Une classe traitable de problèmes de planification temporelle

Martin Cooper, Frédéric Maris, Pierre Régnier, Florian Franc

Une classe traitable de problèmes de Planification temporelle

Martin C. Cooper* **Frédéric Maris*** **Pierre Régnier*** **Florian Franc***

IRIT, Université de Toulouse
Toulouse, France
{cooper, maris, regnier}@irit.fr

Résumé

Cet article présente une classe de problèmes de planification temporelle solubles en temps polynomial. Ce résultat découle de deux hypothèses. Nous supposons d'abord que les sous-buts ne peuvent être établis que par une action unique, ce qui nous permet de déterminer rapidement les actions qui sont nécessaires dans tous les plans. Nous supposons également que les sous-buts sont monotones, ce qui nous permet d'exprimer la planification comme une instance de STP$^{\neq}$ (Simple Temporal Problem, difference constraints). Notre classe contient des problèmes temporellement expressifs, ce que nous illustrons avec un exemple de planification de processus chimique.

1 Introduction

La planification est un domaine de l'IA qui, dans le cas général, n'est pas soluble en temps polynomial (on dit encore non traitable) [11]. [5] montre en particulier que la planification propositionnelle est PSPACE-complète.

Malgré ces résultats généraux peu encourageants, de nombreux travaux ont été réalisés pour étudier la complexité de la planification non-optimale et optimale des benchmarks classiques. [18],

[19], [30] ont ainsi montré qu'un bon nombre de ces problèmes pouvaient être résolus par des procédures simples s'exécutant en temps polynomial. Les planificateurs FF [20] et eCPT [31] ont aussi prouvé (bien que de manière empirique) que le nombre de benchmarks qui pouvaient être résolus sans recherche devait être bien plus important.

A partir des travaux de [1] sur le formalisme de planification SAS (Simplified Action Structure), un bon nombre d'études ont également été réalisées afin de mettre en évidence des classes de problèmes de planification traitables. Une bonne partie des résultats obtenus est basée sur des restrictions syntaxiques qui portent sur l'ensemble des opérateurs [5], [2], [11], [25]. Un autre ensemble de travaux utilise la structure sous-jacente des problèmes de planification qui peut être mise en évidence grâce au graphe de causalité [27]. En imposant des restrictions sur la structure de ce graphe, on peut exhiber des classes traitables [23], [24], [25], [32], [10], [3], [4], [18],

[19], [22], [16], [17], [15], [26]. Un cadre unificateur permettant de classifier la complexité de la planification sous restrictions sur le graphe de causalité est donné dans [6].

Malgré ces nombreux résultats, les hypothèses de la planification classique (monde fini, statique, actions instantanées, déterministes…), sont bien trop restrictives pour permettre de modéliser des applications réelles. Pour cela, il est nécessaire d'utiliser un cadre de planification temporelle afin de pouvoir formaliser les relations temporelles entre actions comme des contraintes temporelles. Cependant, dans le cadre temporel de PDDL 2.1 [9], [12], tester l'existence d'un plan valide devient un problème EXPSPACE-complet [29]. La complexité PSPACE-complète de la planification classique ne peut être préservée que lorsque différentes instances d'une même action ne peuvent se chevaucher.

Dans cet article, nous présentons un sous-problème de la planification temporelle soluble en temps polynomial. A notre connaissance, aucun travail antérieur n'a spécifiquement abordé cette question.

L'article est organisé ainsi : la Section 2 présente notre cadre temporel. La section suivante introduit ensuite la notion de monotonie des fluents (propositions atomiques). La Section 4 montre comment déterminer certains fluents monotones. La Section 5 donne un exemple d'un problème de planification temporelle (Temporal Chemical Process Planning) soluble en temps polynomial. Toutes

* Les auteurs sont soutenus par le Projet ANR-10-BLAN-0210.

les solutions de cet exemple nécessitent la concurrence des actions. Les Sections 6 et 7 concluent et présentent nos perspectives de recherche.

2 Planification Temporelle

Nous étudions la planification temporelle propositionnelle dans un langage basé sur les aspects temporels de PDDL 2.1. Un *fluent* est une proposition atomique positive ou négative. Comme dans le langage PDDL 2.1, nous considérons que les changements de valeur des fluents sont instantanés mais que les conditions sur leurs valeurs peuvent être imposées sur un intervalle. Une *action a* est un quadruplet <Cond(a), Add(a), Del(a), Constr(a)>, où l'ensemble des conditions Cond(a) est l'ensemble des fluents qui doivent nécessairement être vrais pour que a soit exécutée, l'ensemble des ajouts Add(a) est l'ensemble des fluents qui sont établis par a, l'ensemble des retraits Del(a) est l'ensemble des fluents qui sont détruits par a, et l'ensemble des contraintes Constr(a) est l'ensemble des contraintes entre les instants relatifs aux événements qui interviennent durant l'exécution de a. Un événement correspond à l'une des quatre possibilités suivantes : l'établissement ou la destruction d'un fluent par une action a, ou le début ou la fin d'un intervalle sur lequel un fluent est requis par a. En PDDL 2.1, les événements ne peuvent intervenir qu'en début ou fin des actions, mais nous relaxons cette hypothèse pour qu'ils puissent survenir à tout instant à condition que les contraintes Constr(a) soient satisfaites.

Nous utilisons la notation $a \rightarrow f$ pour dénoter l'événement qui correspond à l'établissement du fluent f par l'action a, la notation $a \rightarrow \neg f$ pour dénoter l'événement qui correspond à la destruction du fluent f par l'action a, et les notations $f \vert\rightarrow a$ et $f \rightarrow\vert a$, respectivement, pour dénoter le début et la fin de l'intervalle sur lequel a nécessite la condition f. Si f est déjà vrai (respectivement faux) quand l'événement $a \rightarrow f$ ($a \rightarrow \neg f$) intervient, nous considérons toujours que a établit (détruit) f. Un plan temporel peut contenir plusieurs instances de la même action, mais puisque la plupart des plans temporels étudiés dans cet article contiennent au plus une instance de chaque action, pour simplifier les notations, nous ne ferons la distinction entre actions et instances d'actions que lorsque cela sera absolument nécessaire. Nous utilisons la notation τ(E) pour représenter l'instant auquel un événement E intervient dans un plan.

