Transformation de problèmes de planification optimale

Martin C. Cooper — Marie de Roquemaurel — Pierre Régnier

IRIT - Université Paul Sabatier
180, route de Narbonne, 31062 Toulouse, cedex 4
{cooper,deroquemaurel,regnier}@irit.fr

RÉSUMÉ. *La planification de coût optimal, dont le but est de minimiser la somme des coûts des actions, est un challenge pour l'IA en raison de sa complexité. Un programme linéaire obtenu par une relaxation qui ignore les contraintes d'ordonnancement des actions peut être utilisé pour déterminer une borne inférieure au coût d'un plan-solution. Nous montrons que le dual de ce programme linéaire fournit une transformation du problème en un problème de planification de coût optimal équivalent dans lequel le coût de l'action qui atteint le but est exactement égal à cette borne inférieure. Cette transformation est une technique universelle dans le sens où elle peut être appliquée comme une technique de prétraitement et peut donc être combinée avec d'autres techniques qui ont été développées pour la planification optimale.*

ABSTRACT. *Cost-optimal planning, in which the aim is to minimize the sum of costs of actions, is a challenging problem due to its computational complexity. A linear program derived from a relaxation which ignores the constraints on the ordering of actions can be used to obtain a lower bound on the cost of a solution-plan. We show that the dual of this linear program provides a transformation of the problem into an equivalent optimal planning problem in which the cost of the goal-achieving action is exactly equal to this lower bound. This transformation is of universal utility since it can be applied as a preprocessing technique and can thus be combined with any of the diverse techniques which have been developed for optimal planning.*

MOTS-CLÉS : *planification, optimisation, programmation linéaire, théorème fondamental de dualité*

KEYWORDS: *planning, optimization, linear programming, fundamental duality theorem*

1. Introduction

En planification, la recherche d'une solution de longueur quelconque est, dans le cadre de la planification propositionnelle STRIPS, PSPACE-Complet (Bylander, 1994). Lorsque, en plus, on cherche à produire des plans d'actions de coût optimal en minimisant la somme des coûts des actions, la difficulté s'accroît encore puisqu'il est alors nécessaire d'explorer la totalité de l'espace du problème. Dans de nombreux problèmes combinatoires comme celui-ci, une phase de prétraitement peut alors permettre de réduire considérablement le temps de recherche. Les algorithmes de recherche intelligents peuvent également utiliser des techniques similaires à chaque nœud de l'arbre de recherche qu'ils développent. Dans ces deux cas, qu'elles soient employées en prétraitement ou durant la recherche, ces techniques doivent être applicables en temps polynomial pour être utiles.

La transformation en temps polynomial d'une instance d'un problème combinatoire en un autre problème équivalent mais plus facile à résoudre est une approche classique en intelligence artificielle. Par exemple, la notion de cohérence d'arc est omniprésente dans les techniques de satisfaction de contraintes. Ces dernières années, plusieurs généralisations de cohérence d'arc ont été développées pour les problèmes de satisfaction de contraintes pondérées (Cooper *et al.*, 2004; Cooper *et al.*, 2007; Cooper *et al.*, 2008). Ces notions sont regroupées sous le terme générique de cohérence d'arc souple. Nous montrons dans cet article comment un problème de planification de coût optimal peut également être transformé, en temps polynomial, en un autre problème équivalent mais plus facile à résoudre et dans lequel la borne inférieure du coût d'un plan-solution est explicitée. Ces techniques sont fortement inspirées par les techniques de cohérence d'arc souple correspondantes.

Les approches de (Bylander, 1997; Benton *et al.*, 2007; van den Briel *et al.*, 2007; van den Briel *et al.*, 2008) permettent, en relaxant le problème de planification pour ignorer les contraintes d'ordonnancement entre actions, d'obtenir un programme linéaire grâce auquel on peut déterminer une borne inférieure au coût d'un plan-solution. Ces travaux, connexes aux nôtres, sont présentés au fur et à mesure de l'article, lorsque cela est nécessaire. Ils sont proches de notre travail mais notre transformation d'un problème de planification de coût optimal en un problème équivalent avec une borne inférieure explicite du coût de la solution est entièrement originale.

