaissance seul**
- **choisi l'ordre de programmation de son plan de transport en empilant, faces cachées, toutes les cartes de transport tirées ce tour-ci** (les cartes sont prises en compte du haut vers le bas)

Pour planifier leur stratégie de déplacement en fonction de leurs cartes de transport disponibles, les joueurs considèrent la ville de départ, les moyens de transport dont elle dispose et l'itinéraire qu'elle pourrait emprunter avec les moyens disponibles et les villes les plus proches qu'elle pourrait traverser.

Phase 3 - Phase de transport

Lors d'une phase de transport, les 4 étapes de transport sont résolues une par une de la manière suivante :

- **tous les joueurs révèlent simultanément la carte de transport du dessus de leurs plans de transport**
- **tous les transports possibles (voir ci-après) des pions de transports sont effectués**
- **Les joueurs ayant la plus haute priorité pour le mode de transport respectif jouent en premier.** La priorité est déterminée par le **nombre de flèches illustrées sur les cartes respectives :** 3 flèches signifient la priorité supérieure et 1 la priorité inférieure.

Phase 3 - Phase de transport

Les transports s'effectuent par la **voie de transport représentée sur la carte**, lorsque cela est possible :

- par voie **terrestre** :
- par **rail**:
- par voie **maritime ou fluviale** :
- par voie **aérienne :**

Chaque ville possède 3 des 4 voies de transport. Il n'est possible de transporter hors d'une ville **qu'en ayant une carte de transport d'une voie de transport permise par la ville.**

Les transports s'effectuent **simultanément**, par ordre de **priorité décroissant, indiqué sur les cartes de transport** :

- priorité haute :
- priorité moyenne :
- priorité basse :

Les transports de même priorité s'effectuent donc tous en même temps.

La carte de transport "**annulation**" annule l'étape de transport pour le joueur.

Les pions de transport ne peuvent être **que sur des villes**. Il ne peut jamais y en avoir deux sur la même ville :

- un **transport devant se terminer sur une ville déjà occupée est annulé**
- deux transports de même priorité pour une même ville libre sont tous les **deux annulés et la ville reste libre**.

Ces cas de figure peuvent **dramatiquement bouleverser la suite d'un plan de transport**, qui a été programmé pour l'ensemble du tour. Cela peut créer des situations imprévues et cocasses pouvant même aller jusqu'au blocage complet d'un transport sur l'ensemble des étapes de transport restantes.

Phase 4 - Phase de comptabilisation

Durant la phase de comptabilisation, chaque joueur gagne un nombre de points **égal au nombre de ville traversé** par son pion transport ce tour-ci. Si un pion de transport d'un joueur termine sur n'importe quelle ville de départ (en orange) autre que la sienne (dont il a la carte), **le joueur gagne 2 points** de plus. N'importe quel joueur peut, à tout moment, demander à l'arbitre de faire la monnaie de ses points. Le premier **joueur à obtenir 15 points ou plus** remporte la partie.

Sujet 1 - La pensée informatique

L'importance de développer la pensée computationnelle et des conseils peuvent être trouvés ici (en) : https://sphero.com/blogs/news/how-to-teach-computational-thinking-in-classroom. En outre, pour vous exercer davantage à la pensée computationnelle, vous pouvez vous référer à d'autres quêtes Unplugged, parmi lesquelles : ***Farm in the City, Good ways, Binary Counting, Peace Magic Grid etc.***