Pour une action a donnée, Events(a) représente les différents événements qui le définissent, à savoir ($a \rightarrow f$) pour tout f dans Add(a), ($a \rightarrow \neg f$) pour tout f dans Del(a), ($f \vert\rightarrow a$) et ($f \rightarrow\vert a$) pour tout f dans Cond(a). La définition d'une action a contient des contraintes Constr(a) sur les instants relatifs aux événements de Events(a). Comme en PDDL 2.1, nous considèrerons que le délai entre événements dans Events(a) n'est pas nécessairement fixé et que Constr(a) correspond à des contraintes d'intervalles sur des paires d'événements, telles que $\tau(f \rightarrow\vert a) - \tau(f \vert\rightarrow a) \in [\alpha, \beta]$ pour des constantes α,β. Nous utilisons $[\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$ pour dénoter l'intervalle des valeurs possibles pour la distance relative entre les événements E_1,E_2 dans l'action a. Un délai fixe entre les événements $E_1,E_2 \in$ Events(a) peut aussi être modélisé en choisissant $\alpha_a(E_1,E_2) = \beta_a(E_1,E_2)$. Nous introduisons maintenant deux contraintes de base que tous les plans temporels doivent vérifier.

Contraintes inhérentes sur l'ensemble des actions A : pour toute $a \in$A, a satisfait Constr(a), i.e. pour toute paire d'événements $E_1,E_2 \in$ Events(a), $\tau(E_1) - \tau(E_2) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$.

Contraintes d'effets contradictoires sur l'ensemble des actions A : pour toutes les actions $a_i,a_j \in$A, pour tout fluent positif $f \in \text{Del}(a_i) \cap \text{Add}(a_j)$, $\tau(a_i \rightarrow \neg f) \neq \tau(a_j \rightarrow f)$.

Définition 1. Un *problème de planification temporelle* <I,A,G> est défini par un ensemble d'actions A, un état initial I et un but G, où I et G sont des ensembles de fluents.

Notation: Si A est un ensemble d'instances d'actions, alors Events(A) est l'union des ensembles Events(a) (pour toutes les instances d'actions $a \in$ A).

Définition 2. P = <A,τ>, avec A un ensemble d'instances d'actions $\{a_1,...,a_n\}$ et τ une fonction réelle sur Events(A), est un *plan temporel* pour le problème <I,A′,G> si
(1) A $\subseteq$ A′, et
(2) P satisfait les contraintes inhérentes et d'effets contradictoires sur A;

et si, lorsque P est executé (i.e. les fluents sont établis ou détruits aux instants donnés par τ), en démarrant de l'état initial I :
(3) pour toute $a_i \in$ A, chaque $f \in$ Cond(a_i) est vrai lorsqu'il est requis, et
(4) tous les fluents du but $g \in$ G sont vrais à la fin de l'exécution de P.

Etudions maintenant plus en détail le type de contraintes que nous imposons sur les instants relatifs aux événements à l'intérieur d'une action.

Définition 3. Une *contrainte d'intervalle* C(x,y) sur des variables réelles x,y est une contrainte de la forme x-y $\in$ [a,b] où a,b sont des constantes réelles.

Définition 4. [21] Une contrainte binaire C(x,y) est *min-closed* si pour toute paire de valeurs (x_1,y_1), (x_2,y_2) qui satisfait C, $(\min(x_1,x_2), \min(y_1,y_2))$ satisfait également C.

Lemme 1. Soit A=$\{a_1,...,a_n\}$ un ensemble d'actions et A′ un ensemble d'instances d'actions dans lequel chaque action a_i (i=1,...,n) intervient $t_i \geq 1$ fois. Soit τ une fonction réelle sur l'ensemble des événements dans A′. Pour chaque E $\in$ Events(a_i), soit E[j] (j=1,...,t_i) l'occurrence de l'événement E dans la $j$ème instance de a_i. Pour $i \in$ {1,...,n}, on définit la fonction réelle $\tau_{\min}$ sur l'ensemble

des événements dans l'ensemble des actions A par : $\tau_{min}(E) = \min\{\tau(E[j]) \mid j=1,...,t_i\}$. Si τ satisfait les contraintes inhérentes sur A′, alors τ_{min} satisfait les contraintes inhérentes sur A.

Preuve : Toutes les contraintes d'intervalle sont min-closed [21]. En appliquant la définition de min-closed t_i-1 fois, pour chaque action a_i, nous pouvons déduire que si τ satisfait une contrainte d'intervalle sur chacune des t_i instances de a_i, alors τ_{min} satisfait cette contrainte sur l'action a_i. En d'autres termes, pour toutes les paires d'événements E_1,E_2 dans Events(a_i), si $\tau(E_1[j]) - \tau(E_2[j]) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$ pour $j=1,...,t_i$, alors $\tau_{min}(E_1) - \tau_{min}(E_2) \in [\alpha_a(E_1,E_2), \beta_a(E_1,E_2)]$. Donc si τ satisfait les contraintes inhérentes sur A′, alors τ_{min} satisfait les contraintes inhérentes sur A.

Définition 5. Un problème de planification temporelle <I,A,G> est *positif* s'il n'y a aucun fluent négatif ni dans les conditions des actions, ni dans le but G.

Dans cet article, nous considèrerons uniquement des problèmes de planification temporelle positifs <I,A,G>. Il est bien connu que tout problème de planification peut être transformé en un problème positif équivalent en temps linéaire par introduction d'un nouveau fluent *notf* pour chaque fluent négatif *f* [14]. Sous cette hypothèse, G et Cond(*a*) (pour toute action *a*) sont composés de fluents positifs. Par convention, Add(*a*) et Del(*a*) sont aussi composés exclusivement de fluents positifs. L'état initial I peut contenir des fluents négatifs.

Pour définir notre classe traitable de problèmes de planification, nous aurons besoin de définir la notion d'ensemble d'actions *establisher-unique*. Cette notion est équivalente à celle de *post-unique* de la planification SAS^+ [25] restreinte aux variables booléennes mais généralisée pour qu'elle s'applique à un sous-ensemble spécifique des fluents positifs. Dans la section suivante, nous l'appliquons à l'ensemble des sous-buts S, qui sont les fluents essentiels à la réalisation du but.

Définition 6. Un ensemble d'actions $A=\{a_1,...,a_n\}$ est *establisher-unique* relativement à un ensemble de fluents positifs S si pour tout $i \neq j$, $\text{Add}(a_i) \cap \text{Add}(a_j) \cap S = \emptyset$, i.e. aucun fluent de S ne peut être établi par deux actions distinctes de A.

Si un ensemble d'actions est establisher-unique relativement à l'ensemble des sous-buts d'un problème, alors nous pouvons déterminer en temps polynomial l'ensemble des actions qui sont nécessairement présentes dans un plan temporel. Pour produire un plan temporel valide il reste encore à déterminer combien de fois chaque action doit intervenir et à ordonner ces instances d'actions.