Dans la section 2, nous commençons par poser plusieurs définitions indispensables pour la suite de l'article. La section 3 décrit alors des inégalités linéaires que l'on peut obtenir en tenant seulement compte du nombre d'occurrences des actions et des fluents. Nous définissons ensuite dans la section 4 une transformation d'un problème de planification en un problème équivalent mais plus facile à résoudre. Enfin, nous montrons dans la section 5 comment obtenir une transformation du problème en un problème équivalent dans lequel le coût de l'action qui atteint le but est exactement égal à une borne inférieure au coût d'un plan-solution optimal.

2. Définitions préliminaires

Définition 1 (Etat, Fluent positif / négatif, Satisfaction) *Un état E est un ensemble de fluents (littéraux propositionnels) tel que : $\nexists f$ tel que $(f \in E) \wedge (\neg f \in E)$. Un fluent est positif (négatif) s'il est une variable propositionnelle non négative (négative). Un ensemble de fluents positifs (négatifs) noté E^+ (E^-) est associé à un état E. Un ensemble de fluents T est satisfait dans un état E si $T \subseteq E$, i.e. $T^+ \subseteq E^+$ et $T^- \subseteq E^-$.*

Nous noterons $\overline{T} = \{\neg f : f \in T\}$ pour représenter l'ensemble de la négation de tous les fluents de T et $A - B = \{f \in A : f \notin B\}$ pour représenter la différence entre deux ensembles.

Définition 2 (Action) *Une action a est un triplet $\langle prec(a), \mathit{eff}(a), cout(a)\rangle$, où $prec(a)$ est l'ensemble des préconditions propositionnelles de l'action a, $\mathit{eff}(a)$ est l'ensemble des effets propositionnels de a, $cout(a)$ est un entier naturel non nul représentant le coût de l'application de l'action a.*

Définition 3 (Application d'une action) *L'application d'une action a à un état E (notée $E \uparrow a$) est possible ssi tous les fluents de prec(a) sont satisfaits dans E. Cette application produit un état E' tel que : $E' = (E - \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$.*

Définition 4 (Problème de planification) *Un problème de planification est un triplet $\Pi = \langle A, I, B\rangle$ où I est l'état initial, A l'ensemble des actions et B l'ensemble des fluents du but qui doivent être satisfaits.*

Définition 5 (Plan) *Un plan est une séquence d'actions, noté $P = \langle a_1, \ldots, a_n\rangle$.*

Définition 6 (Application d'un plan) *Etant donné un état E_0, nous notons E_i ($i = 1, \ldots, n$) le résultat de l'application de l'action a_i à l'état E_{i-1} (en supposant qu'elle est applicable). Un plan $P = \langle a_1, \ldots, a_n\rangle$ est applicable dans un état E_0 si $\forall i \in \{1, \ldots, n\}$, a_i est applicable dans E_{i-1}. Si P est applicable dans un état E_0, alors $E_0 \uparrow P$ dénote l'état final E_n.*

Définition 7 (Plan-solution) *Un plan $P = \langle a_1, \ldots, a_n\rangle$ est un plan-solution pour le problème $\Pi = \langle A, I, B\rangle$ si $a_1, \ldots, a_n \in A$, P est applicable à partir de l'état initial I et que le but B est satisfait dans l'état final $I \uparrow P$.*

La qualité d'un plan P est estimée par une fonction appelée métrique du plan. Dans cet article, nous considérons une métrique additive qui est la somme des coûts des actions appartenant au plan P. C'est une hypothèse essentielle pour appliquer les approches de programmation linéaire décrites dans les sections 3 et 5. Cette métrique est généralement satisfaite par un coût financier par exemple.

Définition 8 (Plan optimal / Problème de planification optimale) *Etant donné un problème de planification $\Pi = \langle A, I, B \rangle$, un plan optimal est un plan-solution $P = \langle a_1, \ldots, a_n \rangle$ qui minimise la métrique*

$$cout_{\Pi}(P) = \sum_{1=1}^{n} cout(a_i)$$

Nous appelons problème de planification optimale un problème de planification pour lequel on cherche un plan-solution qui minimise la métrique.

Nous n'imposons pas toujours des coûts non négatifs pour les actions. En d'autres termes, une action peut produire un profit. Cependant, lorsqu'on considère de telles actions, il devient possible de définir des problèmes de planification optimale qui ont un nombre infini de plans-solutions qui diminuent le coût (ou qui augmentent le profit) et ces problèmes n'ont donc pas de plan-solution optimal.