Sujet 2 - Naviguer dans l'espace

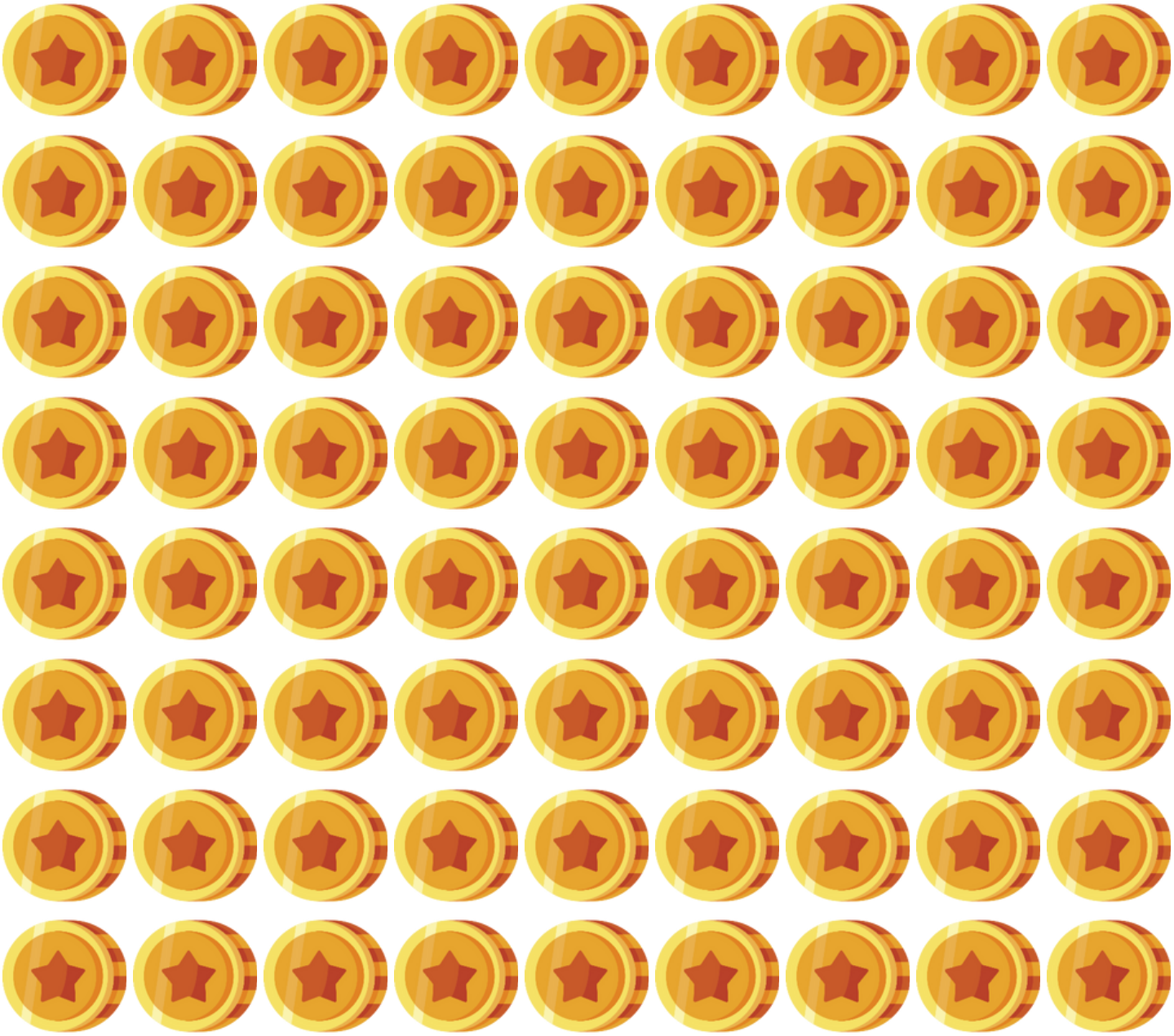
Pour vous entraîner davantage à l'orientation spatiale, vous pouvez vous référer à d'autres quêtes d'Unplugged, parmi lesquelles : ***Farm in the City, Good ways, Plastic continent, etc.*** L'importance de l'intelligence spatiale et d'autres conseils pour l'améliorer peuvent être trouvés ici (en) : https://www.parentingforbrain.com/visual-spatial-reasoning-skills-stem/

Sujet 3 - Coexistence

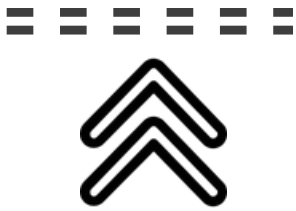
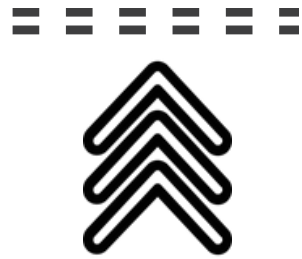
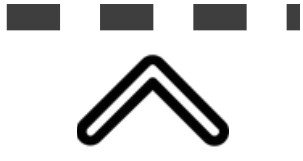
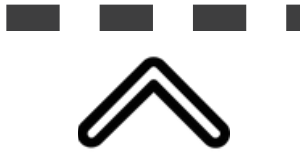
Vous pouvez combiner ce jeu avec la quête "Vivre en harmonie" disponible dans le jeu Unplugged. En outre, pour approfondir les pratiques de la pensée computationnelle, vous pouvez vous référer aux autres quêtes de Unplugged, parmi lesquelles : ***Likadislike, The Perfect City etc.***