3 Planification Monotone

Dans cette section, nous introduisons la propriété de monotonie des fluents. Avec la notion d'ensemble d'actions establisher-unique, la monotonie des fluents est une condition suffisante pour montrer qu'il existe un algorithme polynomial pour la planification temporelle.

Définition 7. Un fluent *f* est *–monotone* (relativement à un probleme de planification temporelle positif <I,A,G>) si, après avoir été détruit, *f* n'est jamais ré-établi dans aucun plan temporel pour <I,A,G>. Ce fluent est *+monotone* si, après avoir été établi, il n'est jamais détruit dans aucun plan temporel pour <I,A,G>. Un fluent est *monotone* (relativement à <I,A,G>) s'il est *+monotone* ou *–monotone* (relativement à <I,A,G>).

Exemples : Dans des contextes évidents, les fluents suivants sont –monotones : vivante (une personne), jamais-utilisée, allumée (une allumette), prêt-à-manger (un plat). De même, les fluents suivants sont tous +monotones : non-vivante, brulée, dissous, explosé (un ballon), mangé, cuisiné, diplômé, né et éteint. La manière de détecter la monotonie des fluents est discutée dans la Section 4.

Notation : Si A est un ensemble d'actions, nous utilisons la notation Del(A) pour représenter l'union des ensembles Del(*a*) (pour toutes les actions $a \in A$). Add(A), Cond(A), Constr(A) sont définies de façon similaire.

Le lemme suivant résulte trivialement de la Définition 7.

Lemme 2. Si $f \notin \text{Add}(A) \cap \text{Del}(A)$, alors *f* est à la fois –monotone et +monotone relativement au problème de planification temporelle positif <I,A,G>.

Nous introduisons maintenant trois autres ensembles de contraintes qui ne s'appliqueront qu'aux fluents monotones :

Contraintes de –autorisation sur les fluents positifs *f* et l'ensemble d'actions A : pour toutes les actions $a_i,a_j \in A$, si $f \in \text{Del}(a_j) \cap \text{Cond}(a_i)$, alors $\tau(f \rightarrow\mid a_i) < \tau(a_j \rightarrow \neg f)$.

Contraintes de +autorisation sur les fluents positifs *f* et l'ensemble d'actions A : pour toutes les actions $a_i,a_j \in A$, si $f \in \text{Del}(a_j) \cap \text{Add}(a_i) \cap (\text{Cond}(A) \cup G)$, alors $\tau(a_j \rightarrow \neg f) < \tau(a_i \rightarrow f)$.

Contraintes de causalité sur les fluents positifs *f* et l'ensemble d'actions A : pour toutes les actions $a_i,a_j \in A$, si $f \in (\text{Cond}(a_j) \cap \text{Add}(a_i)) \backslash I$ alors $\tau(a_i \rightarrow f) < \tau(f \mid\rightarrow a_j)$.

Définition 8. Un plan temporel pour un problème de planification temporelle positif <I,A,G> est *monotone* si chaque paire d'actions (dans A) satisfait les contraintes de +autorisation pour tous les fluents +monotones et les contraintes de –autorisation pour tous les fluents –monotones.

Le lemme suivant résulte directement de la définition de la monotonie et du fait qu'un fluent ne peut être simultanément établi et détruit dans un plan temporel.

Lemme 3. Supposons qu'un fluent positif f est monotone relativement à un problème de planification temporelle positif <I,A,G> et que les actions $a_i,a_j \in$ A sont telles que $f \in$ Add(a_i) ∩ Del(a_j). Si f est +monotone relativement à ce problème, alors dans tous les plans temporels contenant à la fois a_i et a_j, $\tau(a_j \rightarrow \neg f) < \tau(a_i \rightarrow$ f). Si f est –monotone relativement à ce problème, alors dans tous les plans temporels contenant à la fois a_i et a_j, $\tau(a_i \rightarrow f) < \tau(a_j \rightarrow \neg f)$.

Proposition 1. Si chaque fluent dans Cond(A) est monotone relativement à un problème de planification temporelle positif <I,A,G>, alors tous les plans temporels pour <I,A,G> sont monotones.

Preuve : Soit P un plan temporel. Considérons d'abord un fluent positif –monotone f. Nous devons montrer que les contraintes de –autorisation sont satisfaites pour f dans P, i.e. que f n'est pas détruit avant qu'il soit requis dans P (ou au même instant). Mais ceci doit être le cas puisque, une fois détruit, f ne peut être ré-établi. Considérons ensuite un fluent positif +monotone f. Par le Lemme 3, les contraintes de +autorisation sont satisfaites pour f dans P.

Définition 9. Etant donné un problème de planification temporelle <I,A,G>, l'*ensemble des sous-buts* est le sous-ensemble minimal SG de Cond(A) ∪ G qui satisfait :
(1) G ⊆ SG et
(2) pour toute action a ∈ A, si Add(a) ∩ (SG\I) ≠ ∅, alors Cond(a) ⊆ SG.

L'*ensemble d'actions réduit* est $A^r = \{a \in A \mid \text{Add}(a) \cap (SG\backslash I) \neq \emptyset\}$.

Il est évident que nous pouvons déterminer SG et A^r en temps polynomial et le résultat est unique. Si chaque fluent dans Cond(A^r) ∪ G est monotone, nous pouvons dire qu'un plan P pour le problème de planification temporelle <I,A,G> *satisfait les contraintes d'autorisation* si chaque fluent –monotone satisfait les contraintes de –autorisation et chaque fluent +monotone satisfait les contraintes de +autorisation (en partant de l'hypothèse que nous savons, pour chaque fluent $f \in$ Cond(A^r) ∪ G, si f est + ou –monotone).

Théoreme 1. Etant donné un problème de planification temporelle positif <I,A,G>, où A est un ensemble d'actions tel que Constr(A) sont des contraintes d'intervalle, soit SG et A^r, respectivement, l'ensemble des sous-buts et l'ensemble d'actions réduit. Si l'ensemble d'actions A^r est establisher-unique relativement à SG, chaque fluent dans Cond(A^r) ∪ G est monotone relativement à <I,A^r,G> et chaque fluent dans I ∩ (Cond(A^r) ∪ G) est –monotone relativement à <I,A^r,G>, alors <I,A,G> a un plan temporel P si est seulement si :
(1) G ⊆ (I\Del(A^r)) ∪ Add(A^r)
(2) Cond(A^r) ⊆ I ∪ Add(A^r)
(3) tous les fluents $g \in$ G ∩ Del(A^r) ∩ Add(A^r) sont +monotones relativement à <I,A^r,G>
(4) l'ensemble des contraintes d'autorisation, inhérentes, d'effets contradictoires et de causalité peut être satisfait sur l'ensemble d'actions A^r.