Dans les sections suivantes, nous introduisons plusieurs techniques pour analyser ou transformer les problèmes de planification optimale. Ces techniques nécessitent quelques définitions et notations supplémentaires que nous présentons maintenant.

Notation 1 *Soit eff_f l'ensemble des actions ayant f dans leurs effets :* $\mathit{eff}_f = \{a \in A : f \in \mathit{eff}(a)\}$.

Définition 9 (Fluent effet-strict) *Un fluent f est un effet-strict si pour toutes les actions a, on a $f \in \mathit{eff}(a) \Rightarrow \neg f \in prec(a)$.*

Définition 10 (Action retrait-strict / Action effet-strict) *Une action a est retrait-strict si pour tous les fluents positifs f, on a $\neg f \in \mathit{eff}(a)^- \Rightarrow f \in prec(a)$. Une action a est effet-strict si pour tous les fluents (positifs ou négatifs) f, on a $f \in \mathit{eff}(a) \Rightarrow \neg f \in prec(a)$.*

Définition 11 (Problème de planification retrait-strict / effet-strict) *Un problème de planification est retrait-strict (ou effet-strict) si toutes ses actions sont retraits-stricts (ou effet-strict).*

Nous notons également F_A^+ l'ensemble des fluents dans leur forme positive associés à l'ensemble des actions A, i.e. l'ensemble des littéraux non négatifs f tels que f ou $\neg f$ appartiennent à une précondition ou un effet d'au moins une action de A.

Définition 12 (Etat complet) *Un état E est complet pour l'ensemble des fluents positifs F si pour chaque fluent $f \in F$ soit $f \in E$ soit $\neg f \in E$. Dans le contexte d'un problème de planification $\Pi = \langle A, I, B \rangle$, nous disons qu'un état E est complet s'il est complet pour l'ensemble des fluents positifs F_A^+.*

3. Occurrences des actions et des fluents

Une approche classique dans l'analyse des problèmes de planification consiste à calculer une borne inférieure au coût d'un plan-solution en résolvant une version relaxée du problème dans lequel tous les retraits (fluents négatifs présents dans les effets) des actions sont ignorés (McDermott, 1999; Bonet *et al.*, 2001; Hoffmann *et al.*, 2001). Une bonne borne inférieure est un élément essentiel pour la recherche complète d'un plan optimal. Bien que le calcul d'un plan optimal relaxé soit NP-dur (Bylander, 1994), son utilisation a néanmoins permis d'obtenir des heuristiques performantes, par exemple, dans le planificateur FF (Hoffmann *et al.*, 2001). Un inconvénient de cette relaxation est que chaque fluent positif n'a besoin d'être établi qu'une seule fois et que chaque action n'a besoin d'être exécutée qu'une fois, puisque les retraits sont ignorés. Une meilleure borne inférieure peut être obtenue en comptant le nombre d'occurrences de chaque action ou fluent pour prendre en compte des occurrences multiples (van den Briel *et al.*, 2007). Cette section reprend brièvement des inégalités linéaires obtenues à partir du nombre d'occurrences des actions et des fluents. La résolution d'une relaxation linéaire d'un programme linéaire en nombres entiers peut fournir une méthode pour calculer une borne inférieure au coût d'un plan-solution optimal, méthode dont la complexité temporelle est d'ordre polynomial. Cette relaxation prend en compte les retraits mais ignore l'ordonnancement des actions.

Nous proposons un ensemble d'outils utilisables pour analyser ou transformer des problèmes de planification optimale. Ces outils s'inspirent des techniques d'analyse ou de transformation de CSP pondérés (Cooper *et al.*, 2004; Cooper *et al.*, 2007; Cooper *et al.*, 2008) qui sont elles-mêmes des généralisations des techniques classiques d'établissement de cohérence dans les CSP (Dechter, 2003). Comme nous le montrons dans la section 5, les solutions du dual de notre programme linéaire correspondent aux transformations du problème de planification optimale en un problème équivalent.

Considérons un plan-solution P d'un problème de planification Π. Dans un état E, nous considérerons que les fluents de E^+ sont vrais, les fluents de E^- sont faux, et que tous les autres fluents ont une valeur indéfinie. C'est une légère généralisation de l'hypothèse du monde clos du formalisme STRIPS dans laquelle un état correspond à l'affectation de valeurs de vérité à tous les fluents. Il est ainsi possible que la valeur d'un fluent soit indéfinie dans l'état initial. Sa valeur sera établie plus tard par une action, mais une fois qu'un fluent est affecté, sa valeur ne peut pas redevenir indéfinie. Dans cette section, nous supposons que Π est un problème de planification qui peut également comporter des fluents négatifs dans les préconditions ou dans les buts.

