HAL
open science

Décomposition des jeux pour le General Game Playing

Aline Hufschmitt, Jean-Noël Vittaut, Nicolas Jouandeau

To cite this version:

Aline Hufschmitt, Jean-Noël Vittaut, Nicolas Jouandeau. Décomposition des jeux pour le General Game Playing. 12èmes Journées de l'Intelligence Artificielle Fondamentale (JIAF'18), Jun 2018, Amiens, France. hal-02317296

HAL Id: hal-02317296

https://hal.science/hal-02317296v1

Submitted on 15 Oct 2019

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.

Actes JIAF 2018

Décomposition des jeux pour le General Game Playing

Aline Hufschmitt **Jean-Noël Vittaut** **Nicolas Jouandeau**

LIASD - University of Paris 8, France

{alinehuf,jnv,n}@ai.univ-paris8.fr

Résumé

Dans cet article nous présentons une approche pour la décomposition des jeux généraux décrits en *Game Description Language* (GDL). Dans le domaine du *General Game Playing* (GGP), les joueurs peuvent significativement diminuer le coût de l'exploration d'un jeu si ils disposent d'une version décomposée de celui-ci. Les travaux existants sur la décomposition des jeux s'appuient sur la structure syntaxique des règles, sur des habitudes d'écriture du GDL ou sur le coûteux calcul de la forme normale disjonctive des règles. Nous proposons une méthode plus générale pour décomposer les jeux solitaires ou multi-joueurs, fondée sur la collecte d'informations durant des simulations (*playouts*). Notre méthode est capable de traiter les différents types de jeux composés et de prendre en charge certains cas difficiles comme les jeux à actions composées et les jeux en série. Nous avons testé notre approche sur un panel de 597 jeux GGP. Pour 70% des jeux, la décomposition nécessite moins d'une minute en faisant 5k *playouts*. Nous montrons de 87% d'entre eux peuvent être correctement décomposés après seulement 1k *playouts*.

Abstract

This paper presents a general approach for the decomposition of games described in *Game Description Language* (GDL). In the *General Game Playing* framework (GGP), players can drastically decrease game search cost if they hold a decomposed version of the game. Previous works on decomposition rely on syntactical structures and writing habits of the GDL, or on the disjunctive normal form of the rules, which is very costly to compute. We offer an approach to decompose single or multi-player games based on information gathering during random simulations of the game (*playouts*). Our approach can handle the different classes of compound games and can process some difficult cases like synchrounous parallel games with compound moves and serial games. We tested our program on 597 games. Given 5k *playouts*, 70% of the game are decomposed in less than one minute. We demonstrate that for 87% of the games, 1k *playouts* are sufficient to obtain a correct decomposition.

1 Introduction

Décomposer un problème en sous-parties solubles séparément et synthétiser leurs résultats pour résoudre le problème global peut significativement réduire le coût de la résolution d'un jeu en *General Game Playing*. Des travaux précédents proposent quelques approches pour jouer avec des jeux décomposés solitaires [1, 3] ou multi-joueurs [10]. Cependant, la décomposition des jeux est un pré-requis essentiel. Dans cet article nous proposons une approche pour obtenir une décomposition applicable à toutes les catégories de jeux.

Différentes catégories de jeux se distinguent en fonction des difficultés qu'elles représentent pour la décomposition [5] : jeux utilisant un compteur (*stepper*) comme *Asteroids* [1], jeux parallèles synchrones ou asynchrones comme *Chinook* [2] ou *Double Tictactoe Dengji* [3], jeux synchrones parallèles avec des actions composées comme *Snake-Parallel* [4], jeux en série comme *Factoring Turtle Brain* [5], jeux multiples comme *Multiple-Sukoshi* [6] et jeux impartiaux débutant avec plusieurs groupes d'objets comme *Nim* [7].

1. *Astéroïds* consiste à diriger un petit vaisseau spatial (en 50 étapes maximum) jusqu'à un astéroïde et à s'arrêter dessus. Si le vaisseau s'arrête avant d'atteindre la destination, la partie prend fin avec le score 0.

2. *Chinook* est composé de deux parties de dames anglaises disputées sur les cases blanches et les cases noires du même damier. *Blanc* commence sur les cases blanches et *Noir* sur les noires. A chaque tour, chaque joueur réplique sur l'autre partie du damier.

3. A chaque tour un joueur choisi une grille de *Tictactoe* parmi deux pour y placer une de ses marques. Le but est de gagner dans les deux grilles.

4. Le but de *Snake* est de déplacer, pendant 50 étapes, un serpent qui grandit régulièrement sans qu'il ne se morde la queue.

5. *Factoring Turtle Brain* est une succession de jeux *LightsOn* où le joueur doit allumer 4 lampes : chaque lampe s'éteint progressivement pendant qu'il allume les autres.

6. *Sukoshi* consiste à créer un chemin en alignant des entiers ordonnés. Dans la version *multiple* une seule grille compte pour le score, les autres sont inutiles.

7. Dans *Nim* chaque joueur prélève tour à tour un nombre quelconque d'allumettes dans l'une des 4 piles proposées. Celui qui n'a plus d'allumette à prélever perd.

```
(role r)
(light a) (light b) (light c) (light d)
(<= (legal r (push ?x)) (not (true (on ?x))) (light ?x))
(<= (next (on ?x)) (does r (push ?x)))
(<= (next (on ?x)) (true (on ?x)))
(<= terminal (true (on a)))
(<= (goal r 0) (not (true (on b))) (not (true (on c))))
(<= (goal r 40) (true (on b)) (not (true (on c))))
(<= (goal r 60) (not (true (on b))) (true (on c)))
(<= (goal r 100) (true (on b)) (true (on c)))
```

FIGURE 1 – Jeu minimaliste représenté en GDL. Le joueur peut allumer 4 lampes. Allumer *a* met fin au jeu. Il n'obtient le score maximal que si *b* et *c* sont allumés à la fin.

Les jeux en série et à actions composées représentent des cas particuliers difficiles à décomposer. Dans les jeux en série, le début d'un sous-jeu est conditionné par l'achèvement du précédent, ceci créant un lien logique fort entre les fluents décrivant l'état des sous-jeux. Les actions composées ont la propriété d'avoir un effet simultané sur l'état de sous-jeux distincts : séparer ces effets est un prérequis pour pouvoir séparer les sous-jeux.

Le GDL utilisé pour décrire les jeux généraux est un langage logique proche de Prolog faisant l'hypothèse d'un monde clos et utilisant la négation par défaut. Les règles logiques, permettant d'inférer les fluents vrais dans l'état suivant en fonction d'une description partielle de l'état courant et de l'action jointe des joueurs, n'indiquent pas clairement les effets des actions. Des axiomes du cadre indiquent par exemple les fluents qui gardent une valeur vraie d'un état à l'autre en fonction de certaines actions : les actions non présentes dans ces règles sont susceptibles d'avoir un effet négatif. Cependant, parmi elles, certaines peuvent être illégales quand la règle s'applique. Une inférence exacte des liens de causalité entre les actions et les fluents d'un jeu décrit en GDL reviendrait à explorer tous ses états ce qui est incalculable dans un temps raisonnable.