Preuve : (⇒) A^r est l'ensemble des actions qui établissent les sous-buts f dans SG\I. SG = Cond(A^r) ∪ G. Puisque A^r est establisher-unique relativement à SG, chaque sous-but $f \in$ SG\I est établi par une action unique. Chaque action dans A^r doit donc intervenir dans le plan P. Par ailleurs, (Add(A)\Add(A^r)) ∩ (Cond(A^r)\I) = ∅ par la Définition 9. Il résulte que (2) est une condition nécessaire pour qu'un plan temporel P existe.

Soit P′ une version de P dans laquelle nous ne conservons que les actions de A^r. P′ est également un plan temporel valide puisqu'aucune condition des actions dans P′ et aucun but dans G ne sont établis par une des actions éliminées de P, sauf peut-être s'il sont également dans I. Ces fluents $f \in$ I ∩ (Cond(A^r) ∪ G) sont –monotones, par hypothèse, et ne peuvent donc être établis dans P après avoir été détruits. Il en résulte que tous ces f sont déjà vrais lorsqu'ils sont établis dans P par toute action dans A\ A^r.

(1) est clairement une condition nécessaire pour que P′ soit un plan valide. Considérons $g \in$ G ∩ Del(A^r) ∩ Add(A^r). Par le Lemme 3, nous pouvons déduire que g ne peut être –monotone, puisque g est vrai à la fin de l'exécution de P′. (3) est ainsi vérifiée. Soit P_{min}=<A^r,τ_{min}> la version du plan temporel P′=<A^r,τ> dans lequel nous ne conservons qu'une seule instance de chaque action $a_i \in A^r$ (et pas d'instances des actions dans A\A^r) et τ_{min} est définie à partir de τ en prenant la première instance de chaque événement dans Events(a_i), pour chaque action $a_i \in A^r$, comme décrit dans l'énoncé du Lemme 1. Nous allons montrer que P_{min} satisfait les contraintes d'autorisation, inhérentes, d'effets contradictoires et de causalité.

D'après la Proposition 1, le plan temporel P′ doit être monotone. Puisque P′ est monotone et d'après la définition d'un plan temporel, les contraintes d'autorisation sont toutes satisfaites. P′ doit également, d'après la définition d'un plan temporel, satisfaire les contraintes inhérentes et d'effets contradictoires. Il résulte du Lemme 1 que P_{min} satisfait également les contraintes inhérentes. Puisque les événements dans P_{min} sont simplement un sous-ensemble des événements dans P′, P_{min} satisfait nécessairement les contraintes d'autorisation et les contraintes d'effets contradictoires.

Considérons un fluent positif $f \in$ (Cond(a_j) ∩ Add(a_i))\I, où a_i, $a_j \in A^r$. Puisque $a_j \in A^r$, nous savons que Add(a_j) ∩ (SG\I) ≠ ∅ et donc que Cond(a_j) ⊆ SG, d'après la définition de l'ensemble des sous-buts SG. Puisque $f \in$ Cond(a_j) nous pouvons déduire que $f \in$ SG. En fait, $f \in$ SG\I puisque nous supposons que $f \notin$ I. Il en résulte que si $f \in$ Add(a) pour une action $a \in$ A, alors $a \in A^r$. Mais nous savons que A^r est establisher-unique (relativement à SG). Donc, puisque $f \in$ Cond(a_j) ⊆ Cond(A^r) et f

$\in$ Add(a_i), f peut être établi par la seule action a=a_i dans A. Puisque $f \notin$ I, le premier établissement de f par une instance de a_i doit intervenir dans P′ avant que f soit requis par n'importe quelle instance de a_j. Il en résulte que les contraintes de causalité sont satisfaites par f dans P_{min}.

($\Leftarrow$) Supposons que (1) et (2) sont satisfaites par A^r. Soit P une solution à l'ensemble des contraintes d'autorisation, inhérentes, d'effets contradictoires et de causalité sur A^r. Une solution à ces contraintes utilise chaque action dans A^r (en fait, elle utilise chaque action exactement une fois puisque elle assigne un instant de démarrage à chaque action dans A^r). Considérons un fluent quelconque $g \in$ G. Par (1), $g \in (I \backslash Del(A^r)) \cup Add(A^r)$. Si $g \notin Del(A^r)$, alors g est nécessairement vrai à la fin de l'exécution de P. D'un autre côté, si $g \in Del(a_j)$ pour certaines actions $a_j \in A^r$, alors par (1) il y a nécessairement une action $a_i \in A^r$ qui établit g. Alors, par (3) g est +monotone. Puisque P satisfait les contraintes de +autorisation pour g, a_i établit g après toutes les destructions de g. Il en résulte que g est vrai à la fin de l'exécution de P.

Considérons des fluents –monotone $f \in Cond(a_j)$ où $a_j \in A^r$. Puisque les contraintes de –autorisation sont satisfaites pour f dans P, f peut seulement être détruit dans P après qu'il soit requis par a_j. Il ne reste donc plus à montrer que f était soit vrai dans l'état initial I ou qu'il a été établi à un instant qui précède celui auquel il est requis par a_j. Par (2), $f \in I \cup Add(A^r)$, et nous n'avons besoin de considérer que le cas dans lequel $f \notin$ I mais $f \in Add(a_i)$ pour une action $a_i \in A^r$. Puisque P satisfait les contraintes de causalité, $\tau(a_i \rightarrow f) < \tau(f \rightarrow a_j)$ et, durant l'exécution de P, f est donc vrai quand il est requis par l'action a_j.

Considérons un fluent $f \in Cond(a_j)$, où $a_j \in A^r$, qui n'est pas –monotone. Par les hypothèses du théorème, f est nécessairement +monotone et $f \notin$ I. Considérons d'abord le cas $f \notin Del(A^r) \cap Add(A^r)$. Par le Lemme 2, f est –monotone ce qui contredit notre hypothèse. Donc $f \in Del(a_k) \cap Add(a_i)$, pour des actions a_i, $a_k \in A^r$, et rappelons que $f \notin$ I. Puisque les contraintes de +autorisation sont satisfaites pour f dans P, toute destruction de f intervient avant que f soit établi par a_i. Il résulte alors des contraintes de causalité que la condition f sera vraie quand elle sera requise par a_j pendant l'exécution de P.

