Classement d'objets Skylines dans les bases de données

Mickaël Martin Nevot, Lotfi Lakhal

Classement d'objets Skylines dans les bases de données

Mickaël Martin Nevot
mickael.martin-nevot@lis-lab.fr
Aix-Marseille Université LIS CRNS UMR 7020
Marseille, France

Lotfi Lakhal
lotfi.lakhal@lis-lab.fr
Aix-Marseille Université LIS CRNS UMR 7020
Marseille, France

ABSTRACT

L'analyse multicritère dans les bases de données a été activement étudiée, en particulier avec l'utilisation de l'opérateur Skyline. Pourtant, peu d'approches proposent un classement pertinent des points Pareto-optimal, ou Skyline, permettant d'ordonner les résultats à forte cardinalité. Nous proposons d'améliorer la méthode dp-idp, inspiré de tf-idf, une approche récente attribuant un score à chaque point du Skyline, en introduisant le concept de hiérarchie de dominance. Comme dp-idp ne garantit pas un classement distinctif, nous introduisons la méthode CoSky, de type TOPSIS, issue à la fois de la recherche d'information et de l'analyse multicritère. 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 à CoSky, nous introduisons l'algorithme DeepSky. La mise en œuvre des méthodes dp-idp et CoSky sont évaluées expérimentalement.

KEYWORDS

Analyse décisionnelle multicritère, Skyline, Recherche d'information, Classement, Pokémon

1 INTRODUCTION

L'opérateur Skyline ([5]), précédemment ensemble de Pareto et vecteurs maximaux ([4]), est capital dans l'analyse multicritère et a été largement étudié. Ses principales problématiques sont une forte cardinalité et une faible corrélation dans un jeu de données. Dans ces cas-là, il est souvent difficile d'extraire d'information significative car plus les points d'un Skyline sont nombreux plus ils peuvent avoir des intérêts proches et de faibles différences significatives. Un classement efficace permet d'y pallier, mais peu de telles approches ont été proposées bien qu'elles permettent de faciliter et prioriser la prise de décision, réduire la complexité de l'analyse, simplifier l'évaluation de compromis et proposer des solutions en fonction de préférences spécifiques ou d'objectifs contextuels.

L'approche dp-idp est certainement une des approches les plus récentes. Elle utilise la dominance de Pareto afin de déterminer un score pour chaque point d'un Skyline. dp-idp reprend l'idée du schéma de pondération tf-idf utilisé en recherche d'information [1].

Dans ce papier, 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 afin de classer efficacement les points d'un Skyline sans privilégier de dominance, avant de présenter l'algorithme DeepSky, un algorithme Skyline multiniveaux utilisant la méthode CoSky pour classer les top-k points de Skyline. Enfin, la mise en œuvre des méthodes dp-idp et CoSky sont présentées avec des évaluations expérimentales.

Ce papier est une révision approfondie du travail initial ([1]). Nos principaux nouveaux apports portent sur la spécification et l'implémentation des méthodes proposées : dp-idp avec hiérarchie de dominance et CoSky, leurs évaluations expérimentales confrontées à celle de l'algorithme de référence SkyIR-UBS et leurs commentaires, l'unification des préférences Skyline ainsi que la clarification, la précision et l'amélioration de la méthode CoSky.

2 CAS D'UTILISATION

Nous considérons ici un exemple appliqué au jeu vidéo Pokémon Showdown![2], et sa relation exemple `Pokémon` (cf. tableau 1).

Table 1: La relation `Pokémon`

RowId	Joueur[a]	Adversaire[b]	Rareté[c]	Durée[d]	Victoire[e]
1	121, 113, 103	121, 113, 121	5	20	70
2	065, 103, 065	065, 143, 065	4	60	50
3	121, 113, 121	065, 103, 065	5	30	60
4	121, 113, 080	065, 143, 065	1	80	60
5	121, 113, 128	121, 113, 121	5	90	40
6	065, 113, 143	065, 113, 143	9	30	50
7	065, 143, 065	121, 113, 143	7	80	60
8	065, 113, 143	065, 103, 065	9	90	30

[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.

Figure 1: Les Pokémon du cas d'utilisation[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.

[1] La recherche d'information (RI, ou IR pour *information retrieval*), englobe la recherche et la récupération de données ou d'informations pertinentes à partir de données non structurées, telles que des textes, des images, des vidéos ou des sons.

[2] Pokémon Showdown! est un jeu vidéo sur navigateur Web et PC *open source*. C'est un simulateur de combat de Pokémon (Pokémon ne prend pas de marque du pluriel, possiblement car il s'agit d'un terme issu du japonais) populaire (des millions d'utilisateurs mensuels, avec jusqu'à plus de 20000 simultanément) permettant de jouer à des combats de Pokémon en ligne animés.

40ème conférence sur la gestion des données (BDA), October 21-24, 2024, Orléans, France 2024. ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 DÉFINITIONS PRÉLIMINAIRES

3.1 Préférence et dominance

Soit r une relation avec des attributs $A_1, \ldots, A_m$. Une préférence Skyline sur A_j est une expression de l'une des deux formes suivantes : $Pref(A_i)$ = `MIN`, ou $Pref(A_i)$ = `MAX`. Une préférence Skyline décrit donc les situations préférables. 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 :

$$(\preceq_d, \prec_d) = \begin{cases} (\leq, <) \equiv Pref(A_j) = \texttt{MIN} \\ (\geq, >) \equiv Pref(A_j) = \texttt{MAX} \end{cases} \quad (1)$$

Exemple 3.1 - La relation `Pokémon` illustrée par le tableau 1 est typique pour l'utilisation de calcul de Skyline. L'attribut `Joueur` est la séquence de Pokémon jouée par le joueur, et `Adversaire`, celle de l'adversaire. Les critères déterminant 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 de `Victoire` de la séquence de Pokémon du joueur par rapport à celle de l'adversaire. `Rareté` et `Durée` sont des critères à minimiser, alors que `Victoire` est à maximiser (les préférences Skyline considérées sont donc mixtes). Pour l'exemple, nous nous limitons à trois Pokémon par séquence, qu'elle soit du joueur ou de l'adversaire, et afin de simplifier leurs usages dans leurs représentations nous les indiquons par des listes de numéros de Pokémon, et leur affectons une lettre. Ainsi, la liste $121, 113, 128(A)$ correspond à la séquence des trois Pokémon Staross, Leveinard et Tauros. Pour la lisibilité, nous arrondissons à un multiple de cinq `Durée` et `Victoire`.

3.2 Opérateur Skyline

Une syntaxe SQL de l'opérateur Skyline a été proposée pour exprimer des requêtes basées sur des préférences ([5]).

Dans l'exemple, la requête SQL avec opérateur Skyline est :

```
01 |  SELECT * FROM Pokémon
02 |  SKYLINE OF Rareté MIN, Durée MIN, Victoire MAX
```

Et, la requête associée sans opérateur Skyline est la suivante :

```
01 |  SELECT * FROM Pokémon AS P1
02 |  WHERE NOT EXISTS (
03 |      SELECT * FROM Pokémon AS P2
04 |      WHERE (P2.Rareté <= P1.Rareté
05 |        AND P2.Durée <= P1.Durée
06 |        AND P2.Victoire >= P1.Victoire)
07 |        AND (P2.Rareté < P1.Rareté
08 |         OR P2.Durée < P1.Durée
09 |         OR P2.Victoire > P1.Victoire));
```

Dans une représentation graphique, spécifiée par la dominance de Pareto, 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.

Exemple 3.2 - Avec notre exemple, 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 séquences 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.

4 CLASSEMENT DE SKYLINE

De nombreux travaux ont été consacrés à l'étude du classement de Skyline. Dans [6], une métrique appelée fréquence Skyline est proposée afin d'ordonner un Skyline en fonction des 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 ([17])[3] peuvent servir d'alternative aux requêtes Skyline. 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 ([2]). Une approche de classement de Skyline pour un Skycube ([11]) se concentrant sur les points de Skyline les plus chargés en information a aussi été proposée ([16]). 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 avantageux ([8]).