Cet article est une version française détaillée de [6]. Nous proposons ici une approche probabiliste fondée sur des simulations pour identifier l'effet des actions et décomposer les différentes catégories de jeux composées mentionnées ci-dessus. Nous commençons par détailler les travaux précédents sur la décomposition (§2) et définissons la nature de la décomposition souhaitée (§3). Nous présentons les différentes étapes de notre approche (§4) et les résultats expérimentaux obtenus sur 597 jeux (§5). L'article s'achève sur une conclusion et la présentation des perspectives (§6).

2 État de l'art

Nous assumons la familiarité du lecteur avec le domaine du GGP [2] ainsi qu'avec le GDL [7]. Une description GDL prend la forme d'un ensemble d'assertions et de règles logiques exprimées dans une syntaxe proche du Lisp. Un ensemble de mots clefs permet de définir les caractéristiques du jeu. La figure 1 donne un exemple de description GDL. Les mots clefs sont représentés en gras. La syntaxe `(<= a x ... y)` signifie que *a* est vrai si la conjonction de prémisses $x \wedge ... \wedge y$ est vraie ; les variables sont signalées par un point d'interrogation. Le prédicat *legal* permet de spécifier la légalité d'une action ; les règles *next* indiquent les conditions dans lesquelles un fluent sera vrai dans l'état suivant ; *terminal* indique si la position courante est une fin de jeu ; *goal* précise le score obtenu si la position est terminale. Les règles sont exprimées en fonction des actions effectuées (*does*) et des fluents décrivant l'état du jeu (*true*).

Pour réaliser des simulations (*playouts*) très rapides, Vittaut et al. [9] proposent de transformer les règles d'un jeu en circuit logique évaluable avec des opérateurs binaires. Les règles sont instanciées rapidement avec Yap Prolog et le principe du *tabling* [8]. Un graphe de règles similaire à un *propnet* est construit à partir de ces règles instanciées. Enfin, après une phase d'optimisation et de factorisation, le circuit obtenu utilise la valeur des fluents (*true*) et actions (*does*) en entrée et produit l'état des prédicats *legal*, *next*, *goal* et *terminal* en sortie.

Pour décomposer les jeux solitaires, Günther et al. [4] proposent la construction d'un graphe de dépendances dont les nœuds sont les actions et les fluents non instanciés du jeu. Les arcs représentent les potentiels liens de précondition, ainsi que les effets positifs et négatifs entre les fluents et les actions. Les parties connexes du graphe représentent les différents sous-jeux. Des fluents indépendants des actions peuvent être utilisés pour indiquer le joueur courant dans les jeux à coups alternés. Comme ils peuvent préconditionner les actions de plusieurs sous-jeux distincts, ces fluents sont isolés dans un sous-jeu séparé pour éviter qu'ils ne les regroupent tous. Leur approche est testée sur le jeu *Incredible*.

Une extension de cette approche aux jeux multi-joueurs est proposée par Zhao et al. [11]. Ils proposent des traitements spécifiques distincts pour les jeux en série et à actions composées [10] qui malheureusement s'appuient sur la syntaxe et sur des habitudes d'écriture des descriptions GDL. Une simple réécriture des règles met ses traitements en échec. L'approche est testée sur les jeux *Double Crisscross2* et différentes variantes de *Tictactoe* (*Double*, *Serial*, *Parallel*).

Dans un article précédent [5], nous proposons une approche plus générale fondée sur une analyse des règles avant le jeu. Le nombre de jeux traités est plus important que celui des approches précédentes. Cependant, les heuristiques utilisées pour identifier l'effet des actions ne prennent en compte que les effets explicitement décrits dans les règles. Les effets implicites sont ignorés ce qui conduit à des cas de sur-décomposition. Pour résoudre le problème des actions composées, cette approche identi-

fie des méta-actions : des ensembles d'actions ayant un même effet dans les mêmes circonstances dans l'un des sous-jeux. Ces actions sont définies comme associées aux mêmes fluents dans les différentes clauses d'une règle *next* donnée et comme ayant un fluent en commun comme prémisse de leur légalité. Cette détection requiert le coûteux calcul de la forme normale disjonctive totalement développée (DNFD) de ces règles. La détection des jeux en série est limitée à seulement deux sous-jeux identifiés par le fait qu'ils correspondent à une partition des actions en deux groupes. Cette approche détecte également les sous-jeux inutiles dans les jeux *multiples*. Elle est testée sur un panel de 40 jeux représentatifs : 7 jeux d'un niveau trivial (*Tictactoe*) à complexe (*Hex*) et 33 jeux composés représentatifs des différentes catégories présentées.

Pour limiter le temps de calcul, il est possible de garder les prédicats auxiliaires sous forme d'atomes et de ne pas développer complètement les expressions. La forme obtenue (DNF) permet une décomposition plus rapide pour certains jeux : une décomposition non obtenue après une heure avec la DNFD peut être obtenue en moins d'une seconde avec la DNF (fig.7). Cependant cette approche est potentiellement moins fiable : en fonction de la formulation des règles, la détection des méta-actions peut échouer et faire échouer la décomposition. Notons qu'avec DNF comme avec DNFD, la décomposition de *Chomp* est un échec et celle de *Blocker-Parallel* requiert un temps de calcul de plus d'une heure.

Dans cet article, nous proposons une approche plus robuste pour la détection de l'effet des actions fondée sur la collecte d'informations statistiques durant les *playouts*. Nous utilisons un circuit logique encodant les règles du jeu pour réaliser des simulations rapides et nous collectons des informations sur la corrélation entre les actions jouées et le changement des fluents. Le circuit est aussi utilisé pour propager ou rétro-propager différents signaux afin d'en déduire les liens de précondition. Notre méthode de décomposition et plus particulièrement la détection des actions composées ne nécessite pas le calcul de DNF ou DNFD. Pour décomposer les jeux en série, nous proposons le concept de *point de croisement* qui permet d'identifier un nombre quelconque de parties d'un jeu jouées séquentiellement. Pour finir, nous présentons les résultats de notre approche sur 597 jeux contre 1,4 et 40 pour les approches précédentes.

3 Jeu, Sous-jeu et décomposition

Un jeu général peut être considéré comme un automate à états finis, la structure duquel peut être représentée par un graphe orienté acyclique. Soit F l'ensemble des fluents du jeu. Un état du jeu est un ensemble $s \subset F$ et une transition un tuple $(s, s') \subset F^2$. Une décomposition est une partition $(F_1, ..., F_n)$ telle que $\bigcup_{i=1}^{n} F_i = F$ et $\forall i, j, i \neq j : F_i \cap F_j = \emptyset$. Dans un sous-jeu i un état est un ensemble $s_i \subset F_i$ et une transition est un couple $(s_i, s'_i) \subset F_i^2$ tel qu'il existe une transition $(s, s') \subset F^2$ où $s_i = s \cap F_i$ et $s'_i = s' \cap F_i$.