Théorème 2. Soit Π^{U+M} la classe des problèmes de planification temporelle positifs <I,A,G> dans lesquels A est establisher-unique relativement à Cond(A) $\cup$ G, tous les fluents dans Cond(A) $\cup$ G sont monotones relativement à <I,A,G> et tous les fluents dans I $\cap$ (Cond(A) $\cup$ G) sont –monotones relativement à <I,A,G>. Alors Π^{U+M} peut être résolue en temps $O(n^3)$ et en espace $O(n^2)$, où n est le nombre total d'événements dans les actions de A. En effet, nous pouvons même trouver (avec la même complexité) un plan temporel avec un nombre minimum d'instances d'actions.

Preuve : Ceci résulte presque directement du Théorème 1 et du fait que l'ensemble des contraintes d'autorisation, inhérentes, d'effets contradictoires et de causalité sont $STP^{\neq}$ [28]. Une instance de $STP^{\neq}$ peut être résolue en temps $O(n^3+k)$ et en espace $O(n^2+k)$ [13], où n est le nombre de variables et k le nombre d'inéquations (i.e. les contraintes de la forme $x_j - x_i \neq d$). Ici, les seules inéquations sont les contraintes d'effets contradictoires dont le nombre est au plus n^2, d'où $k=O(n^2)$. Par ailleurs, la calcul de SG et de A^r est clairement en $O(n^2)$.

La notion d'establisher-unique nous indique exactement quelles actions doivent appartenir à un plan temporel. Puis, (comme le montre la preuve du Théorème 1) les hypothèses de monotonie impliquent que nous n'avons besoin que d'une seule instance de chacune de ces actions. Il en résulte alors trivialement que nous résolvons la version optimale du problème de planification temporelle, dans lequel le but est de trouver un plan temporel avec un nombre minimal d'instances d'actions, en résolvant l'ensemble des contraintes d'autorisation, inhérentes, d'effets contradictoires et de causalité.

4 Détection de la Monotonie des Fluents

Une classe Π d'instances de problèmes NP-difficiles est généralement considérée comme traitable si elle satisfait deux conditions : (1) il existe un algorithme polynomial en temps pour résoudre Π, et (2) il existe un algorithme polynomial en temps pour reconnaître Π. Détecter si toutes les actions sont establisher-unique est clairement polynomial en temps. D'un autre côté, notre définition très générale de la monotonie des fluents implique que ce n'est pas le cas pour déterminer si des fluents sont monotones.

Théorème 3. Déterminer si un fluent d'un problème de planification temporelle <I,A,G> est monotone est PSPACE-difficile si le chevauchement d'instances d'une même action n'est pas permis dans les plans et EXPSPACE-complet si le chevauchement d'instances d'une même action est permis.

Preuve : Remarquons que si <I,A,G> n'a pas de solution, alors tous les fluents sont trivialement monotones par la Définition 7, puisqu'ils ne sont jamais établis ni détruits dans aucun plan. Il est suffisant d'ajouter deux nouveaux fluents buts f_1 et f_2 et deux nouvelles actions instantanées à A : a_1 qui ajoute simplement f_1 et a_2 qui a f_1 comme condition, ajoute f_2 et détruit f_1 (a_1 et a_2 étant indépendantes de tous les autres fluents) à tout problème <I,A,G> : f_1 est monotone si et seulement si le problème résultant n'a aucun plan temporel. Le théorème résulte alors du fait que tester l'existence d'un plan temporel pour un problème de planification temporelle <I,A,G> est PSPACE-difficile si le chevauchement d'instances d'une même action n'est pas permis dans les plans et EXPSPACE-complet si le chevauchement d'instances d'une même action est permis [29].

Nous pouvons néanmoins détecter la monotonie de certains fluents en temps polynomial. Il y a plusieurs façons de démontrer qu'un fluent est monotone. Dans cette section nous donnons quelques exemples qui donnent lieu à des algorithmes polynomiaux d'ordre faible. Etant donné le Théorème 3, nous affirmons clairement que nous ne pouvons détecter tous les fluents monotones avec ces règles. L'ensemble des problèmes de planification temporelle dont les fluents peuvent être prouvés +monotone ou –monotone avec les règles données dans cette section, comme requis dans les conditions du Théorème 2, représentent une classe traitable, puisqu'elle peut à la fois être reconnue et résolue en temps polynomial.

Notre première règle reformule simplement le Lemme 2 de la section précédente.

Règle 1 : S'il n'existe pas d'actions a, $a' \in A^r$ telles que $f \in \text{Del}(a) \cap \text{Add}(a')$ alors f est à la fois +monotone et –monotone.

Il peut également être possible de montrer qu'un fluent est monotone à cause de ses liens avec d'autres fluents, déjà connus pour être monotones. Considérons un exemple simple d'un problème de planification temporelle avec deux fluents *item_in_shop*, *item_for_sale* et uniquement deux actions qui ont ces fluents comme condition ou effet : l'action DISPLAY_ITEM qui a pour condition *item_in_shop* et établit *item_for_sale*, et l'action SELL_ITEM qui a pour condition *item_for_sale* et détruit à la fois *item_for_sale* et *item_in_shop*. Pour simplifier, supposons que les deux actions sont instantanées. Le fluent *item_in_shop* est –monotone puisqu'il ne peut jamais être établi (il peut être présent ou pas dans l'état initial). Nous ne pouvons pas appliquer la Règle 1 pour déduire la monotonie de *item_for_sale*, puisqu'il peut à la fois être établi et détruit. Cependant, puisque *item_in_shop* est –monotone, il est clair que DISPLAY_ITEM ne peut être exécutée après SELL_ITEM. Il résulte alors que *item_for_sale* est –monotone. Nous formalisons cette idée dans les deux règles suivantes.

Règle 2 : Supposons que l'ensemble réduit d'actions A^r est establisher-unique relativement à l'ensemble des sous-buts SG (comme défini par la Définition 9) et soit a_f l'unique action qui établit le fluent $f \in$ SG. Si pour toute $a \in A^r$ telle que $f \in \text{Del}(a)$:

- ∃ un fluent –monotone $p \in \text{Del}(a) \cap \text{Cond}(a_f)$ tel que $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f)$, ou
- ∃ un fluent –monotone $p \in \text{Del}(a) \cap \text{Add}(a_f)$ tel que $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f)$, ou
- ∃ un fluent +monotone $p \in \text{Add}(a) \cap \text{Del}(a_f)$ tel que $\tau(a \rightarrow p) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f)$, ou
- ∃ un fluent +monotone $p \in (\text{Cond}(a) \cap \text{Del}(a_f))\backslash I$ tel que $\tau(p \rightarrow\mid a) - \tau(a \rightarrow \neg f) \leq \tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f)$,

alors f est –monotone.