4.1 Méthode dp-idp

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. La dominance d'un point est inversement proportionnelle au nombre de points qui le domine ([15]), *i.e.* :

$$dp(p, sp) = \frac{1}{lm(p, sp)} \quad (2)$$

dp-idp prend en compte les positions relatives des points dominés pour les différencier en se concentrant sur les points qui sont peu dominés : *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 sont similaires par rapport 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 $p \in r \backslash S$ 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 :

$$idp(p) = \log \frac{|S|}{|\{sp \in S : sp \prec_d p\}|} \quad (3)$$

Afin de calculer la valeur 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. Ainsi, il est nécessaire de calculer la couche de

[3] top-k est une méthode de classement avec fonction d'évaluation.

minima (ou *layer of minima*)[4] $lm(p, sp)$ 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 :

$$Score(sp) = \sum_{p : sp \prec_d p} dp(p, sp) \cdot idp(p) \quad (4)$$

Les étapes de l'approche naïve permettant de classer un Skyline grâce à dp-idp sont :

- calcul des couches de *minima* des points du Skyline sp ;
- mise à jour de $Score(sp)$ 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.

Malheureusement, cette approche est peu efficace, et elle est notamment assez couteuse en temps en raison de calculs répétés. Elle est également dépourvue de toute notion de progressivité puisque nous devons d'abord ordonner le Skyline entier ([15]).

Pour ces raisons, une approche alternative plus efficace, SkyIR, a été proposée ([15]). L'algorithme a été décliné en fonction du modèle de priorité utilisé (*Round-robin*, priorité par nombre des points non encore traités ou priorité par borne supérieure, ou *Upper Bound (UBS)*). La configuration la plus avantageuse est presque toujours celle avec le système de priorité *UBS*, soit SkyIR-UBS.

Le principal défaut de SkyIR-UBS apparaît être la complexité induite par les calculs de toutes les couches de *minima* qu'il effectue. Nous proposons de perfectionner cette approche en ne prenant en compte, pour chaque point du Skyline, que ses dominés les plus proches, et plus aucun de ses dominés " indirects ". De la sorte, les $lm(p, sp)$ sont composés d'un sp et de la suite des points directement dominés de sp jusqu'à p.

4.2 Amélioration de dp-idp

La relation de dominance peut être vue 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 calcul bien plus rapide des couches de *minima*, le graphe étant élagué de ses arêtes inutiles lors de son parcours, et conduit, par conséquent, à une plus grande efficacité. Le graphe que nous proposons est un graphe orienté acyclique donnant une représentation d'un ensemble partiellement ordonné en établissant son graphe de couverture[5]. Un graphe orienté acyclique offre un ordre topologique qui peut donner une excellente représentation d'une hiérarchie d'un point de Skyline sp par rapport aux points qu'il domine.

[4]Couche de *minima* est un concept analogue à celui de couche de *maxima*, plus commun. Il s'agit de tous les points minimaux d'un ensemble donné. C'est la première couche, ou " frontière ", de points qui ne domine aucun autre de l'espace multidimensionnel.
[5]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.

Definition 4.1 (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*) est le graphe de couverture de l'ensemble ordonné $(D, \prec_d)$.

Exemple 4.1 - La figure 2 représente une hiérarchie de dominance ayant comme sommet le point du Skyline sp. L'ordre de dominance est illustré par les arrêtes entre sp et les points qu'il domine (p_1, p_2, p_3 et p_4).

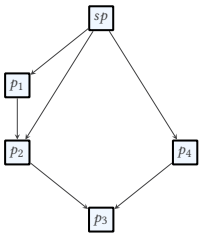

Figure 2: Exemple de graphe de hiérarchie de dominance

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 4.2 - Pour calculer la couche *minima* $lm(p3, sp)$, nous voyons sur la figure 2 qu'il y a deux chemins 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 $lm(p_3, sp) = 3$.

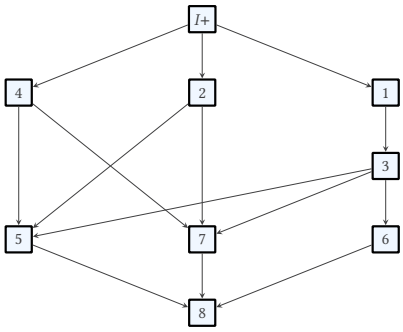

Figure 3: Graphe de hiérarchie de dominance de l'exemple

Exemple 4.3 - En considérant à nouveau la relation `Pokémon` (cf. tableau 1), 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 3 (I^+ étant le point idéal théorique, ou abstrait, qui domine tous les points du Skyline, et les points sont représentés par leur `RowId`).

En considérant à nouveau la relation Pokémon (cf. tableau 1), pour calculer le score de chaque point du Skyline, nous utilisons la formule 4. Ainsi, le classement par ordre décroissant des points du Skyline, représentés par leur RowId, est soit 1, 2, 4, soit 1, 4, 2.

La formule de calcul des idp a pour particularité que chaque point dominés par tous les points du Skyline aura un score de 0. En effet, ces points ne modifient pas le classement du Skyline car ils affectent tous les points du Skyline de la même manière ([15]).

Sur la base des résultats obtenus (cf. tableau 2), nous montrons 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.

Table 2: Calcul de score avec la méthode dp-idp améliorée

RowId des sp	RowId des p	$lm(p, sp)$	$Score(sp)$
1	3	$lm(3,1) = 2$	0.398
	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	$lm(8,4) = 3$	

4.2.1 Algorithme de la méthode améliorée. L'algorithme dp-idp avec hiérarchie de dominance est un algorithme qui fait appel à quatre sous-algorithmes : matriceDesDominants, grapheDeCouverture, lm et score_{dp-idp}. Il est décrit dans l'algorithme 1.

Algorithme 1 Algorithme dp-idp avec hiérarchie de dominance

```
Entrée :
  La relation r.
Sortie :
  Le tableau des scores de dp-idp score.
  //Appel au quatre sous-algorithmes
  m_≺d, S_\D+lm := matriceDesDominants(r);
  m_≺d, S_\D+lm, S_≺d C, idpC :=
  grapheDeCouverture(m_≺d, S_\D+lm);
  S_\D+lm := lm(m_≺d, S_\D+lm, S_≺d C);
  retourner  score_dp-idp(S_\D+lm, idpC);
```

Exemple 4.4 - Le tableau 3 représente la relation Pokémon qui, pour des raisons de commodité, ne conserve aucun attribut ou commentaire, seulement le RowId des tuples.

Le sous-algorithme matriceDesDominants, décrit dans l'algorithme 2, génère, depuis la relation r, un tableau à deux dimensions carré, ou matrice, $m_{\prec_d}$, de la cardinalité de r. $m_{\prec_d}$ indique les dominances, *i.e.* en colonne est signalé le RowId du tuple dominant le tuple représenté en ligne par son RowId. Le sous-algorithme est donné dans le cas où toutes les préférences Skyline sont MIN, mais il est aisément transposable pour n'importe quelle combinaisons de préférences Skyline. Le Skyline $S_{\setminus \mathcal{D}+lm}$ est aussi calculé à cette occasion, et il est retourné sans dimension et destiné à recevoir les couches *minima*.

Table 3: La relation Pokémon_1 simplifiée

RowId	Rareté	Durée	Victoire
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

Algorithme 2 Algorithme matriceDesDominants ($\mathcal{O}(|r|^2 \cdot |\mathcal{D}|)$)

```
Entrée :
  La relation r.
Sortie :
  Le tableau à deux dimensions carré indiquant les dominances m_≺d.
  Le Skyline sans dimension destiné à recevoir les lm(sp, p) des
  points dominés S_\D+lm.
  soit D := {d_1, ..., d_n} : l'ensemble des dimensions de r.
  soit m_≺d : un tableau de |r| tableaux de |r| booléens faux
  soit idpC : une relation sans dimension de |r| tuples avec les
  mêmes RowId que ceux de r
  pour i := 0, ..., |r| − 1 faire
    pour j := 0, ..., |r| − 1 faire
      si i ≠ j alors
        soit t_i : t_i ∈ r, t_i[RowId] = i;
        soit t_j : t_j ∈ r, t_j[RowId] = j;
        soit sup := vrai;
        pour tout d_k ∈ D faire
          //Calcul de dominance
          //Comparateur > si ∀{d_1, ..., d_n} ∈ D, Pref(d_k) = MIN
          si t_j[d_k] > t_i[d_k] alors
            sup := faux; //t_j ⊀_d t_i
            exit pour
          fin si
        fin pour
        si sup alors
          soit t'_i : t'_i ∈ S_\D, t'_i[RowId] = i;
          //Ajout de l'arête au graphe
          m_≺d[i][j] := vrai; //t_j ≺_d t_i
          S_\D := S_\D \ t'_i; //t_i ∉ S
        fin si
      fin si
    fin pour
  fin pour
  soit S_\D+lm : une relation de |r| dimensions de S_\D+lm tuples
  (0, ..., 0) avec les mêmes RowId que ceux de S_\D+lm
  retourner  m_≺d, S_\D+lm;
```

Exemple 4.5 - En considérant à nouveau la relation Pokémon (cf. tableau 3), résultant de l'algorithme 2, le tableau 4 montre la matrice des dominants $m_{\prec_d}$ et le tableau 5 montre le Skyline, sans dimension et destiné à recevoir les couches *minima* $S_{\setminus \mathcal{D}+lm}$. Avec le tableau 4, nous voyons que le tuple de RowId 6 est dominé à la fois par celui de de RowId 1 et celui de RowId 3. Le tableau 5 nous confirme que le Skyline est composé des tuples de RowId 1, 2 et 4, visibles en ligne, alors que les RowId de l'ensemble de tuple de la relation sont donnés en colonnes.