Definition 1 (choix libre d'une transition) *Considérons un état $s = s_1 \cup s_2$ d'un jeu, avec s_1 un état du premier sous-jeu et s_2 un état du second. Nous pouvons choisir* librement *une transition (s_1, s'_1) dans le premier sous-jeu et (s_2, s'_2) dans le second si, dans le jeu global, il existe une transition $(s_1 \cup s_2, s'_1 \cup s'_2)$ ou une séquence de transitions $(s_1 \cup s_2, s'_1 \cup s_2), (s'_1 \cup s_2, s'_1 \cup s'_2)$.*

Definition 2 (décomposition compatible) *La décomposition d'un jeu en deux sous-jeux est* compatible *avec le jeu global si dans chaque sous-jeu il est possible de se déplacer librement d'un état à un autre état distinct.*

La généralisation des définitions 1 et 2 à plus de deux sous-jeux est évidente.

Definition 3 (décomposition correcte) *Une décomposition est* correcte *si elle est compatible avec le jeu global et si il existe des conditions de victoire indépendantes (fonction d'évaluation) dans les sous-jeux telles que gagner dans tous les sous-jeux implique de gagner le jeu global.*

Dans l'exemple de la figure 1, chaque lampe peut être isolée dans un sous-jeu distinct, car il est possible de choisir librement une transition pour allumer une des lampes, et des fonctions d'évaluation indépendantes sont identifiables car chaque lampe impliquée dans le calcul du score contribue à l'obtention d'une partie des points.

D'après ces définitions, nous considérons en revanche qu'un jeu comme *Nine Board Tictactoe* n'est pas décomposable. Dans ce jeu regroupant 9 grilles de *Tictactoe* disposées en 3 lignes par 3 colonnes, chaque coup d'un joueur dans la case d'indice i d'une grille détermine la grille i suivante ou l'adversaire devra répliquer, le but étant d'aligner 3 marques dans l'une des grilles. Les transitions dépendent du coup précédent et ne peuvent donc pas être choisies librement si chaque grille constitue un sous-jeu. Un jeu comme *Blocker* est également non décomposable. Dans ce jeu, disputé sur un plateau de 4×4 cases, *Crosser* doit placer sa marque dans les cases pour bâtir un pont d'un côté à l'autre du plateau tandis que *Blocker* essaye de lui barrer la route en déposant ses propres marques. Même s'il est possible de choisir librement les transitions dans les 16 sous-jeux constitués chacun d'une seule case du jeu global, il n'existe pas de conditions de victoire indépendantes permettant d'évaluer le score dans ces sous-jeux, une case isolée pouvant, quelque soit le joueur, tout aussi bien appartenir à une position globale gagnante que perdante.

4 Notre approche

Pour identifier les différents sous-jeux, nous construisons un *graphe de dépendances* dont les nœuds sont des fluents totalement instanciés et des *méta-actions* (voir déf.8 et §4.4). Les arcs représentent les relations de causalité (effet ou précondition) entre eux. La construction de ce graphe est détaillée à la section 4.6. Les zones connexes représentent les sous-jeux. Les arcs sont associés à des pondérations permettant d'identifier des zones faiblement connexes du graphe et de séparer certains sous-jeux faiblement connectés sous certaines conditions expliquées à la section 4.7.

Pour l'analyse des relations entre les fluents et les actions, nous utilisons les définitions suivantes.

Definition 4 (prémisse) *Soit F l'ensemble des fluents instanciés du jeu et $f \in F$ un fluent positif ou négatif. Soit R l'ensemble des rôles r et A l'ensemble de toutes les actions instanciées a des joueurs telles que $a = does(r, o)$. Soit h le conséquent d'une règle GDL totalement instanciée. $g \in F \cup A$ est une prémisse de h si :*

- *g est dans le corps de cette règle, ou*
- *g est dans le corps d'une règle instanciée de conséquent i avec i une prémisse de h.*

*g est une **prémisse non-contradictoire** de h si aucune contradiction n'existe entre g et une autre prémisse de h, i.e. si h peut être vraie quand g est vraie.*
*g est une **prémisse exclusive** de h si h est vraie quand g est vraie quelque soit la valeur des autres prémisses.*

L'effet des actions n'étant pas explicitement décrit dans les règles GDL, nous définissons un *effet* comme une observation constatée lors d'une transition dont la cause n'est pas connue *à priori* :

Definition 5 (effet positif ou négatif) *Un effet positif f^+ (resp. négatif f^-) est le changement de valeur d'un fluent de* faux *(resp.* vrai*) à* vrai *(resp.* faux*) observé durant une transition d'un état à un autre. Un effet quelconque (positif ou négatif) observé pour un fluent $f \in F$ est représenté par f^*.*

Certains effets surviennent simultanément. Nous distinguons deux types d'effets cooccurrents :

Definition 6 (Effets Globalement Cooccurrents (EGC)) *L'ensemble des effets globalement cooccurrents avec f^* notés $EGC(f^*)$ est composé des effets qui surviennent simultanément avec f^* dans toutes les transitions explorées du jeu. Notons $|CGE(f^*)|$ le nombre de fois que cette cooccurrence a été observée durant les playouts.*

Definition 7 (Effets Cooccurrents par Actions (ECA)) *L'ensemble des effets cooccurrents par action avec f^* notés $ECA(a, f^*)$ est composé des effets qui surviennent simultanément avec f^* dans les transitions explorées du jeu où l'action a est jouée. Notons $|ECA(a, f^*)|$ le nombre de fois que cette cooccurrence a été observée durant les playouts.*

Une méta-action est l'ensemble des actions a responsables d'un même effet f^* dans les mêmes circonstances. Ces circonstances correspondent aux fluents conditionnant la légalité des actions i.e. les prémisses des règles de conséquent $legal(a)$, mais également aux fluents intervenant en conjonction avec les actions dans les prémisses des règles de conséquent *next(f)*. Ces règles indiquent si les circonstances sont réunies pour que l'action ait bien un effet. Identifier les actions intervenant en conjonction avec les mêmes fluents dans les différentes clauses d'une règle *next* implique de disposer de la forme normale disjonctive de cette règle. Cependant, par une comparaison des ensembles d'actions intervenant en conjonction avec chaque fluent séparément il est également possible de les identifier.