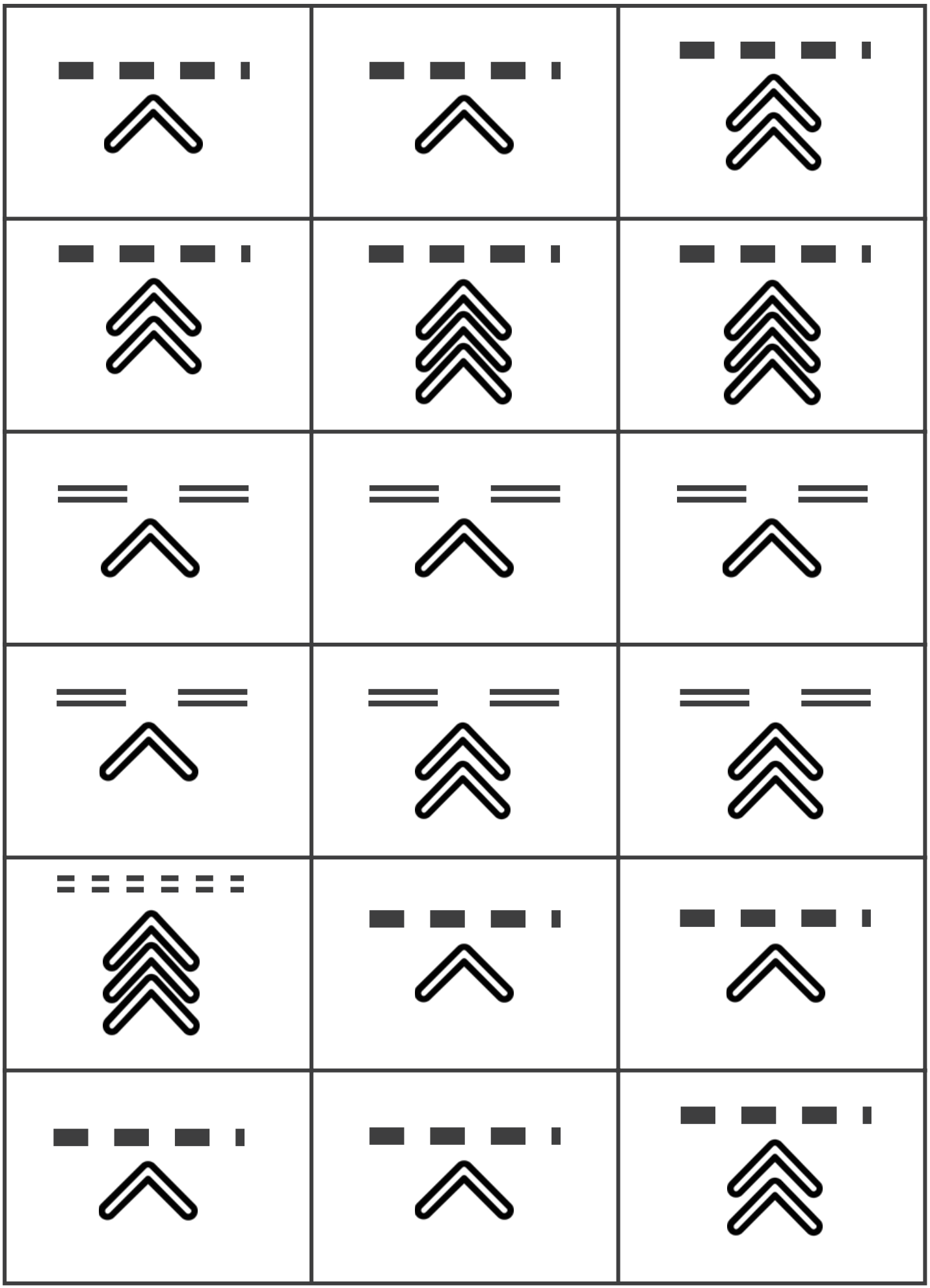
Matériaux

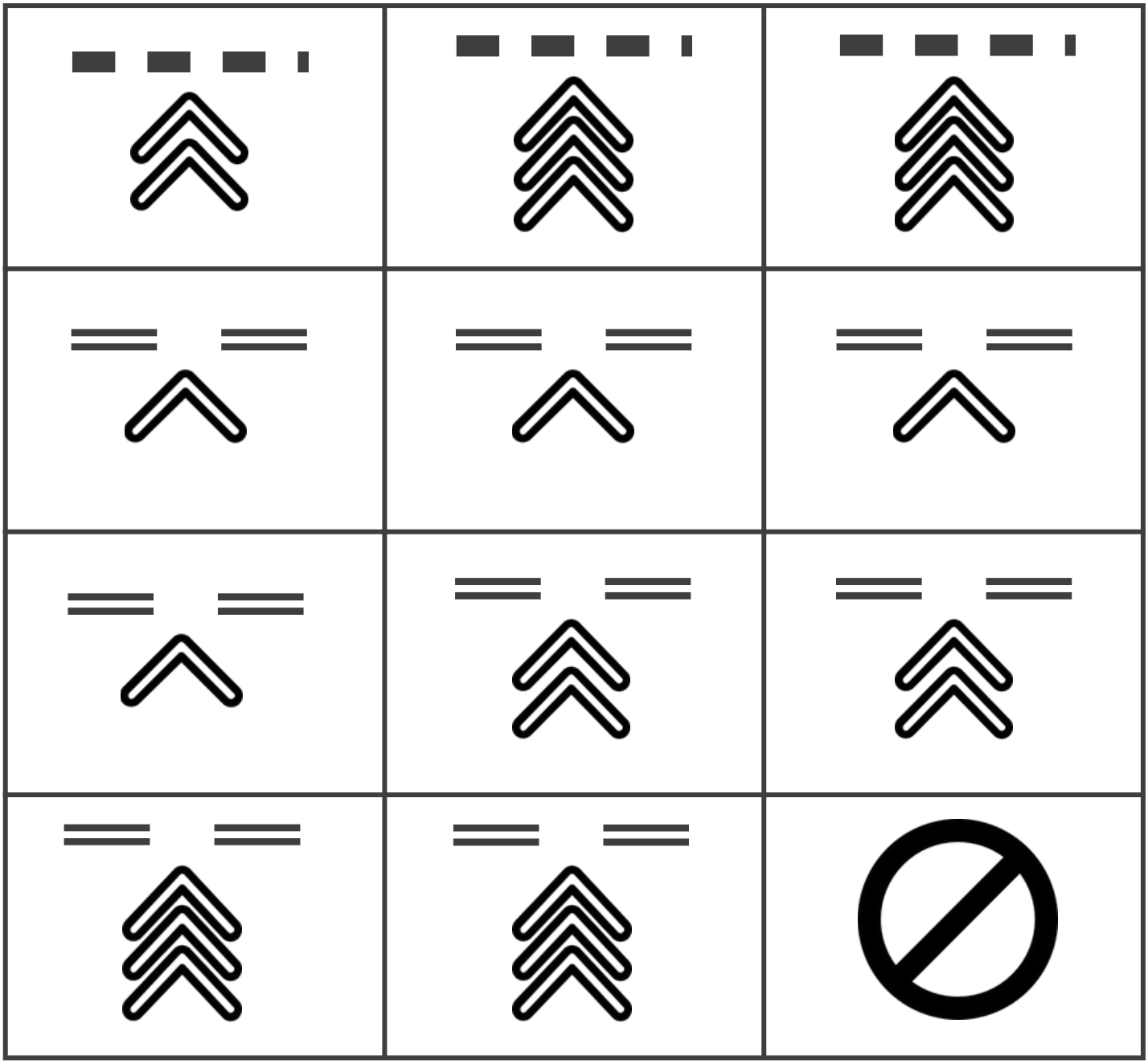
Points & Pièces



Helsinki
Stockholm
Riga
Vilnius
København
Dublin
Amsterdam
Berlin
Warszawa
Bruxelles
Kraków
Frankfurt am Main
Praha
Brest
Paris
München
Wien
Budapest
Milano
Venezia
Zagreb
București
Marseille
Sofiya
Barcelona
Madrid
Roma
Sevilla
Catania
Atnina
Lefkosía





Matériaux – Destinations

Amsterdam	**Athènes**	**Berlin**
Bucarest	**Dublin**	**Helsinki**
Cracovie	**Madrid**	**Milan**
Paris	**Rome**	**Vienne**