Le sous-algorithme grapheDeCouverture, décrit dans l'algorithme 3, a pour objectif de mettre à jour la matrice des dominants $m_{\prec_d}$ en ne considérant plus que le graphe de couverture. Le Skyline sans dimension $S_{\setminus \mathcal{D}+lm}$ est aussi mis à jour afin d'être prêt à recevoir les

Table 4: La matrice des dominants $m_{\prec_d}$

RowId	1	2	3	4	5	6	7	8
1								
2								
3	✓							
4								
5	✓	✓	✓	✓				
6	✓		✓					
7	✓	✓	✓	✓				
8	✓	✓	✓	✓	✓	✓	✓	

Table 5: Le Skyline sans dimension voué à avoir les *lm* $S_{\setminus \mathcal{D}+lm}$

RowId	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0

couches *minima*, chacune des couches à calculer étant indiquée par la valeur 1. A des fins d'optimisation, les cardinalités de dominance des points du Skyline $S_{\prec_d}C$ ainsi que les cardinalités utiles à la mesure des idp, *idpC*, sont également calculés durant le parcours du graphe de couverture.

Algorithme 3 Algorithme grapheDeCouverture ($O(|r|^3/2)$)

```
Entrée :
  Le tableau à deux dimensions carré indiquant les dominances m_{≺_d}.
  Le Skyline sans dimension destiné à recevoir les lm(sp,p) des
  points dominés S_{\D+lm}.
Sortie :
  Le tableau à deux dimensions carré indiquant les dominances du
  graphe de couverture m_{≺_d}.
  Le Skyline sans dimension prêt à recevoir les lm(sp,p) des points
  dominés S_{\D+lm}.
  Les cardinalités de dominance des points de S S_{≺_d}C.
  Les cardinalités des sp ∈ S, sp ≺_d p  idpC.
  soit S_{≺_d}C : un tableau de |S_{\D+lm}| entiers 0
  soit idpC : un tableau de |m_{≺_d}| entiers 0
  pour i := 0,...,|m_{≺_d}| − 1 faire
    si t_i, t[RowId] = i, t_i ∉ S_{\D+lm} alors
      pour j := 0,...,|m_{≺_d}| − 1 faire
        si m_{≺_d}[i][j] alors
          si t_j, t[RowId] = j, t_j ∈ S_{\D+lm} alors
            //Marquage des dominances
            t_j[i] := 1
            //Mise à jour des cardinalités de dominance
            S_{≺_d}C[j] := S_{≺_d}C[j] + 1;
            idpC[i] := idpC[i] + 1;
          sinon
            pour k = 0,...,|m_{≺_d}| − 1 faire
              si m_{≺_d}[j][k] alors
                //Suppression de l'arête inutile du graphe de
                couverture
                m_{≺_d}[j][k] := faux;
              fin si
            fin pour
          fin si
        fin si
      fin pour
    fin si
  fin pour
  retourner  m_{≺_d}, S_{\D+lm}, S_{≺_d}C, idpC;
```

Exemple 4.6 - En considérant à nouveau la relation Pokémon (cf. tableau 3), résultant de l'algorithme 6, le tableau 6 montre la matrice des dominants $m_{\prec_d}$ selon le graphe de couverture et le tableau 7 montre le Skyline, sans dimension et prêt à recevoir les couches *minima* $S_{\setminus \mathcal{D}+lm}$, fusionné, en colonne, avec les cardinalités de dominance des points du Skyline $S_{\prec_d}C$ et, en ligne, avec les cardinalités utiles à la mesure des idp, *idpC*. Cette présentation nous apparaît plus pratique et a pour objectif d'améliorer aussi la compréhension. Avec le tableau 6 nous retrouvons le graphe de hiérarchie de dominance de Pokémon Showdown! (cf. figure 3). Et, avec le tableau 7, nous voyons (avec les valeurs 1) toutes les dominances de chaque point du Skyline (dont les RowId sont en ligne). De même, nous voyons le nombre de points dominés par chaque point du Skyline dans la colonne $S_{\prec_d}C$, et le nombre de points du Skyline dominant chaque point de la relation (dont les RowId sont en colonne) avec la ligne unique du tableau *idpC*.

Table 6: $m_{\prec_d}$ d'après le graphe de couverture

RowId	1	2	3	4	5	6	7	8
1								
2								
3	✓							
4								
5		✓	✓	✓				
6			✓					
7		✓	✓	✓				
8					✓	✓	✓	

Table 7: $S_{\setminus \mathcal{D}+lm}$ prêt à recevoir les *lm* avec $S_{\prec_d}C$ et *idpC*

RowId	1	2	3	4	5	6	7	8	$S_{\prec_d}C$
1	0	0	1	0	1	1	1	1	5
2	0	0	0	0	1	0	1	1	3
4	0	0	0	0	1	0	1	1	3
idpC	0	0	1	0	3	1	3	3	/

Le sous-algorithme lm, décrit dans l'algorithme 4, calcule les couches *minima* pour chaque point du Skyline selon le graphe de couverture indiqué dans la matrice des dominants $m_{\prec_d}$. Les résultats sont enregistrés dans le Skyline sans dimension $S_{\setminus \mathcal{D}+lm}$. L'utilisation d'une fonction récursive serait toute indiquée pour ce sous-algorithme, mais son usage le rendrait moins efficace. Aussi, nous préférons ne présenter que la version sans récursivité.

Exemple 4.7 - En considérant à nouveau la relation Pokémon (cf. tableau 3), résultant de l'algorithme 4, le tableau 8 montre le Skyline sans dimension mais avec les couches *minima* $S_{\setminus \mathcal{D}+lm}$. Nous rappelons que les RowId des tuples du Skyline sont affichés en ligne, alors que tous les RowId de tous les tuples de la relation le sont en colonne. De la sorte, nous voyons que $lm(3,1) = 2$, $lm(8,1) = 4$ ou encore que $lm(2,5) = 2$.

Algorithme 4 Algorithme lm ($O(|r| \cdot |S|)$)

```
Entrée :
  Le tableau à deux dimensions carré indiquant les dominances m_{≺_d}.
  Le Skyline sans dimension prêt à recevoir les lm(sp,p) des points
  dominés S_{\D+lm}.
  Les cardinalités de dominance des points de S S_{≺_d}C.
Sortie :
  Le Skyline S avec les lm(sp,p) des points dominés S_{lm}.
  pour tout t ∈ S_{\D+lm} faire
     soit i := t[RowId];
     couche := i
     prof := 2
     tant que S_{≺_d}C[i] > 0 faire
        couche_{+1} := couche
        pour j = 0,...,|m_{≺_d}| − 1 faire
           si m_{≺_d}[j][couche] ∧ t[j] > 0 alors
              si t[j] = 1 ∨ t[j] ≠ 1 ∧ t[j] > prof alors
                 //Ajustement de la profondeur de lm(t,t_j), t_j ∈ r
                 t[j] := t[j] + prof;
              fin si
              S_{≺_d}C[i] := S_{≺_d}C[i] − 1; //Dominance traitée
              couche_{+1} := j
              si S_{≺_d}C[i] = 0 alors
                 //Le traitement du point du Skyline s'arrête lorsque
                 toutes ses dominances sont traitées
                 exit pour
              fin si
           fin si
           prof := prof + 1
           si couche_{+1} = couche alors
              exit pour//Il n'y a plus de couche à traiter
           fin si
           couche := couche_{+1}
        fin pour
     fin tant que
  fin pour
  retourner S_{\D+lm};
```

Table 8: Le Skyline sans dimension avec les *lm* $S_{\setminus \mathcal{D}+lm}$

RowId	1	2	3	4	5	6	7	8
1	0	0	2	0	3	3	3	4
2	0	0	0	0	2	0	2	3
4	0	0	0	0	2	0	2	3

Le sous-algorithme score_{dp-idp}, décrit dans l'algorithme 5, calcule les scores des points du Skyline selon la méthode dp-idp. Pour cela, il a besoin des couches *minima* enregistrés dans le Skyline sans dimension $S_{\setminus \mathcal{D}+lm}$, utiles pour le calcul de dp, et il a aussi besoin, pour chaque point de la relation, du nombre de points du Skyline qui le domine, *idpC*, utiles pour le calcul de idp.