Les actions non responsables d'un effet f^*, mais utilisées en conjonction avec un même fluent g dans les prémisses d'une règle de conséquent $next(f)$, correspondent à un ensemble $A' = A - (M \cup I)$ avec M la méta-action responsable de l'effet f^* et I un ensemble d'actions illégales dans les mêmes conditions. Ces actions illégales peuvent appartenir à un autre sous-jeu que f : dans ce cas, chaque méta-action responsable d'un effet sur les fluents de cet autre sous-jeu est incluse dans I. La comparaison des ensembles d'actions associés à un même effet, avec les ensembles des actions utilisées en conjonction avec une même prémisse dans une règle *next*, et avec les ensembles d'actions ayant la même prémisse dans leur règle *legal* permet donc d'identifier les méta-actions, même si les liens de causalité ne sont pas explicitement décrits dans les règles GDL. Nous proposons donc la définition suivante d'une méta-action ne nécessitant pas de disposer de la forme normale disjonctive des règles :

Definition 8 (méta-action)
Une méta-action $M(r, \mathcal{E}, \mathcal{N}, \mathcal{L})$ est un ensemble d'actions $a = does(r, o)$ tel que les conditions suivantes sont vérifiées :

- *$\mathcal{E} \neq \emptyset$ et pour chaque $f^* \in \mathcal{E}$, a est responsable de l'effet f^*.*
- *pour chaque fluent $g \in \mathcal{N}$, g est utilisé en conjonction avec a dans les prémisses de next(f), a est responsable de l'effet f^* et a n'est pas une* précondition exclusive *de next(f).*
- *pour chaque fluent $h \in \mathcal{L}$, h est une précondition non-contradictoire de legal(r,o). Si legal(r,o) est toujours vrai, $\mathcal{L} = \emptyset$.*
- *il n'existe pas $A' \subsetneq M(r, \mathcal{E}, \mathcal{N}, \mathcal{L})$ tel que pour chaque $a' \in A'$, g' est utilisé en conjonction avec a' dans les prémisses de next(f'), a' n'est pas une précondition exclusive de next(f') et a' n'a aucun effet*

sur f'.

Les jeux en série sont caractérisés par la présence d'un état ou d'un ensemble d'états nécessairement visités pendant une partie et correspondant à l'achèvement d'un sous-jeu et le début d'un autre. Aucune séquence d'action ne permet d'atteindre le reste du jeu sans passer par un de ces états. Ces états sont caractérisés par l'apparition d'un fluent ou d'un ensemble de fluents vrais que nous nommons *point de croisement*. Par exemple, dans *Blocker Serial*, un jeu constitué de deux parties de *Blocker* jouées séquentiellement, le fluent *game1overlock* signale la fin du premier sous-jeu et conditionne la légalité des actions du second sous-jeu. Dans *Asteroids Serial*, un jeu constitué de deux parties de *Asteroid*, le premier sous-jeu prend fin quand le vaisseau s'arrête, ce qui est représenté par la conjonction de deux faits (*north-speed1(0)* $\wedge$ *east-speed1(0)*) indiquant une vitesse nulle sur les 2 axes cardinaux. Ces *points de croisement* peuvent être identifiés à partir d'un *graphe causal* représentant les relations de causalités entre actions et fluents. Chaque fluent composant un *point de croisement* est un *fluent charnière* :

Definition 9 (fluent charnière) *Soit G un graphe orienté représentant les relations de causalités entre les actions et les fluents (positifs ou négatifs) d'un jeu. Soit C(G) la fermeture transitive de ce graphe. Un fluent x parmi les nœuds du graphe est un fluent charnière si dans C(G) :*

- *il existe au moins un arc $y \rightarrow x$,*
- *il existe au moins un arc $x \rightarrow a$ avec a une action,*
- *x n'est pas dans l'état initial du jeu.*

Un *point de croisement X* est un ensemble d'au moins un *fluent charnière*.

4.1 Détection statistique des effets

Pour identifier le lien de causalité entre les actions et les fluents, nous réalisons des *playouts* aléatoires et collectons des informations sur les effets observés. A chaque étape du jeu, chaque joueur doit choisir une action et une seule. L'ensemble des actions des différents joueurs constitue un mouvement joint. Dans les jeux à coups alternés, à une étape donnée du jeu, un seul joueur a le choix entre plusieurs actions légales tandis que les autres ne peuvent jouer qu'une seule action sans effet (souvent nommée *noop*). Pour chaque transition d'un *playout*, nous collectons le mouvement joint associé aux effets observés. Après un nombre donné de simulations, pour chaque action a qui apparaît dans $|J(a)|$ mouvements joints distincts, nous calculons la probabilité $\mathcal{P}(a, f^+)$ qu'une action a soit suivie d'un effet positif sur le fluent f :

$$\mathcal{P}(a, f^+) = (\sum_{0<i<|J(a)|} \frac{E(J(a)_i, f^+)}{O(J(a)_i)})/|J(a)|$$

avec $J(a)$ l'ensemble des mouvements joints incluant l'action a, $O(J(a)_i)$ le nombre d'occurrences d'un mouvement joint de cet ensemble, $E(J(a)_i, f^+)$ le nombre de fois où un effet f^+ a suivi l'application de ce mouvement joint. La probabilité qu'une action a soit suivie d'un effet négatif f^- est calculée de manière similaire.

$\mathcal{P}$ indique la probabilité d'observer un effet suite à une action. Cependant, certains fluents utilisés pour définir l'alternance des joueurs (prédicat *control*) ou pour définir un compteur de coups (*step*) changent à une étape du jeu quelque soit l'action jouée : ils sont *indépendants des actions*. La formulation des règles ne permet pas toujours de les détecter. Par exemple, la présence d'une action *noop* parmi les prémisses d'une règle *next* décrivant l'état suivant d'un fluent *control* ou *step*, peut laisser penser que l'action *noop* a un effet et que *control* ou *step* sont dépendants d'une action [8].

4.2 Filtrage des effets

Une valeur positive de $\mathcal{P}(a, f^*)$ n'indique pas si l'action a est responsable d'un effet sur f ou si il s'agit d'une simple corrélation. L'étape suivante pour l'identification des liens de causalité consiste à examiner les relations potentielles suggérées par une valeur positive de $\mathcal{P}$ pour filtrer les liens de causalité et éliminer les corrélations.

Nous commençons par détecter la présence de coups alternés : notre but est d'identifier les actions *noop* sans effet et les fluents *control* indépendants des actions. Un jeu à n joueurs est un jeu à coups alternés si à chaque tour $n-1$ joueurs n'ont qu'une seule action possible qui peut alors être considérée comme une action *noop*. Le fluent qui a la plus grande probabilité de changer quand une action *noop* est jouée est le fluent *control* correspondant. Ceci permet de détecter les actions *noop* et les fluents *control*. Si $next(f)$, indiquant qu'un fluent f est vrai à l'état suivant, ne compte que des actions *noop* parmi ses prémisses, alors ce fluent est considéré indépendant des actions.

Pour chaque action a nous vérifions toutes les relations potentielles suggérées par $\mathcal{P}(a, f^*) > 0$ afin de confirmer ou infirmer une relation de causalité entre a et f^*. En observant les règles du jeu, il est possible de déterminer si une règle décrit un lien explicite ou sous-entend un lien implicite possible de causalité entre une action et un effet. Si aucune règle de ce type n'existe dans la description GDL, on peut conclure qu'aucun lien de causalité ne peut exister entre l'action et l'effet : la probabilité $\mathcal{P}$ est alors remise à 0.

Par exemple, une règle *next(f) :- does(r,o), not(true(f)).* indique explicitement que l'action $a = does(r, o)$ est responsable d'un effet positif sur f. Dans une règle, *next(f) :- does(r,o), true(f).* un effet négatif potentiel provoqué par toute autre action que *does(r,o)* est suggéré. Enfin, *next(f) :-*

8. *noop*, *step* et *control* ne sont pas des mots clef du GDL. En compétition, les règles sont obfusquées ce qui ne permet pas d'identifier ces éléments par leur nom.

does(r,o). est une règle ambiguë qui laisse supposer une relation possible de causalité avec toute action selon la valeur courante de f.