Règle 3 : Supposons que l'ensemble réduit d'actions A^r est establisher-unique relativement à l'ensemble des sous-buts SG et soit a_f l'unique action qui établit le fluent $f \in$ SG. Si pour toute $a \in A^r$ telle que $f \in \text{Del}(a)$:

- ∃ un fluent –monotone $p \in \text{Cond}(a) \cap \text{Del}(a_f)$ tel que $\tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f) \leq \tau(p \rightarrow\mid a) - \tau(a \rightarrow \neg f)$, ou
- ∃ un fluent –monotone $p \in \text{Add}(a) \cap \text{Del}(a_f)$ tel que $\tau(a_f \rightarrow \neg p) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow p) - \tau(a \rightarrow \neg f)$, ou
- ∃ un fluent +monotone $p \in \text{Del}(a) \cap \text{Add}(a_f)$ tel que $\tau(a_f \rightarrow p) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f)$, ou
- ∃ un fluent +monotone $p \in (\text{Del}(a) \cap \text{Cond}(a_f))\backslash I$ tel que $\tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f) \leq \tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f)$,

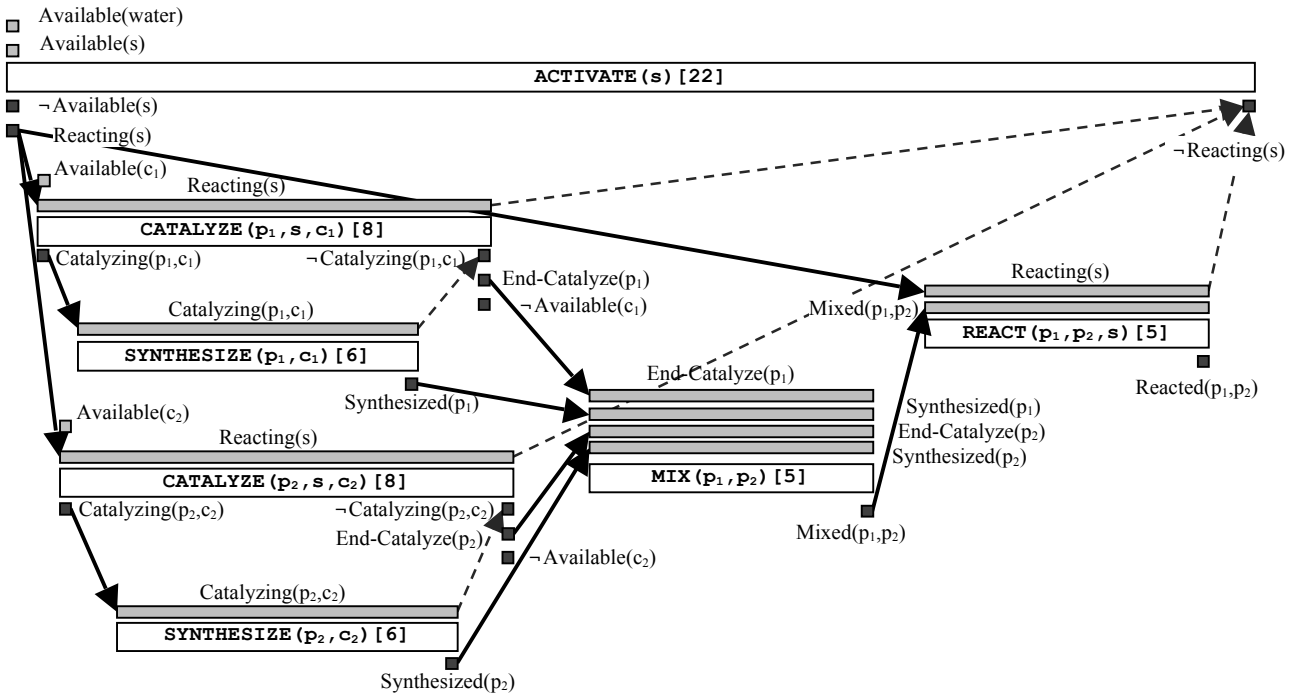
alors f est +monotone.

Proposition 2. Les règles 2 et 3 sont valides.

Preuve : Nous ne donnons que la preuve de validité de la Règle 2, puisque la preuve de la Règle 3 est entièrement similaire. Supposons que les prémisses de la Règle 2 soient vérifiées. Considérons un fluent arbitraire $f \in$ SG et soit a_f l'unique action qui établit le fluent $f \in$ SG. Supposons que $f \in \text{Del}(a)$ et qu'il existe un fluent –monotone $p \in \text{Del}(a) \cap \text{Cond}(a_f)$ tel que $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow\mid a_f) - \tau(a_f \rightarrow f)$. Mais, puisque p est –monotone, nous savons que $\tau(p \rightarrow\mid a_f) < \tau(a \rightarrow \neg p)$. Il résulte directement que $\tau(a_f \rightarrow f) < \tau(a \rightarrow \neg f)$. Par un argument similaire pour les trois autres cas listés dans la Règle 2, nous pouvons déduire que pour toutes les actions $a \in A^r$ telles que $f \in \text{Del}(a)$, $\tau(a_f \rightarrow f) < \tau(a \rightarrow \neg f)$. Puisque a_f est l'unique action qui établit f, nous pouvons déduire que f ne peut jamais être établi après qu'il soit détruit et donc qu'il est –monotone.

Le théorème suivant résulte maintenant du fait que les Règles 1, 2 et 3 peuvent clairement être appliquées jusqu'à la convergence en temps polynomial.

Théorème 4. Soit Π la classe des problèmes de planification temporelle positifs <I,A,G> dans lesquels A est establisher-unique relativement à Cond(A) ∪ G, tous les fluents dans Cond(A) ∪ G sont monotones et tous les fluents dans I ∩ (Cond(A) ∪ G) sont –monotones, où la monotonie des fluents peut être détectée en appliquant les Règles 1, 2 et 3 jusqu'à convergence. Alors Π est traitable.