Algorithme 5 Algorithme score_{dp-idp} ($O(|S| \cdot |\mathcal{D}|)$)

```
Entrée :
  Le Skyline sans dimension avec les lm(sp,p) des points dominés
  S_{\D+lm}.
  Les cardinalités des sp ∈ S, sp ≺_d p  idpC.
Sortie :
  Le tableau des scores de dp-idp score.
  soit score : un tableau associatif de |S_{\D+lm}| entiers 0
  pour tout t ∈ S_{\D+lm} faire
     soit i := t[RowId];
     pour j = 0,...,|t| − 1 faire
        si t[j] > 0 alors
           score[i] := score[i] + 1/t[j] × log(|S_{\D+lm}|/idpC[j]);
           //cf. formule 4
        fin si
     fin pour
  fin pour
  retourner score;
```

Exemple 4.8 - En considérant à nouveau la relation `Pokémon` (cf. tableau 3), résultant de l'algorithme 5, le tableau 9 montre les scores respectifs des tuples de `RowId` 1, 2 et 4, qui compose le Skyline.

Table 9: Le tableau des scores de dp-idp amélioré

RowId	1	2	4
score	0.398	0	0

4.3 Méthode CoSky

Afin de proposer une solution de classement de Skyline qui permette 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 ou de fonction mathématique gourmande en temps comme le logarithme[6]. C'est, à notre connaissance, la première méthode de type TOPSIS[7] ([3, 10]) appliqué à ce type de classement. 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.

Ce calcul est présenté en détails dans cette sous-section, et sa préparation ainsi que des remarques générales sont données en annexe. Pour chaque étape, 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).

4.3.1 I. Normalisation par la somme. Le Skyline est normalisé par la somme. Cette méthode garantit que toutes les valeurs normalisées sont comprises entre -1 et 1, et que leur somme est égale à 1. Cela est utile lors de la représentation de données où la contribution relative de chaque valeur par rapport à l'ensemble est importante. Cette normalisation 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.

Il est important de noter qu'il est préférable de ramener, préalablement, les valeurs non normalisées des attributs au dessus de 0 afin d'éviter que les valeurs négatives soient confondues avec leurs opposées. En outre, de la sorte, chaque valeur d'attribut de chaque point du Skyline est convertie en une valeur sur une échelle allant de 0 à 1.

Soit S_N l'ensemble des points de Skyline normalisés, ou Skyline normalisé, et le tuple $u_i = (u_i[A_1], u_i[A_2], \ldots, u_i[A_m]) \in S_N$, alors nous avons :

$$u_i[A_j] = \frac{t_i[A_j]}{\sum_{i'=1}^{n} t_{i'}[A_j]}, \forall t_i \in S \tag{5}$$

[6]Le logarithme est une fonction transcendantale significativement plus couteuse, notamment en raison des optimisations matérielles fréquemment disponibles pour l'addition et la multiplication, que les opérations algébriques standards.

[7]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.

Cette méthode nécessite tout de même une vigilance afin d'éviter une division par 0, sur la somme des valeurs non normalisées des attributs : $\sum_{i'=1}^{n} t_{i'}[A_j] \neq 0$.

4.3.2 II. 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. Une méthode basée sur le concept d'entropie a ainsi été proposée dans des problèmes multicritères ([9, 12]). Cette méthode s'adapte bien à un contexte Skyline, pour lequel nous cherchons à différencier les valeurs des attributs par un classement de Skyline afin d'avoir une meilleure prise de décision. Cependant, elle a aussi ses limites, notamment en ce qui concerne le calcul de l'entropie qui nécessite l'usage d'une fonction logarithmique, gourmande en temps.

Nous préférons présenter une autre mesure, l'indice de Gini[8], 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 :

$$Gini(A_j) = 1 - \sum_{i=1}^{n} u_i[A_j]^2 \tag{6}$$

Nous appelons W le poids de l'attribut A_j. Le décideur peut spécifier directement $(W(A_1), W(A_2), \ldots, W(A_m))$ de sorte que $\texttt{SUM}(W(A_1), W(A_2), \ldots, W(A_m)) = 1$. Le poids d'un attribut est alors son importance, et est donné par la formule suivante :

$$W(A_j) = \frac{Gini(A_j)}{\sum_{j'=1}^{m} Gini(A_{j'})} \tag{7}$$

Soit S_P Skyline pondéré, ou l'ensemble des points de Skyline après pondération, et le tuple $v_i = (v_i[A_1], v_i[A_2], \ldots, v_i[A_m]) \in S_P$, alors nous avons :

$$v_i[A_j] = W(A_j) \times u_i[A_j], \forall u_i \in S_N \tag{8}$$

4.3.3 III. 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 Skyline.

Ainsi, soit $I^+ = (I^+[A_1], I^+[A_2], \ldots, I^+[A_m])$, alors nous avons :

$$I^+[A_j] = \begin{cases} \texttt{MAX}(v_i[A_j]) \equiv Pref(A_j) = \texttt{MAX} \\ \texttt{MIN}(v_i[A_j]) \equiv Pref(A_j) = \texttt{MIN} \end{cases} \tag{9}$$

Exemple 4.9 - Avec la relation `Pokémon` (cf. tableau 10), rechercher la séquence idéale 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 d'`Échec` le plus bas possible.

4.3.4 IV. Scores avec le cosinus de Salton. Cette étape vise à déterminer le score d'un point de Skyline avec le cosinus de Salton [9]. Pour cela, le cosinus de l'angle entre le point idéal et le point du Skyline est calculé. 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 valeurs de scores des points du Skyline, le tuple $v_i = (v_i[A_1], v_i[A_2], \ldots, v_i[A_m]) \in S_P$ et le point idéal $I^+ = (I^+[A_1], I^+[A_2], \ldots, I^+[A_m])$, alors nous avons :

$$s_i = S_c(v_i, I^+) := cos(\theta) = \frac{v_i \cdot I^+}{||v_i|| \cdot ||I^+||} \tag{10}$$

$$s_i = \frac{\sum_{j=1}^{m} v_i[A_j] \cdot I^+[A_j]}{\sqrt{\sum_{j=1}^{m} v_i[A_j]^2} \cdot \sqrt{\sum_{j=1}^{m} I^+[A_j]^2}}, \forall s_i \in S_{Score} \tag{11}$$

Une conséquence est que $s_i = 1$ si et seulement si le point du Skyline est considéré comme le plus intéressant, et $s_i = 0$ si et seulement s'il est considéré comme 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 anti-idéal alors, $\forall v_i \in S_P$, si nous considérons $I^- = (I^-[A_1], I^-[A_2], \ldots, I^-[A_m])$, nous avons :

$$I^-[A_j] = \begin{cases} \texttt{MAX}(v_i[A_j]) \equiv Pref(A_j) = \texttt{MIN} \\ \texttt{MIN}(v_i[A_j]) \equiv Pref(A_j) = \texttt{MAX} \end{cases} \tag{12}$$

4.3.5 V. Classement des résultats. La dernière étape a pour objectif d'ordonner les points du Skyline en fonction des scores de manière décroissante.

4.3.6 CoSky en SQL. Il est à noter que de la méthode CoSky est complètement intégrable aux systèmes de gestion de bases de données (SGBD) relationnelles. Autrement dit, il est toujours possible d'utiliser une requête SQL pour la mettre en œuvre.