En examinant les effets cooccurrents, il est possible de filtrer les liens de causalités non cohérents. Pour chaque lien de causalité éliminé, les différents effets cooccurrents sont reconsidérés jusqu'à ce qu'il ne soit plus possible d'en éliminer de nouveaux. Une action a a une probabilité nulle d'être la cause d'un effet f^* si $\mathcal{P}(a, g^*) = 0$ avec $g^* \in EGC(f^*)$. Au contraire le lien entre a et f^* est confirmé si il existe un g^* avec $\mathcal{P}(a, g^*) > 0$ et soit $g^* \in EGC(f^*)$ avec $|EGC(f^*)| > \Psi$ ou $g^* \in ECA(a, f^*)$ avec $|ECA(a, f^*)| > \Theta$, ou Ψ et Θ sont des seuils nécessaires pour ne considérer que les cooccurrences réelles et écarter les coïncidences.

Pour les jeux multi-joueurs, nous comparons également les probabilités attribuées aux différentes actions d'un mouvement joint pour un même effet f^*. Si une action a d'un mouvement joint a une probabilité Φ fois supérieure à celle d'une action b d'être suivie de l'effet f^* alors a est considéré comme la cause de cet effet et $\mathcal{P}(b, f^*) = 0$.

Si un effet intervient systématiquement dans toute transition depuis la position initiale et jamais dans une autre position du jeu, il est considéré indépendant des actions. Quand tous les liens de causalité ont été vérifiés, si aucune action n'a été identifiée comme responsable d'un effet, le fluent correspondant est étiqueté comme indépendant des actions.

4.3 Fluents indépendants des actions

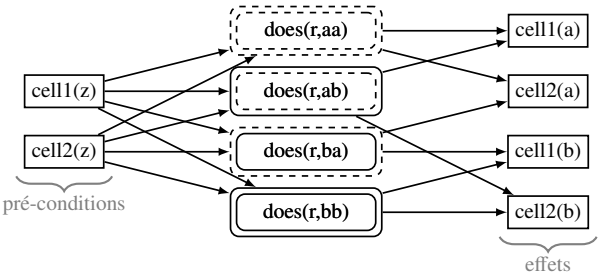
Pour identifier les liens existants entre les fluents indépendants des actions, nous utilisons le circuit logique. Pour chaque fluent f indépendant des actions, la sortie *next(f)* est marquée *vraie* puis le signal est rétro-propagé dans le circuit avec 4 état possibles (*indéfini*, *vrai*, *faux*, *vrai-et-faux*). Les fluents activés en entrée indiquent les prémisses de *next(f)* et leur valeur : prémisse utilisée telle quelle, dans une négation ou les deux.

Chaque prémisse g est ensuite vérifiée avec une logique à 3 états : si elle permet de changer la valeur de *next(f)* de *faux* à *vrai*, il s'agit bien d'une précondition de f, sinon, si elle force f à rester *vrai*, le fluent inverse $\neg g$ est responsable d'un effet négatif sur f.

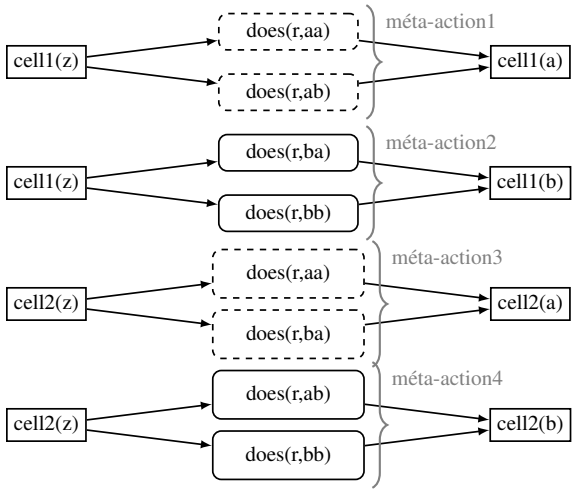
4.4 Traitement des actions composées

Pour identifier les méta-actions d'après la définition 8, nous utilisons les résultats de la section 4.2 et considérons que pour tout $\mathcal{P}(a, f^*) > 0$, a = *does(r,o)* est responsable de l'effet f^*.

Pour identifier chaque fluent h prémisse de *legal(r,o)*, nous utilisons le circuit logique. La sortie *legal(r,o)* est marquée *vraie* puis le signal est rétro-propagé dans le circuit avec 4 états possibles comme précédemment (§4.3).



(a) Identification des groupes d'actions avec une même précondition et responsables d'un même effet.



(b) Séparation des méta-actions

FIGURE 2 – Représentation graphique de l'identification de méta-actions. Les fluents de prédicat *cell1* appartiennent au premier sous-jeu, ceux de prédicat *cell2* au second. L'identification des méta-actions permet de supprimer les liens qui les unissent.

Chaque prémisse est ensuite vérifiée avec une logique à 3 états pour s'assurer qu'elle permet réellement d'obtenir *legal(r,o)* i.e. c'est une prémisse non-contradictoire.

Pour identifier chaque fluent g utilisé en conjonction avec une action a dans les prémisses de *next(f)*, nous marquons l'entrée *does(r,o)* à *vrai* et propageons le signal à travers le circuit sans tenir compte des portes logiques de manière à marquer chaque porte dépendant de cette entrée. A partir d'une des sorties *next(f)* activées, nous rétro-propageons le signal en utilisant différents marqueurs, pour étiqueter les portes représentant une conjonction entre l'action et une autre entrée (ce peut être une porte OU en amont d'une négation), et pour étiqueter les fluents en amont de ces conjonctions.

Nous effectuons finalement une comparaison des différents ensembles d'actions obtenus pour en déduire les méta-actions (fig.2).

4.5 Traitement des jeux en série

Dans les jeux en série, pour identifier les points de croisement à partir des fluents charnière (§9), nous construisons un *graphe causal G* dont les nœuds sont des actions et des fluents (positifs ou négatifs). Les relations logiques entre les nœuds sont représentées par des arcs orientés. Nous ajoutons un arc y $\rightarrow z$ dans G si :

— y est une action et z le résultat d'un effet dont y est responsable ;
— y est un fluent (peut-être indépendant des actions), prémisse de legal(r,p) avec $z = does(r, p)$;
— y est un fluent, z est une action et $y \wedge z$ est un prémisse de *next(f)* ou z est responsable de l'effet f^* ;
— y est un prémisse de *next(z)* ou z est un fluent indépendant des actions.

Un *fluent charnière* unique est un *point de croisement* si il n'existe pas deux arc $x \rightarrow y$ et $y \rightarrow x$ and C(G). La figure 3 représente une version très simplifié du graphe causal obtenu pour le jeu *Tictactoe Serial*. Le fluent *game1overLock* marque une limite nette entre deux parties distinctes du jeu et constitue un *point de croisement*.