5 Un Exemple de Planification de Processus Chimique

Le domaine « Temporal Chemical Process » comporte différents types d'opérations sur des produits chimiques qui sont réalisées dans la production industrielle de composés. Il comporte par exemple un opérateur qui peut *activer* une source de matière première. Cette matière première peut ensuite être *catalysée* de deux manières pour *synthétiser* deux produits différents. Ces produits peuvent être *mélangés* et *réagir* en utilisant la matière première une nouvelle fois pour produire le composé souhaité. Ce processus est illustré par le plan temporel donné dans la figure qui suit. Nous représentons les actions non-instantanées par un rectangle. La durée d'une action est donnée entre crochets après son nom Les conditions sont données au-dessus d'une action, et ses effets en-dessous. L'état initial et le but du problème de planification correspondant sont :

I={Available(water),Available(s),Available(c_1), Available(c_2)}

G={Reacted(p_1,p_2)}

Etant donné le problème de planification temporelle <I,A,G>, où A est l'ensemble de toutes les actions du domaine « Temporal Chemical Process », l'ensemble des sous-buts SG et l'ensemble réduit d'actions A^r sont :

SG={Reacted(p_1,p_2), Reacting(s), Mixed(p_1,p_2), Available(water), Available(s), End-Catalyze(p_1), End-Catalyze(p_2), Synthesized(p_1), Synthesized(p_2), Available(c_1), Available(c_2), Catalyzing(p_1,c_1), Catalyzing(p_2,c_2)}

A^r={REACT(p_1,p_2,s), ACTIVATE(s), MIX(p_1,p_2), CATALYZE(p_1,s,c_1), SYNTHESIZE(p_1,c_1), CATALYZE(p_2,s,c_2), SYNTHESIZE(p_2,c_2)}

Pour tout $i \neq j$ tels que $\{a_i, a_j\} \subset A^r$ nous avons $\text{Add}(a_i) \cap \text{Add}(a_j) \cap SG = \varnothing$. L'ensemble des actions A^r est donc establisher-unique relativement à SG. Nous pouvons remarquer immédiatement que le fluent Available(water) n'est jamais ajouté ou détruit, que les fluents Reacted(p_1,p_2), Mixed(p_1,p_2), End-Catalyze(p_1), Synthesized(p_1), End-Catalyze(p_2), Synthesized(p_2) sont uniquement ajoutés, et les fluents Available(s), Available(c_1), Available(c_2) sont uniquement détruits. Ainsi, aucun de ces fluents n'est dans $\text{Add}(A^r) \cap \text{Del}(A^r)$, et par le Règle 1, ils sont –monotones. En utilisant la Règle 2, nous pouvons prouver que Reacting(s) est –monotone. a_f = ACTIVATE(s) est l'unique action qui établit le fluent f = Reacting(s) $\in$ SG. a = ACTIVATE(s) est également l'unique action qui détruit f = Reacting(s) et pour le fluent –monotone p = Available(s) $\in \text{Del}(a) \cap \text{Cond}(a_f)$, nous avons $\tau(a \rightarrow \neg p) - \tau(a \rightarrow \neg f) \leq \tau(p \rightarrow | a_f) - \tau(a_f \rightarrow f)$. Donc, par la Règle 2, Reacting(s) est –monotone. Par un argument similaire, en utilisant encore la Règle 2, nous pouvons prouver que Catalyzing(p_1,c_1) et Catalyzing(p_2,c_2) sont –monotones.

L'ensemble d'actions A^r est establisher-unique relativement à SG, chaque fluent dans $\text{Cond}(A^r) \cup G$ est monotone et chaque fluent dans $I \cap (\text{Cond}(A^r) \cup G)$ est –monotone relativement à $<I,A^r,G>$. En outre :
(1) $G \subseteq (I \backslash \text{Del}(A^r)) \cup \text{Add}(A^r)$
(2) $\text{Cond}(A^r) \subseteq I \cup \text{Add}(A^r)$

(3) tous les fluents $g \in G \cap Del(A^r) \cap Add(A^r)$ sont +monotone relativement à $<I,A^r,G>$ (trivialement, puisque cet ensemble est vide)
(4) il n'y a pas de contraintes d'effets contradictoires ni de +autorisation. L'ensemble des contraintes restantes a une solution sur l'ensemble des actions A^r.

Ainsi, par le Théorème 1, le problème $<I,A,G>$ a un plan-solution, montré dans la figure (les contraintes de causalité sont représentées par les flèches en gras, et les contraintes de –autorisation par des flèches en pointillés), et par le Théorème 2, cette solution peut être trouvée en temps polynomial.

Beaucoup d'autres problèmes de planification temporelle issus de l'industrie chimique peuvent également être résolus en temps polynomial d'une manière similaire. Par exemple, l'acétylène est une matière première dérivée du carbure de calcium grace à l'eau. Ensuite, le chlorure de vinyle est un monomère produit à partir de l'acétylène et du chlorure d'hydrogène en utilisant du chlorure mercurique comme catalyseur. Le PVC est alors produit par polymérisation. D'autres exemples interviennent dans l'industrie pharmaceutique dans la production de médicaments (comme le paracétamol ou l'ibuprofène) et, en général, dans beaucoup de processus qui nécessitent la production et la combinaison de plusieurs molécules, sachant qu'il existe une façon unique de les obtenir (souvent imposée par des raisons industrielles, économiques ou écologiques).

6 Discussion

Les résultats obtenus ici peuvent également être appliqués à la planification non-temporelle puisque, par exemple, un problème de planification classique STRIPS peut être modélisé comme un problème de planification temporelle dans lequel toutes les actions sont instantanées. Il est intéressant de souligner que la classe traitable des problèmes de planification classique dans lesquels toutes les actions sont establisher-unique et ou tous les fluents sont détectables comme étant (à la fois + et –) monotones en appliquant seulement la Règle 1, est couverte par la classe traitable PA de [25].

Pour simplifier la présentation et pour demeurer dans le cadre de PDDL2.1, nous avons considéré que les contraintes inhérentes entre les instants correspondant à des événements à l'intérieur d'une même instance d'action sont toutes des contraintes d'intervalle. Nous pouvons cependant généraliser nos classes traitables pour autoriser des contraintes min-closed arbitraires puisque c'est la seule propriété requise pour les contraintes dans la preuve du Théorème 1. Un exemple d'une telle contrainte C(x,y) est une contrainte binaire d'intervalle avec des bornes variables : $y\text{-}x \in [f(x,y),g(x,y)]$, qui est min-closed à condition que f(x,y) soit une fonction monotone croissante de x et g(x,y) soit une fonction monotone décroissante de y. Un autre exemple de contrainte min-closed est la contrainte ternaire $(x+y)/2 \leq z$, qui peut par exemple être utilisée pour imposer qu'un effet ait lieu durant la dernière moitié de l'exécution d'une action.