Table 10: La relation `Pokémon`$_2$

RowId	Rareté	Durée	Échec
1	5	20	1/70
2	4	60	1/50
3	5	30	1/60
4	1	80	1/60
5	5	90	1/40
6	9	30	1/50
7	7	80	1/40
8	9	90	1/30

Exemple 4.10 - Il est possible d'appliquer les différentes étapes de calcul de la méthode CoSky à la relation `Pokémon` (cf. tableau 10), avec préférences Skyline unifiées (`MIN`, `MIN`, `MIN`) en utilisant la requête SQL suivante[10] :

[8] L'indice (ou coefficient) de Gini est une mesure statistique utilisée pour évaluer l'inégalité d'une variable par rapport à une population donnée. Initialement, et toujours principalement, il est employé pour mesurer le degré d'inégalité des revenus d'un pays. Il varie entre 0 (égalité parfaite) et 1 (inégalité totale), et l'inégalité est d'autant plus forte que l'indice est élevé.

[9] Le cosinus de Salton, ou mesure de similarité cosinus, ou encore de cosinus de similitude, mesure, entre 0 à 1, la similarité entre vecteurs. Il permet de représenter une information par un vecteur et son importance par un angle dans un espace vectoriel. Il permet de calculer ainsi, classiquement, la pertinence d'une page Web pour une recherche donnée.

[10] Si le Skyline n'a qu'un seul point, son classement est inutile, ou doit être fait en encadrant de `COALESCE(NULLIF(...,0),1)` chaque dénominateur de la requête.

```
01 |  WITH S AS (SELECT * FROM Pokémon
02 |      SKYLINE OF Rareté MIN, Durée MIN, Échec MIN
03 |  ), SN AS (SELECT RowId,
04 |         Rareté / TRare AS NRare,
05 |         Durée / TDurée AS NDurée,
06 |         Échec / TÉchec AS NÉchec
07 |  FROM S, (SELECT SUM(Rareté) AS TRare,
08 |    SUM(Durée) AS TDurée,
09 |    SUM(Échec) AS TÉchec FROM S) AS ST
10 |  ), SGini AS (SELECT
11 |    1 - SUM(NRare * NRare) AS GRare,
12 |    1 - SUM(NDurée * NDurée) AS GDurée,
13 |    1 - SUM(NÉchec * NÉchec) AS GÉchec FROM SN
14 |  ), SW AS (SELECT
15 |    GRare / (GRare + GDurée + GÉchec) AS WRare,
16 |    GDurée / (GRare + GDurée + GÉchec) AS WDurée,
17 |    GÉchec / (GRare + GDurée + GÉchec) AS WÉchec
18 |    FROM SGini
19 |  ), SP AS (SELECT RowId,
20 |    WRare * NRare AS PRare,
21 |    WDurée * NDurée AS PDurée,
22 |    WÉchec * NÉchec AS PÉchec FROM SN, SW
23 |  ), Idéal AS (SELECT MIN(PRare) AS IRare,
24 |    MIN(PDurée) AS IDurée,
25 |    MAX(PÉchec) AS IÉchec FROM SP
26 |  ), SScore AS (SELECT RowId,
27 |     (IRare * PRare + IDurée * PDurée +
28 |      IÉchec * PÉchec) /
29 |     (SQRT(PRare * PRare + PDurée *
30 |      PDurée + PÉchec * PÉchec) *
31 |      SQRT(IRare * IRare + IDurée *
32 |      IDurée + IÉchec * IÉchec)) AS Score
33 |    FROM Idéal, SP)
34 |  SELECT P.RowId AS RowId, Rareté, Durée, Échec,
35 |         ROUND(Score, 3) AS Score
36 |  FROM S P INNER JOIN SScore rs
37 |    ON P.RowId = rs.RowId
38 |  ORDER BY Score DESC;
```

Les résultats obtenus sont donnés au tableau 11.

Table 11: Classement de Skyline avec CoSky

RowId	Rareté	Durée	Échec	Score
2	4	60	1/50	0.909
4	1	80	1/60	0.847
1	5	20	1/70	0.774

Nous voyons donc que, contrairement à dp-idp, 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. La méthode CoSky permet donc bien d'avoir un classement de Skyline complet. En outre, avec dp-idp, l'importance d'un point est inversement proportionnelle au nombre de points de Skyline qui le dominent, ce qui peut être contestable suivant les applications. CoSky permet de mesurer un écartement par rapport à un idéal, ce qui constitue une autre vision au moins aussi pertinente. Dans l'exemple, non seulement les points de Skyline de `RowId` 2 et 4 sont différenciés, mais en plus le classement n'est pas le même qu'avec la méthode dp-idp.

4.3.7 Algorithme de CoSky. Bien que l'intégrabilité de CoSky aux SGBD relationnelles soit une propriété avantageuse de la méthode, il peut arriver qu'une implémentation algorithmique soit préférable. En effet, dans les cas d'une cardinalité dimensionnelle importante ou volatile, ou encore d'une volonté d'intégration simplifiée au sein d'un ensemble algorithmique plus vaste, la généricité et l'adaptabilité offerte par une solution algorithmique peut être préférable.

Dans cette optique, nous proposons l'algorithme naïf[11] 6.

L'algorithme CoSky ne calcule pas, à proprement parler, le Skyline. Il est donc nécessaire de faire appel à une solution externe pour cela. Assez classiquement, notre choix s'est porté sur l'algorithme branch-and-bound skyline (BBS) ([13]) pour son efficacité.

L'algorithme CoSky est, bien sûr, très proche de l'implémentation de CoSky en SQL. Les étapes de la méthodes sont respectées bien que, pour des raisons d'optimisation, et notamment le plus souvent de mutualisation de traitements ou d'itérations, elles peuvent parfois être scindées ou voir leur ordre être bouleversé.

Afin d'aider à rendre les étapes algorithmiques plus aisées à comprendre, nous les avons annoté d'exemples tirés directement de l'application de la méthode CoSky à la relation `Pokémon` (cf. tableau 10), avec préférences Skyline unifiées (`MIN`, `MIN`, `MIN`).

4.4 Méthode top-k

Pour cette dernière méthode, nous utilisons le principe de Skyline multiniveaux ([14]) 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 :

- si $Card(S_0(r)) > k$: Q_k ne renvoie que k points de $S_0(r)$;
- si $Card(S_0(r)) = k$: Q_k retourne le Skyline entier (*i.e.* tous les points de $S_0(r)$) ;
- si $Card(S_0(r)) < k$: 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 $(r \backslash S_0(r))$ sont retournés, mais aussi potentiellement certains points de $S_2(r)$ de $(r \backslash (S_0(r) \cup S_1(r))$, de $S_3(r)$... tant que le nombre cumulé de résultats retournés est inférieur à k.

4.4.1 DeepSky. L'algorithme DeepSky (cf. algorithme 7) 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 4.11 - Avec la relation `Pokémon` (cf. tableau 10), et $k = 4$, l'algorithme DeepSky (cf. algorithme 7) 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.

5 DISCUSSION

Le classement de Skyline par CoSky est avantageux, aussi bien par sa justesse que par son efficacité.

Le cosinus de Salton est une méthode puissante, flexible ainsi que simple à calculer et à interpréter. Il est également invariant aux transformations linéaires des vecteurs (comme la mise à l'échelle

[11] L'algorithme CoSky a été optimisé, mais il n'emploie pas de statistiques sur les données, ne tire pas partie du parallélisme ou de la gestion de caches, etc.