Soit X_c l'ensemble des fluents charnière. Un ensemble $X_i \subset X_c$ est un *point de croisement potentiel*, si il existe dans C(G) un ensemble de nœuds Q tel que pour tout $x \in X_i$ et pour tout $q \in Q$ il existe un arc $x \rightarrow q$. Pour chaque X_i qui est potentiellement un point de croisement nous ajoutons un nœud o_i dans G. Nous utilisons alors le circuit pour identifier les relations logiques entre chaque nœud o_i et les autres.

Dans G, nous ajoutons un arc $y \rightarrow o_i$ pour chaque arc $y \rightarrow x$ avec $x \in X_i$ et nous ajoutons un arc $o_i \rightarrow y$ si :

— une action $y = does(r, p)$ est légale quand tous les $x \in X_i$ sont *vrais* et il existe un $x \in X_i$ prémisse de legal(r,p) ;
— *next(f)* peut être *vrai* quand tous les $x \in X_i$ sont *vrais* et il existe $x \wedge y$ prémisse de next(f) ou f^* est un effet de l'action y ;
— y est un fluent indépendant des actions et y^* est observé quand tous les fluents $x \in X_i$ sont *vrais*.

X_i est un point de croisement si il n'existe pas deux arcs $o_i \rightarrow y$ et $y \rightarrow o_i$ dans $C(G)$.

Un *point de croisement* est écarté si il a pour unique précondition un autre *point de croisement* ou un fluent de l'état initial ou si il inclut un autre *point de croisement*.

4.6 Construction du graphe de dépendances

Nous utilisons toutes les informations collectées précédemment pour construire le *graphe de dépendances* nécessaire à l'identification des sous-jeux. Nous ajoutons un nœud pour chaque fluent f et pour chaque méta-action M. Nous ajoutons un arc entre une méta-action $M(r, \mathcal{E}, \mathcal{N}, \mathcal{L})$ et un fluent f si :

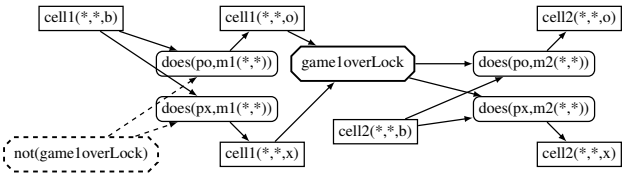


FIGURE 3 – Graphe causal extrêmement simplifié pour le jeu *Tictactoe Serial*. *m1*, *m2*, *po* et *px* représentent respectivement *mark1*, *mark2*, *playero* et *playerx*. Un ensemble de termes avec des arguments variables est représenté avec des jokers *. Une partie des liens est omise pour clarifier le graphe. *Game1overLock* est identifié comme *point de croisement*.

(1) $f \in \mathcal{L} \backslash C$ avec C l'ensemble des fluents *control* détecté (pour éviter de lier f aux méta-actions de différents sous-jeux).
(2) $f \in \mathcal{N} \backslash C'$ avec C' l'ensemble des fluents *control* détectés ou, à défaut, l'ensemble des fluents indépendants des actions.
(3) $f* \in \mathcal{E}$

Les arcs (1) et (2) sont associés à une pondération de 1. La pondération W des arcs (3) est la moyenne des probabilités d'effet associées à chaque action : $W = \sum_{0<i<N} \frac{\mathcal{P}(a_i, f^*)}{N}$ avec $a_i \in M(r, \mathcal{E}, \mathcal{N}, \mathcal{L})$.

Nous ajoutons également un arc entre les fluents indépendants des actions et leurs préconditions. Pour séparer les sous-jeux en série, nous supprimons les arcs entre chaque fluent d'un *point de croisement* et les fluents ou actions que ce *point de croisement* pré-conditionne.

Certaines actions peuvent ne pas avoir été testées durant les *playouts* et certains effets peuvent ne pas avoir été observés. Les actions et fluents correspondants sont considérés comme appartenant à tous les sous-jeux jusqu'à ce que plus de *playouts* apportent d'avantage d'informations.

4.7 Vérification et re-composition

Dans certains jeu, il n'existe pas de relation entre les différentes structures internes en dehors de l'évaluation du score. C'est le cas pour les cases de *Tictactoe*, les colonnes de *Connect Four* [9], ou les régions colorées de *Rainbow*. Inversement, dans un jeu comme *LightsOn Parallel*, toutes les lampes sont réunies en un seul sous-jeu car chaque action d'allumer une lampe a comme effet collatéral de laisser s'éteindre les autres. Cependant, nous aimerions séparer le jeu en groupes de 4 lampes car allumer toutes les lampes d'un des groupes est suffisant pour obtenir la victoire.

Pour résoudre ces deux problèmes, nous identifions des *sous-buts* en utilisant le circuit. Le sous-circuit permettant de calculer les sorties *goal* (évaluation du score) fait l'objet d'un tri topologique : chaque porte logique dont la sortie

9. *Connect Four* est connu en français sous le nom *Puissance 4*.

est utilisée comme entrée d'une autre porte est examinée en premier. Chaque porte p du sous-circuit est activée indépendamment des autres avec une valeur $o \in \{vrai, faux\}$ et, en utilisant une logique à 3 états pour diffuser le signal, l'état des sorties *goal* est examiné. Si une sortie représentant le score maximum pour un des joueurs est activée, alors la valeur o pour la porte p représente une *condition de victoire*. Sinon, si un score non nul peut néanmoins être obtenu (le score 0 est *faux*, un score supérieur à 0 est *vrai* ou bien la valeur $\neg o$ rend le score maximum impossible à atteindre), alors la valeur o pour la porte p est un sous-but. Chaque porte utilisant la sortie d'une porte p détectée comme *sous-but* ou *condition de victoire* est exclue de la recherche : ceci permet de collecter uniquement les conditions minimales pour obtenir un score non nul.

Si plusieurs sous-jeux dépendant des actions sont détectés, i.e. le *graphe de dépendances* présente plusieurs zones connexes comprenant des nœuds méta-action, nous vérifions si le jeu n'est pas sur-décomposé. Nous vérifions que chaque *sous-but* ou *condition de victoire* ne dépend pas de fluents placés dans différents sous-jeux. Dans le cas contraire, les sous-jeux sont regroupés en un. Cela permet de s'assurer que chaque sous-jeu, si ses fluents sont impliqués dans le calcul du score, apporte au moins une partie des points : il existe une fonction d'évaluation.

Si un seul sous-jeu dépendant des actions est détecté : il regroupe peut-être des sous-jeux permettant chacun d'obtenir le score maximum, mais liés par des effets collatéraux des actions. Pour vérifier si de tels sous-jeux existent, les liens dont le poids est inférieur au seuil Ω sont supprimés du graphe puis les *conditions de victoire* sont examinées pour vérifier si chaque sous-jeu identifié peut apporter la victoire à lui seul.