Un aspect important de la planification temporelle qui est absent de la planification non-temporelle, est que certains problèmes de planification temporelle, appelés problèmes temporellement expressifs, nécessitent la concurrence des actions afin d'être résolus [8]. Un exemple typique de problème temporellement expressif est la cuisine : plusieurs ingrédients ou plats doivent être cuisinés simultanément afin d'être prêts au même moment. Dans les environnements industriels, la concurrence des actions est souvent utilisée pour conserver l'espace de stockage et les délais d'exécutions dans des limites données. [7] ont identifié une sous-classe des problèmes temporellement expressifs qu'ils ont appelés problèmes temporellement cycliques et qui nécessitent des ensembles d'actions cycliquement dépendantes afin d'être résolus. Un exemple simple de ce type de problème est le développement de deux parties d'un logiciel, écrites par deux sous-traitants différents, chacun devant connaître la spécification de l'autre programme afin de construire correctement leur interface. La classe traitable de problèmes de planification temporelle décrite dans le Théorème 4 contient à la fois des problèmes temporellement expressifs et des problèmes temporellement cycliques. Ceci résulte du fait que, comme illustré par l'exemple donné dans [7], il est possible de construire un exemple de problème temporellement cyclique qui soit establisher-unique et dans lequel aucun fluent n'est détruit par aucune action (et donc dans lequel, d'après le Lemme 2, tous les fluents sont à la fois + et – monotones). Le problème de planification du processus chimique donné dans la Section 5 est un autre exemple de problème qui est temporellement expressif puisque la concurrence des actions est requise dans toutes les solutions.

7 Conclusion

Nous avons présenté une classe de problèmes de planification temporelle qui peut être résolue en temps polynomial. Nous avons identifié un certain nombre d'applications possibles dans l'industrie chimique. De nouvelles recherches sont nécessaires pour trouver d'autres domaines d'applications possibles et, au niveau théorique, pour découvrir d'autres règles permettant de prouver la monotonie des fluents.

Références

[1] Bäckström C., Klein I. (1991) Parallel non-binary planning in polynomial time. Proceedings IJCAI'1991, pp. 268-273.

[2] Bäckström C., Nebel B. (1995) Complexity results for SAS+ planning). Computational Intelligence 11(4), pp. 625-655.

[3] Brafman R.I., Domshlak C. (2003) Structure and Complexity in Planning with Unary Operators. Journal of Artificial Intelligence Research 18, pp. 315-349.

[4] Brafman R.I., Domshlak C. (2006) Factored Planning: How, When, and When Not". Proceedings of the 21st National Conference on AI.

[5] Bylander T. (1994) The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 69(1-2), pp.165-204.

[6] Chen H., Giménez O. (2008) Causal Graphs and Structurally Restricted Planning. Proceedings of the 18th International Conference on Automated Planning and Scheduling, ICAPS'2008.

[7] Cooper M.C., Maris F., Régnier P. (2010) Solving temporally cyclic planning problems, International Symposium on Temporal Representation and Reasoning (TIME), p. 113-120.

[8] Cushing W., Kambhampati S., Mausam, Weld D.S. (2007) When is Temporal Planning Really Temporal? Proceedings of 20th International Joint Conference on Artificial Intelligence, IJCAI'2007, pp. 1852-1859.

[9] McDermott D. (1998) PDDL, The Planning Domain Definition Language. Technical Report, http://cs-www.cs.yale.edu/ homes/dvm/.

[10] Domshlak C., Dinitz Y. (2001) Multi-agent off-line coordination: Structure and complexity. Proceedings of 6th European Conference on Planning, ECP'2001.

[11] Erol K., Nau D.S., Subrahmanian V.S. (1995) Complexity, decidability and undecidability results for domain-independent planning. Artificial Intelligence, 76(1-2), pp.75-88.

[12] Fox M., Long D. (2003) PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, Journal of Artificial Intelligence Research 20, pp. 61-124.

[13] Gerevini A., Cristani M. (1997) On Finding a Solution in Temporal Constraint Satisfaction Problems. Proceedings of 15th International Joint Conference on Artificial Intelligence, IJCAI'1997, pp. 1460-1465.

[14] Ghallab M., Nau D.S., Traverso P. (2004) Automated Planning: Theory and Practice, Morgan Kaufmann.

[15] O. Giménez, A. Jonsson (2008) The complexity of planning problems with simple causal graphs. Journal of AI Research 31, pp. 319-351.

[16] Haslum P. (2007) Reducing Accidental Complexity in Planning Problems. Proceedings of IJCAI'07, pp. 1898-1903.

[17] Haslum P. (2008) A New Approach To Tractable Planning. Proceedings of ICAPS'2008.

[18] Helmert M. (2003) Complexity results for standard benchmark domains in planning. Artificial Intelligence 143 (2), pp. 219-262.

[19] Helmert M. (2006) New Complexity Results for Classical Planning Benchmarks. Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS'2006, pp. 52-61.

[20] Hoffmann J. (2005) Where Ignoring Delete Lists Works, Local Search Topology in Planning Benchmarks. Journal of Artificial Intelligence Research 24, pp. 685-758.

[21] Jeavons P., Cooper M.C. (1995) Tractable constraints on ordered domains, Artificial Intelligence 79, pp. 327-339.

[22] Jonsson A. (2007) The Role of Macros in Tractable Planning Over Causal Graphs. Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'2007, pp. 1936-1941.

[23] Jonsson P., Bäckström C. (1994) Tractable planning with state variables by exploiting structural restrictions. Proceedings of AAAI'1994, pp. 998-1003.

[24] Jonsson P., Bäckström C. (1995) Incremental Planning. In New Directions in AI Planning: 3rd European Workshop on Planning, EWSP'1995, pp. 79-90.

[25] Jonsson P., Bäckström C. (1998) State-variable planning under structural restrictions: Algorithms and complexity. Artificial Intelligence, 100(1-2), pp. 125- 176.

[26] Katz M., Domshlak C. (2008) New Islands of Tractability of Cost-Optimal Planning. Journal of Artificial Intelligence Research, 32, pp. 203-288.

[27] Knoblock C.A. (1994) Automatically Generating Abstractions for Planning. Artificial Intelligence, 68(2), pp. 243-302.

[28] Koubarakis M. (1992) Dense Time and Temporal Constraints with ≠. Proceedings of 3rd International Conference on Principles of Knowledge Representation and Reasoning, KR'1992, pp. 24-35.

[29] Rintanen J. (2007) Complexity of concurrent temporal planning. Proceedings of the 17th International Conference on Automated Planning and Scheduling, ICAPS, pp. 280-287.

[30] Slaney J., Thiébaux S. (2001) Blocks World revisited. Artificial Intelligence 125, pp. 119-153.

[31] Vidal V., Geffner H. (2005) Solving Simple Planning Problems with More Inference and No Search. Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP'05, p. 682-696.

[32] Williams B.C., Nayak P. (1997) A reactive planner for a model-based executive. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 1178-1185.