Algorithme 6 Algorithme CoSky ($O(|S| \cdot |\mathcal{D}|)$)

```
Entrée :
  La relation r.
Sortie :
  Le tableau des scores CoSky score.
  soit S : le Skyline de r //Calculé avec BBS
  soit 𝒟 = {d_1, ..., d_n} : l'ensemble des dimensions de r.
  soit sum𝒟_S : un tableau associatif de |𝒟| entiers 0
  soit S_N : la future relation S normalisée
  soit sum𝒟²_{S_N} : un tableau associatif de |𝒟| entiers 0
  soit gini : nouveau tableau associatif de |𝒟| entiers
  soit sum_gini := 0;
  soit S_NP : la future relation S normalisée pondérée
  soit sum𝒟²_{S_NP} : un tableau associatif de |𝒟| entiers 0
  soit ideal : un tableau associatif de |𝒟| entiers 1
  soit sum²_ideal := 0;
  soit sqrt_{sum²_ideal} : un entier positif
  soit score : un tableau associatif de |S| entiers 0
  pour tout t ∈ S faire
    pour tout d_j ∈ 𝒟 faire
      sum𝒟_S[d_j] := sum𝒟_S[d_j] + t[d_j]; //i.e. SUM(Durée)
    fin pour
  fin pour
  pour tout t ∈ S_N faire
    soit u : u ∈ S, u[RowId] = t[RowId];
    pour tout d_j ∈ 𝒟 faire
      t[d_j] := u[d_j]/sum𝒟_S[d_j]; //i.e. Durée / TDurée
      sum𝒟²_{S_N}[d_j] := sum𝒟²_{S_N}[d_j] + t[d_j]²;
      //i.e. SUM(NDurée * NDurée)
    fin pour
  fin pour
  pour tout d_j ∈ 𝒟 faire
    gini[d_j] := 1 − sum𝒟²_{S_N}[d_j];
    //i.e. 1 - SUM(NDurée * NDurée)
    sum_gini := sum_gini + gini[d_j];
    //i.e. GRareté + GDurée + GÉchec
  fin pour
  pour tout t ∈ S_NP faire
    soit i := t[RowId];
    soit u : u ∈ S_N, u[RowId] = i;
    pour tout d_j ∈ 𝒟 faire
      t[d_j] := gini[d_j]/sum_gini × u[d_j];
      //i.e. GDurée / (GRareté + GDurée + GÉchec)...
      //... AS WRareté et WDurée * NDurée
      sum𝒟²_{S_NP}[i] := sum𝒟²_{S_NP}[i] + t[d_j]²;
      //i.e. PRareté * PRareté + ... + PÉchec * PÉchec
      si t[d_j] < ideal[d_j] alors
        ideal[d_j] := t[d_j]; //i.e. MIN(Durée)
      fin si
    fin pour
  fin pour
  pour tout d_j ∈ 𝒟 faire
    sum²_ideal := sum²_ideal + ideal[d_j]²;
    //i.e. IRareté * IRareté + ... + IÉchec * IÉchec
  fin pour
  sqrt_{sum²_ideal} := √(sum²_ideal);
  pour tout t ∈ S_NP faire
    soit i := t[RowId];
    soit score_numerator := 0;
    pour tout d_j ∈ 𝒟 faire
      score_numerateur := score_numerateur + ideal[d_j] × t[d_j];
      //i.e. IRareté * PRareté + ... + IÉchec * PÉchec
    fin pour
    score[i] := score_numerateur/(√(sum𝒟²_{S_NP}[i]) × sqrt_{sum²_ideal});
    //cf. formule 11
  fin pour
  retourner score;
```

des valeurs) et est tout indiqué pour comparer des vecteurs dans des espaces de grande dimension, ce qui nous intéresse tout particulièrement en analyse de données. En revanche, il est sensible aux vecteurs nuls et ne prend pas en compte la magnitude des vecteurs, seulement leur direction. Ainsi, bien que rare, un point de Skyline composant un vecteur superposant le vecteur formé avec le point idéal, bien qu'il soit d'une amplitude plus forte, sera considéré, à

Algorithme 7 Algorithme DeepSky ($O(k)$)

```
Entrée :
  La relation r.
  Le nombre k.
Sortie :
  Les top-k tuples/points avec les meilleurs scores top_k.
  soit top_k := ∅;
  soit tot := 0; //Nombre total de résultats calculés
  soit r_l := r; //Niveau courant
  tant que tot < k ∨ r_l = ∅ faire
    S := CoSky(r_l);
    tot := tot + |S|;
    si tot ≤ k alors
      top_k := top_k ∪ S;
      r_l := r_l \ S;
    sinon
      soit S_trunc : les k premiers points de S
      top_k := top_k ∪ S_trunc;
      retourner top_k
    fin si
  fin tant que
  retourner top_k
```

tord, comme optimal. De même, plusieurs points peuvent avoir avoir la même mesure de similarité.

Nous avons considéré qu'un point idéal est optimal sur l'ensemble des dimensions. Nous pourrions comparer, plus finement, avec un point dominant tous les autres.

D'autres tentatives de tri de l'ensemble des points de Skyline ont été proposées dans la littérature avec l'objectif de maîtriser la taille du résultat. Parmi ces méthodes, celle utilisant la " minimisation de regret " ([7]) pourrait aussi être comparée, ne serait-ce qu'expérimentalement, avec CoSky.

6 ÉVALUATIONS EXPÉRIMENTALES

Les expérimentations ont été réalisées sur une machine avec Intel(R) Xeon(R) W-11955M CPU @ 2.60GHz 2.61 GHz, avec 32Gb de mémoire RAM, fonctionnant sous Linux. Le code source a été écrit en Python 3.8 et interprété avec PyPy 3.9. PyPy est une implémentation alternative du langage de programmation Python, conçue pour être plus rapide et plus efficace en termes de consommation de mémoire par rapport à l'implémentation standard de Python CPython. En moyenne, PyPy 3.9 est 4.8 fois plus rapide que CPython 3.7. Les durées indiquées sont exprimées en secondes, mesurées en tant que temps de traitement du processeur et en supposant une valeur par défaut de 8 ms par défaut de page.

Spécifiquement concernant l'évaluation de l'algorithme SkyIR-UBS, les ensembles de données considérés ont été indexés avec agrégations dans un R*-arbre d'une taille de page de 4Ko. Un cache associé contenant 20 % des blocs du R*-arbre correspondant a été utilisé lors de l'expérimentation.

Nous avons généré des ensembles réalistes et représentatifs de données synthétiques décorrélées composés de 10 à 1 milliard de tuples pour respectivement 3, 6 et 9 dimensions reproduisant, chacune, des caractéristiques spécifiques (type, domaine de valeurs…) du cas d'utilisation.

6.1 L'ensemble des solutions

Pour la comparaison avec l'ensemble des solutions, nous n'avons conservé que l'algorithme de dp-idp initial le plus performant, à savoir SkyIR-UBS.

Même à cette condition, la figure 4, montrant le temps de réponses des différentes solutions, lorsque la cardinalité de l'ensemble des

données varie jusqu'à 50000 tuples, pour 3 dimensions, évalue SkyIR-UBS comme la moins efficace des solutions. Notre proposition de dp-idp avec hiérarchie de dominance explose nettement moins vite. Pourtant, bien que meilleure que celle de SkyIR-UBS, son efficacité est négligeable face aux implémentations de CoSky. Même dans le pire cas (50000 tuples et 3 dimensions), les implémentations SQL et algorithmique de CoSky ont respectivement un temps de réponse de 0.268 seconde et de 4 minutes et 23 secondes là où notre version de dp-idp et SkyIR-UBS en ont un respectif de plus de 3 heures et de plus de 6 heures !

Pour la suite des évaluations, nous délaissons les algorithmes SkyIR-UBS et dp-idp avec hiérarchie de dominance, peu efficaces par rapport aux implémentations de CoSky, afin de pouvoir les étudier avec de plus fortes cardinalités et de plus grandes dimensionnalités.

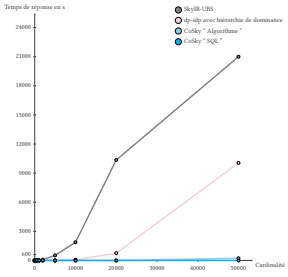

Figure 4: Temps de réponse des différentes solutions

6.2 Implémentations de CoSky

Pour la comparaison des implémentations de CoSky, nous considérons la figure 5. Les évaluations sont effectuées pour des cardinalités allant jusqu'à 200000 tuples avec, respectivement, 3, 6 et 9 dimensions. Dans le pire des cas, les temps de réponse sont respectivement d'environ 40 minutes, 53 minutes et 1 heure 43 minutes pour l'implémentation algorithmique de CoSky alors qu'elles sont respectivement d'environ 1 seconde, 54 secondes et 2 minutes et 8 secondes pour l'implémentation SQL. Nous constatons que, bien que les deux solutions soient particulièrement efficaces, et notamment par rapport aux solutions existantes, la version intégrée au SGBD relationnelles l'est considérablement.

6.3 CoSky en SQL

Pour la comparaison de l'implémentation SQL de CoSky, nous considérons la figure 6. Les évaluations sont effectuées pour des cardinalités allant jusqu'à 2 millions de tuples avec 3, 6 et 9 dimensions. Dans le pire des cas, les temps de réponse sont d'environ 3 heures pour 9 dimensions, 16 minutes pour 6 dimensions et 10 secondes pour 3 colonnes. Comme attendu pour toutes sortes de calculs dans un contexte Skyline, notre implémentation SQL de CoSky est sensible à l'augmentation du nombre de dimensions. Ce problème est connu pour être difficile pour de hautes dimensionnalités, même avec une modélisation en RAM ([6]).