5 Expérimentations

Il n'existe pas de données expertes indiquant ce qu'est la décomposition attendue pour chacune des descriptions de jeu présentes dans les dépôts. D'après les définitions de la section 3, nous avons donc établi pour 597 jeux trouvés dans les dépôts *Base*, *Stanford* et *Dresden* de `http://games.ggp.org/` [10] le nombre de sous-jeux attendus avec séparément le nombre de sous-jeux dépendants des actions, indépendants des actions et pour chacun le nombre de sous-jeux inutiles [11]. Cependant, comme il est possible de partitionner les fluents d'un jeu en N sous-jeux de différentes manières, dans chacune de nos expériences, nous avons vérifié manuellement chaque décomposition signalant la détection de plus d'un sous-jeu pour nous assurer qu'elle corresponde à une décomposition experte humaine

10. Nous avons exclu les descriptions GDL-II et certaines descriptions trop mal formées pour être jouables.

11. Ces données sont disponibles sur simple demande au premier auteur.

5 kp	<1s	<10s	<30s	<1min	<3min	<10min	<30min	<1h
total	72	307	391	434	491	527	540	557

FIGURE 4 – Nombre de jeux pour lesquels le processus de décomposition est achevé dans le temps donné (pour 5k *playouts*).

et soit en accord avec les définitions 1 à 3.

Les expériences ont été réalisées sur une machine dotée d'un processeur Intel Core i7 2,7GHz avec 8Go de DDR3 à 1.6GHz. Les valeurs des seuils et ratios ont été ajustées empiriquement aux valeurs suivantes : $\Psi = 3$, $\Theta = 5$, $\Phi = 1.5$ et $\Omega = 0.1$. Ces paramètres sont peu sensibles à de légères variations. Par exemple la valeur de Ψ amène à ignorer les effets cooccurrents pendant les Ψ premières observations : la valeur de Ψ doit être assez haute pour écarter les coïncidences. Une valeur paranoïaque nécessitera juste un nombre plus important de *playouts* pour obtenir le même résultat.

Pour chaque jeu nous avons mesuré le temps nécessaire pour la construction du circuit, la réalisation de 5k *playouts* et le processus de décomposition. Les résultats présentés sur la figure 4 montrent que 70% des jeux peuvent être ainsi analysés en moins de 1 minute, temps compatible avec les contraintes des compétitions de *General Game Playing*. La majorité du temps est utilisée pour la création du circuit. Par exemple, pour le jeu *Blocker Parallel*, une fois le circuit créé, 35 secondes sont suffisantes pour réaliser les 5k simulations et la décomposition. Pour 40 jeux la décomposition n'a pas été obtenue en moins d'une heure, parmi eux 32 sont décomposables.

Le processus de décomposition peut être réitéré lorsque plus de simulations ont été effectuées. Sur la figure 5 nous évaluons le nombre de décompositions correctes obtenues après différents lots de 1k *playouts*. Nous constatons que 1k playouts sont suffisants pour décomposer correctement 87% des jeux. Le seul jeu sous-décomposé au bout de 10k playouts est *Simultaneous Win 2* pour lequel aucun sous-but (fonction d'évaluation) n'a pu être identifié pour les sous-jeux.

	1kp	2kp	3kp	4-6kp	7kp	8kp	9-10kp
under-decomp.	9	5	4	2	2	2	1
decomposed	521	525	527	529	530	532	533
over-decomp.	26	26	25	25	24	22	22

FIGURE 5 – Nombre de jeux correctement décomposés après chaque phase de 1k à 10k *playouts*.

Parmi les 597 jeux testés, 195 sont des jeux non décomposables. Cinq d'entre eux sont sur-décomposés au bout de 10k playouts. Les cas de sur-décomposition sont en majorité dus au manque d'information. Par exemple, pour les jeux *Snake* ou *Tron* certaines parties du plateau de jeu ne

sont pas explorées pendant les *playouts* et constituent un sous-jeu séparé. D'autres cas de sur-décomposition sont observés pour les jeux qui consistent à survivre pendant N étapes du jeu (*Queens*, *Max-Knights*) : une mauvaise action met fin au jeu. Dans ce cas, le rôle du sous-jeu principal est mal détecté : le compteur de coups est le seul sous-jeu jugé utile. Dans les jeux à coups simultanés comme *Point Grab* ou *Smallest*, chaque joueur est placé dans un sous-jeu distinct : les sous-buts identifiés ne font pas le lien entre les actions des deux joueurs.

Tous les jeux décomposables pour lesquels la décomposition n'a pu être obtenue en moins d'une heure, sont constitués d'un compteur de coups (*stepper*) associé à un sous-jeu dépendant des actions. Une détection ad-hoc des compteurs de coups permettrait de gagner du temps sur les jeux présentant cette structure particulière.

Des cas intéressants de décomposition ont été obtenus pour certains jeux qui ne semblent pas décomposables *à priori*. Par exemple le jeu *Snake Assemblit* présente la particularité d'avoir un plateau de jeu inutilement grand : la description du jeu prévoit un plateau de 50×50 cases quand seulement 6×6 sont réellement utilisées. Les cases inutiles sont détectées et placées dans un sous-jeu étiqueté comme inutile. Dans les jeux *Nonogram* (5×5 et 10×10) le statut de certaines cases est décidable indépendamment des autres : ces cases sont donc isolées dans des sous-jeux distincts (fig.6). Le reste du plateau est découpé en différents sous-jeux où le statut des cases peut être décidé indépendamment. Cette décomposition peut permettre de résoudre le jeu plus rapidement.

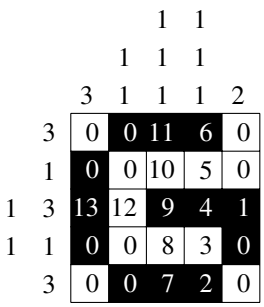


FIGURE 6 – Illustration de la décomposition obtenue pour le jeu *Nonogram5x5*. Le numéro dans chaque case représente le sous-jeu auquel elle appartient.

Nous avons comparé le résultat de notre décomposition avec ceux des travaux précédents : nous obtenons une décomposition correcte pour tous les jeux testés par nos prédécesseurs [12]. Aucune mesure du temps de décomposition n'est indiquée par [4] et [11]. La figure 7 met en parallèle nos résultats et les résultats obtenus précédemment avec une approche fondée sur l'analyse des règles [5]. Il faut noter que l'approche DNF, bien que plus rapide que la DNFD, est bien moins fiable et les heuristiques utilisées pour la détection des liens de causalité sont très faibles. La technique de décomposition présentée dans cet article permet, contrairement à cette précédente approche, d'obtenir une décomposition correcte pour *Chomp* et *Blocker Parallel* dans un délai inférieur à 1 heure.