6.4 CoSky avec 3 dimensions

Pour l'évaluation de l'implémentation SQL de CoSky avec 3 dimensions, nous considérons la figure 7. Les évaluations sont effectuées

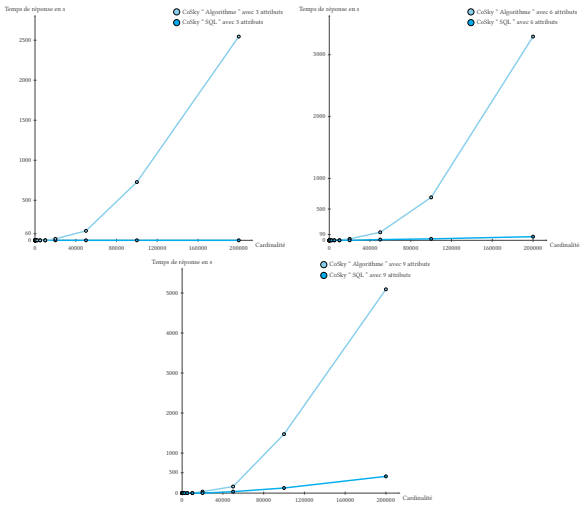

Figure 5: Temps de réponse de CoSky SQL (3, 6 et 9 attributs)

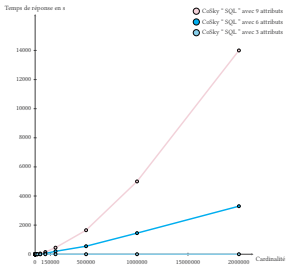

Figure 6: Temps de réponse de CoSky en SQL

pour des cardinalités allant jusqu'à 1 milliard de tuples avec 3 dimensions. Dans le pire cas, le temps de réponse est de moins de 3 heures. De plus, la lecture de la figure 7 semble nous indiquer qu'à nombre de dimension constant, l'évolution du temps de réponse est linéaire par rapport à l'augmentation de la cardinalité. Cela constitue une propriété très avantageuse de l'implémentation de CoSky en SQL, et c'est sans aucun doute la seule solution évaluée lors des expérimentations dans ce cas.

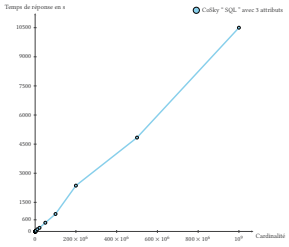

Figure 7: Temps de réponse de CoSky en SQL avec 3 attributs

7 CONCLUSION

Dans cet article, nous avons présenté des nouvelles méthodes efficaces de classement de Skyline.

La première proposée consiste en l'amélioration de la méthode dp-idp par l'utilisation d'une hiérarchie de dominance, offrant un classement plus rapide que la méthode initiale. Un exemple d'implémentation algorithmique de la méthode a été proposé.

La deuxième est la méthode CoSky, basée à la fois sur l'approche TOPSIS issue de l'aide à la décision multicritère et la mesure de similarité cosinus de Salton issue de la recherche d'information. Un exemple d'implémentation SQL ainsi qu'un exemple d'implémentation algorithmique de la méthode ont également été proposés.

L'algorithme DeepSky a été introduit afin 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.

Au vu de leur pertinence et de leur performance mises en évidence lors de l'évaluation expérimentale, nous pensons que les solutions exposées pourraient faire l'objet de futures publications, et d'un développement prochain d'une plateforme algorithmique de recherche *open source*.

8 ANNEXE

8.1 Préparation et remarques

8.1.1 Conversion d'une préférence Skyline. Soit r une relation composée des attributs $A_1, \ldots, A_m$. La conversion de $Pref(A_i) = MIN$ à $Pref(A_i) = MAX$, ou sa réciproque, est délicate dans les schémas d'ordonnacement basés sur un modèle vectoriel comme c'est le cas pour CoSky avec le cosinus de Salton. Il vaut mieux pour cela privilégier l'inversion mathématique, et non le complémentaire (pas plus qu'une conversion de minimum vers maximum ou de maximum vers minimum), des valeurs de l'attribut du Skyline.

En effet, le plus classiquement, la formule de calcul de valeurs complémentaires employée est $\forall t \in r, t[A_i'] = \bigvee A_i - t[A_i]$ où $\bigvee A_i$ est le *supremum* de A_i, une valeur théorique suffisamment grande pour garantir que toutes les valeurs transformées restent positives et interprétables. $\bigvee A_i$ peut être la valeur maximale actuelle (*optimum*) ou possible (maximum) de l'attribut A_i. Une approche alternative est d'utiliser la formule $\forall t \in r, t[A_i'] = \bigwedge A_i + \bigvee A_i - t[A_i]$ où $\bigwedge A_i$ est l'*infimum* de A_i. Avec cette approche, *infimum* et *supremum* sont le plus souvent respectivement le minimum et les maximum.

Dans tous les cas, $\bigvee A_i$ (respectivement $\bigwedge A_i$) peut être inconnu ou varier.

L'inversion de valeur pose moins de problème, en particulier pour des valeurs strictement positives. Cette solution a plusieurs avantages et convient souvent mieux que l'utilisation des valeurs complémentaires. Cette méthode est indépendante de valeurs de bornes, il n'est ainsi pas nécessaire de connaître ou de calculer de *supremum*, et maintient les proportions correctes.

Cependant, même ainsi nous n'obtenons pas une correspondance parfaite à cause de la dispersion (l'étendue, la variance, la déviation absolue moyenne, la somme, etc.) des données.

Exemple 8.1 - Avec la relation `Pokémon` (cf. tableau 1), si nous cherchons à convertir la condition de taux de `Victoire` le plus fort possible par celle du taux d'`Échec` (ou $\texttt{Victoire}^{-1}$) le plus bas possible, nous pouvons inverser les valeurs correspondantes. De la sorte, nous obtenons le tableau 10, qui, pour des raisons de commodité, ne conserve aucun attribut ou commentaire, seulement le `RowId` des tuples. Notons cependant, bien que les relations de dominations entre les tuples restent correctes après transformation, les sommes des valeurs des attributs des points du Skyline (de `RowId` 1, 2 et 4), respectivement $\Sigma = 160$ et $\Sigma = 9/140 \approx 0,0643$ sont différentes. Dans cette situation, la plupart des normalisations, comme celle par la somme employée par CoSky, donnent des résultats différents suivant que $Pref(\texttt{Victoire}) = MAX$ ou $Pref(\texttt{Échec}) = MIN$.

8.1.2 Unification des préférences Skyline. Nous considérons que, pour les schémas d'ordonnacement basés sur un modèle vectoriel comme c'est le cas pour CoSky avec le cosinus de Salton, il y a nécessité d'avoir des valeurs comparables entre elles, et donc d'avoir des préférences Skyline identiques. En effet, par essence, pour que le cosinus de Salton donne une mesure significative de similarité, il est crucial que les composantes des vecteurs représentent des entités comparables ou au moins normalisées de manière cohérente. Cela signifie que les valeurs des vecteurs doivent être sur des échelles comparables et avoir des unités similaires.

De la sorte, nous pouvons calculer un score par " l'idéal mimimum " ou alors un score par " l'idéal maximum ", au choix, bien que nous préconisons, pour une meilleure précision et une économie de calculs, d'unifier les préférences en fonction de la préférence initialement majoritaire.

Le processus d'unification des préférences Skyline doit naturellement être effectué préalablement à toute étape de calcul.

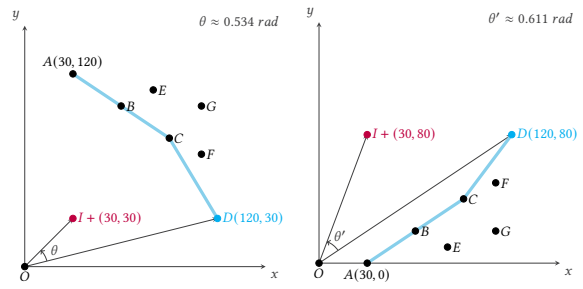

Figure 8: Préf. (`MIN`, `MIN`) (gauche) / (`MIN`, `MAX`) (droite)

Exemple 8.2 - Dans la figure 8, nous considérons, par commodité de représentation, deux critères d'évaluation ayant, dans le premier cas, des préférences Skyline unifiées (`MIN`, `MIN`), et dans le second cas, des préférences Skyline mixtes (`MIN`, `MAX`) : le critère de préférence `MIN` étant en abscisse, et celui de préférence `MAX` en ordonnée.