Jeu	DNFD	DNF	stats
Blocker Parallel	>1hr	>1hr	≈48min
Asteroids	<1sec	<1sec	<2sec
EightPuzzle	<2sec	<2sec	≈3sec
Checkers	>1hr	<12min	≈28min
Breakthrough	<16min	<16min	≈14sec
Sheep and wolf	>1hr	<5sec	≈6sec
Tictactoe	≈1sec	<1sec	<1sec
Nineboardtictactoe	>1hr	<2sec	<17sec
Tictactoex9	>1hr	<5sec	≈11sec
Chomp	<1sec	<1sec	<2sec
Multiplehamilton	<1sec	<1sec	<2sec
Multipletictactoe	<10sec	<1sec	≈1sec
Blockerserial	<20min	<10min	≈3sec
Dualrainbow	≈1min	<8sec	≈6sec
Asteroidsparallel	<1sec	<1sec	≈2sec
Dualhamilton	<1sec	<1sec	<2sec
Dualhunter	<2sec	<2sec	<3sec
Asteroidsserial	<1sec	<1sec	<4sec
LightsOnParallel	<8min	<1sec	<1sec
LightsOnSimul4	<8min	<1sec	≈3sec
LightsOnSimultaneous	<8min	<1sec	≈3sec
Nim3	<2sec	<2sec	<2sec
Chinook	<14sec	<14sec	≈21sec
Double tictactoe dengji	>1hr	<1sec	<1sec
SnakeParallel	>1hr	<2sec	<7sec
TicTacToeParallel	>1hr	≈2sec	<2sec
Doubletictactoe	>1hr	<1sec	<1sec
TicTacHeaven	>1hr	<2sec	≈17sec
TicTacToeSerial	>1hr	<1sec	<1sec
ConnectFourSimultaneous	>1hr	<1sec	<2sec
DualConnect4	>1hr	<1sec	<2sec
Jointconnectfour	>1hr	<1sec	<2sec

FIGURE 7 – Comparaison des résultats de [5] (DNFD, DNF) avec ceux obtenus pour notre approche statistique (stats) pour 32 des 40 jeux qui étaient testés. Les résultats des deux approches sont identiques pour les 8 jeux restants.

6 Conclusion et perspectives

Nous avons présenté une technique de décomposition des jeux généraux que nous avons testé sur un large panel de jeux. Notre approche fondée sur la collecte d'informations durant des simulations apporte une solution au problème de l'identification des liens de causalité entre les actions et le changement des fluents. Nous avons montré que la détection de *méta-actions* pour traiter le cas spécifique des jeux parallèles à actions composées est possible sans calculer au préalable la forme normale disjonctive des règles. Pour identifier les jeux en série nous avons proposé le concept de *points de croisement* qui, contrairement aux approches précédentes, permet d'identifier un nombre quelconque de sous-jeux.

Nous avons testé notre approche sur 597 jeux

12. Nous n'avons pas testé *Double Crisscross2* dont la description n'est pas disponible dans les dépôts.

provenant des dépôts *Base*, *Stanford* et *Dresden* de `http://games.ggp.org/`. Nos résultats montrent qu'il est possible de transformer les règles en circuit logique, effectuer 5k *playouts* et obtenir une décomposition en moins d'une minute pour 70% d'entre eux. Nous avons également montré que 1k playouts sont suffisants pour décomposer correctement 87% de ces jeux par comparaison avec la décomposition attendue par un expert humain. Ces temps sont compatibles avec les conditions des compétitions de GGP. Le gain de performance espéré pour un joueur utilisant ces décompositions surpasse le temps nécessaire à la décomposition, de plus, cette décomposition peut-être calculée parallèlement à l'exploration du jeu global.

Les décompositions obtenues ont été calculées à partir de l'état initial du jeu. Quand plus d'informations sont disponibles suite à un lot supplémentaire de *playouts*, la décomposition peut être recalculée en réutilisant une grande partie des résultats, les tentatives ultérieures de décomposition sont donc plus rapides à calculer. A chaque étape du jeu, il est envisageable de détecter les fluents dont la valeur restera fixée pour le restant de la partie (les *latches*), de les éliminer du graphe et de détecter ainsi de nouvelles décompositions. Notre technique de décomposition est donc exploitable *online*, durant les *start-clock* et *play-clock* d'une compétition de GGP. Une première décomposition peut-être rapidement évaluée pendant la phase préparatoire, puis affinée à chaque étape du jeu lorsque plus d'informations sont disponibles i.e. à mesure que plus de *playouts* sont effectués et que les *latches* sont éliminés.

Nous avons ici envisagé la décomposition de jeux en N parties disjointes. Cependant, dans certains jeux comme *Tic-Block* [13] ou *Factoring-Mutually-Assured-Destruction* [14] des fluents sont réutilisés entre différentes parties du jeu. Une piste à développer serait la généralisation de notre approche pour décomposer des jeux en sous-jeux se chevauchant.

Les approches synthétisant les solutions de sous-jeux pour mieux résoudre un jeu global sont restreintes à certains types de jeux facilement décomposables de manière ad-hoc (puzzles ou jeux parallèles synchrones à 2 joueurs). Comme notre approche permet d'obtenir des décompositions robustes sur un large panel de jeux dans un temps compatible avec les délais imposés en compétition de GGP, notre premier objectif est d'implémenter un joueur utilisant le résultat de ces décompositions pour les différents types de jeux.

Références

[1] Cerexhe, Timothy, David Rajaratnam, Abdallah Saffidine et Michael Thielscher: *A Systematic Solution to the (De-)Composition Problem in General Game Playing*. Dans *Proceedings of ECAI*, pages 1–6, 2014.

[2] Genesereth, Michael R., Nathaniel Love et Barney Pell: *General Game Playing: Overview of the AAAI Competition*. AI Magazine, 26(2) :62–72, 2005.

[3] Günther, Martin: *Decomposition of Single Player Games*. Mémoire de maîtrise, TU-Dresden, 2007.

[4] Günther, Martin, Stephan Schiffel et Michael Thielscher: *Factoring General Games*. Dans *Proceedings of the IJCAI-09 Workshop on General Game Playing (GIGA'09)*, pages 27–33, 2009.

[5] Hufschmitt, Aline, Jean Méhat et Jean Noël Vittaut: *A General Approach of Game Description Decomposition for General Game Playing*. Dans *Proceedings of the IJCAI-16 Workshop on General Game Playing (GIGA'16)*, pages 23–29, 2016.

[6] Hufschmitt, Aline, Jean Noël Vittaut et Nicolas Jouandeau: *Statistical GGP Games Decomposition, En cours de soumission à Proceedings of the IJCAI-18 Workshop on Computer Games (CGW 2018)*. pages 1–19, juillet 2018.

[7] Love, Nathaniel, Timothy Hinrichs, David Haley, Eric Schkufza et Michael Genesereth: *General Game Playing : Game Description Language Specification*. Rapport technique LG-2006-01, Stanford University, 2008.

[8] Vittaut, Jean-Noël et Jean Méhat: *Fast Instantiation of GGP Game Descriptions Using Prolog with Tabling*. Dans *Proceedings of ECAI*, pages 1121–1122, 2014.

[9] Vittaut, Jean Noël: *LeJoueur : un programme de General Game Playing pour les jeux à information incomplète et/ou imparfaite*. Thèse de doctorat, Université Paris 8, 2017.

[10] Zhao, Dengji: *Decomposition of Multi-Player Games*. Mémoire de maîtrise, TU-Dresden, 2009.

[11] Zhao, Dengji, Stephan Schiffel et Michael Thielscher: *Decomposition of Multi-Player Games*. Dans *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 475–484, 2009.

13. *Tic-Block* est constitué d'une partie de *Tictactoe* suivie d'une partie de *Blocker*. Le plateau de jeu est réutilisé et agrandi d'une partie à l'autre.

14. *Factoring-Mutually-Assured-Destruction* est composé de 9 jeux de *LightsOn* joués en série avec recouvrement des groupes de lampes.