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, ou front, (d'efficacité) de Pareto (l'ensemble des segments en couleur) parce qu'ils sont dominés par les autres points. On qualifie A, B, C et D d'efficaces, et de Pareto-optimaux. Le point idéal théorique, ou abstrait, noté $I+$, de coordonnées $(30, 30)$ à gauche de la figure 8, et de coordonnées $(30, 80)$ à droite, domine tous les points du Skyline.

Le cosinus de Salton respectivement de l'angle θ et θ', tout deux formés par le vecteur allant de l'origine à $I+$ et le vecteur allant de l'orgine vers D, a pour valeur environ 0.534 *rad* à gauche de la figure 8, et une valeur différente, d'environ 0.611 *rad* à droite.

Pour passer d'un cas à l'autre, nous avons employé le calcul de valeurs complémentaires classique (avec l'utilisation du *supremum*) par raison de commodité. En effet, notamment, l'inversion d'uniquement certaines valeurs rend la représentation peu claire (certaines étant plus petites que 1 là où d'autres sont bien plus grandes), mais quelle que soit la méthode de conversion employée, la problématique reste la même.

Exemple 8.3 - Le plus judicieux est donc d'unifier les préférences Skyline de la relation `Pokémon` (cf. tableau 1), et de le faire avec (`MIN`, `MIN`, `MIN`) en inversant les valeurs de l'attribut `Victoire` (devenant ainsi $\texttt{Victoire}^{-1}$ ou `Échec`), comme avec le tableau 10.

8.1.3 Normalisation et standardisation. La normalisation et la standardisation sont des techniques utilisées pour transformer les données afin qu'elles soient sur une échelle comparable, généralement entre 0 et 1. Nous présentons, dans ce paragraphe, quelques méthodes courantes de normalisation, avec leurs formules de calcul permettant de transformer une valeur originale x en sa valeur normalisée ou standardisée x'.

Normalisation statistique. La normalisation statistique, ou normalisation par plage, est adaptée pour des données sans distribution normale. C'est une méthode sensible aux valeurs aberrantes, idéale pour des données avec des minimums et des maximums bien définis. Sa formule de calcul est :

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardisation. La standardisation centre les données autour de la moyenne et les met à l'échelle en termes d'écart-type. Elle est adaptée lorsque la distribution des données est approximativement normale. Cette méthode est moins sensible aux valeurs aberrantes que ne l'est la normalisation statistique. Avec $\bar{x}$ la moyennes des valeurs et σ leur écart-type, sa formule de calcul est :

$$x' = \frac{x - \bar{x}}{\sigma}$$

Normalisation par la mise à l'échelle décimale. La normalisation par la mise à l'échelle décimale transforme les données en déplaçant la virgule décimale des valeurs originales. Bien que simple à comprendre et à appliquer, cette méthode est moins couramment utilisée que les autres. Avec k le plus grand nombre de chiffres à gauche de la virgule de x, sa formule de calcul est :

$$x' = \frac{x}{10^k}$$

Normalisation par la somme. La normalisation par la somme transforme les données en divisant chaque valeur par la somme totale de toutes les valeurs. Cette méthode est particulièrement adaptée lorsque l'échelle relative des données est plus importante que leurs valeurs absolues, et elle convient bien pour les distributions proportionnelles. C'est cette solution que nous avons choisi pour la méthode CoSky. Avec Σx la somme de toutes les valeurs, sa formule de calcul est :

$$x' = \frac{x}{\Sigma x}$$

Comparaison des méthodes. Chacune des méthodes de normalisation ou de standardisation a ses applications privilégiées et ses avantages, et la méthode la plus appropriée et efficace dépend souvent de la nature spécifique des données et des objectifs de l'analyse.

REFERENCES

[1] Hana Alouaoui, Lotfi Lakhal, Rosine Cicchetti, and Alain Casali. [n. d.]. CoSky: A Practical Method for Ranking Skylines in Databases. In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2019, Volume 1: KDIR, Vienna, Austria, September 17-19, 2019* (2019), Ana L. N. Fred and Joaquim Filipe (Eds.). ScitePress, 508–515. https://doi.org/10.5220/0008363005080515

[2] Ilaria Bartolini, Paolo Ciaccia, Vincent Oria, and M. Tamer Özsu. [n. d.]. Flexible Integration of Multimedia Sub-Queries with Qualitative Preferences. 33, 3 ([n. d.]), 275–300. https://doi.org/10.1007/s11042-007-0103-1

[3] Majid Behzadian, S. Khanmohammadi Otaghsara, Morteza Yazdani, and Joshua Ignatius. [n. d.]. A State-of-the-Art Survey of TOPSIS Applications. 39, 17 ([n. d.]), 13051–13069. https://doi.org/10.1016/j.eswa.2012.05.056

[4] Jon Louis Bentley, H. T. Kung, Mario Schkolnick, and Clark D. Thompson. [n. d.]. On the Average Number of Maxima in a Set of Vectors and Applications. 25, 4 ([n. d.]), 536–543.

[5] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. [n. d.]. The Skyline Operator. In *ICDE* (2001). 421–430.

[6] Chee-Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. [n. d.]. On High Dimensional Skylines. In *Advances in Database Technology - EDBT 2006*, Yannis Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Mike Hatzopoulos, Klemens Boehm, Alfons Kemper, Torsten Grust, and Christian Boehm (Eds.). Vol. 3896. Springer Berlin Heidelberg, 478–495. https://doi.org/10.1007/11687238_30

[7] Vittorio Fabris. [n. d.]. *Flexible Skylines, Regret Minimization and Skyline Ranking: A Comparison to Know How to Select the Right Approach.* https://doi.org/10.48550/ARXIV.2201.10179

[8] Yunjun Gao, Qing Liu, Lu Chen, Gang Chen, and Qing Li. [n. d.]. Efficient Algorithms for Finding the Most Desirable Skyline Objects. 89 ([n. d.]), 250–264. https://doi.org/10.1016/j.knosys.2015.07.007

[9] Jingwen Huang. [n. d.]. Combining Entropy Weight and TOPSIS Method for Information System Selection. In *2008 IEEE Conference on Cybernetics and Intelligent Systems* (Chengdu, China, 2008-09). IEEE, 1281–1284. https://doi.org/10.1109/ICCIS.2008.4670971

[10] Young-Jou Lai, Ting-Yun Liu, and Ching-Lai Hwang. [n. d.]. TOPSIS for MODM. 76, 3 ([n. d.]), 486–500. https://doi.org/10.1016/0377-2217(94)90282-8

[11] Lotfi Lakhal, Sébastien Nedjar, and Rosine Cicchetti. [n. d.]. Multidimensional Skyline Analysis Based on Agree Concept Lattices. 21, 5 ([n. d.]), 1245–1265. https://doi.org/10.3233/IDA-163111

[12] Farhad Hosseinzadeh Lotfi and Reza Fallahnejad. [n. d.]. Imprecise Shannon's Entropy and Multi Attribute Decision Making. 12, 1 ([n. d.]), 53–62. https://doi.org/10.3390/e12010053

[13] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. [n. d.]. Progressive Skyline Computation in Database Systems. 30, 1 ([n. d.]), 41–82.

[14] Timotheus Preisinger and Markus Endres. [n. d.]. Looking for the Best, but Not Too Many of Them: Multi-Level and Top-k Skylines. ([n. d.]).

[15] George Valkanas, Apostolos N. Papadopoulos, and Dimitrios Gunopulos. [n. d.]. 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* (2014) *(CEUR Workshop Proceedings, Vol. 1133)*, K. Selçuk Candan, Sihem Amer-Yahia, Nicole Schweikardt, Vassilis Christophides, and Vincent Leroy (Eds.). CEUR-WS.org, 182–187. https://ceur-ws.org/Vol-1133/paper-31.pdf

[16] Akrivi Vlachou and Michalis Vazirgiannis. [n. d.]. Ranking the Sky: Discovering the Importance of Skyline Points through Subspace Dominance Relationships. 69, 9 ([n. d.]), 943–964. https://doi.org/10.1016/j.datak.2010.03.008

[17] Man Yiu and Nikos Mamoulis. [n. d.]. Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data. 483–494.