Définition 13 (Etablissement d'un fluent) *Un fluent f est établi par une action a de P si f passe de faux ou indéfini à vrai durant l'exécution de a.*

Soit e_f le nombre de fois que le fluent f est établi durant l'application de P. Notons que la définition 13 s'applique pour des fluents négatifs aussi bien que positifs. $e_{\neg f}$ est le nombre de fois que le fluent f passe de la valeur vrai ou indéfini à la valeur faux.

Soit b_f (i_f) le nombre de fois que le fluent f est vrai dans le but (l'état initial). b_f et i_f valent soit 0 soit 1.

Lemme 1 *Si f est un fluent, alors*

$$b_f + i_{\neg f} - 1 \leq e_f - e_{\neg f}$$

Si $\neg f$ est effet-strict et $f \notin I$, alors

$$b_f \leq e_f - e_{\neg f}$$

Preuve : pour qu'un fluent soit établi par une action a, il doit être faux (ou indéfini) quand l'action a est appliquée. Ceci implique que durant l'exécution d'un plan, les établissements de f et de $\neg f$ alternent. Par conséquent, pour tous les fluents f, $-1 \leq (e_f - e_{\neg f})$. Si $b_f = 1$ (i.e. f doit être vrai à la fin de l'application du plan) ou $i_{\neg f} = 1$ (i.e. f est faux avant l'application du plan), alors $0 \leq (e_f - e_{\neg f})$. De plus, si nous avons à la fois $b_f = 1$ et $i_{\neg f} = 1$, alors $1 \leq (e_f - e_{\neg f})$. On en déduit immédiatement que $b_f + i_{\neg f} - 1 \leq e_f - e_{\neg f}$.

Lorsque $\neg f$ est effet-strict, pour toutes les actions a, $\neg f \in \mathit{eff}(a) \Rightarrow f \in prec(a)$. Si $f \notin I$, alors nous avons l'inégalité plus forte $b_f \leq e_f - e_{\neg f}$ car, dans ce cas, f doit être établi avant $\neg f$. ∎

Corollaire 1 *Pour tous les fluents f,*

$$b_f + i_{\neg f} - 1 \leq e_f - e_{\neg f} \leq 1 - b_{\neg f} - i_f$$

Preuve : l'application du Lemme 1 à $\neg f$ produit l'inégalité $b_{\neg f} + i_f - 1 \leq e_{\neg f} - e_f$. En le réarrangeant et en le combinant avec l'application du Lemme 1 à f, nous obtenons $b_f + i_{\neg f} - 1 \leq e_f - e_{\neg f} \leq 1 - b_{\neg f} - i_f$.

∎

Soit o_a le nombre d'occurences de l'action a dans le plan P.

Lemme 2 *Si f est un fluent, alors*

$$e_f \leq \sum_{a \in \mathit{eff}_f} o_a$$

et de plus, si f est effet-strict, alors nous avons une égalité.

Preuve : l'inégalité provient du fait que chaque établissement d'un fluent f doit être effectué par une action a pour laquelle $f \in \mathit{eff}(a)$. Si f est effet-strict, alors toutes les actions de eff_f permettent réellement d'établir f, et nous avons alors l'égalité.

∎

Exemple 1 *Considérons le problème de planification* $\Pi = \langle \{A1, A2, A3, A4\}, I = \{c\}, B = \{a, b\}\rangle$ *:*

$A1 = \langle \{a\}, \{b, \neg a\}, 1\rangle$
$A2 = \langle \{c\}, \{a, \neg c\}, 1\rangle$
$A3 = \langle \{d\}, \{b, \neg d\}, 1\rangle$
$A4 = \langle \{b\}, \{c, d, \neg b\}, 1\rangle$

Pour trouver une borne inférieure au coût d'une solution optimale, nous cherchons une borne inférieure au nombre d'occurences des actions dans tous les plans-solutions. L'application du Lemme 2 aux fluents a*,* b *et* d *(qui ne sont pas effets-stricts) produit les inégalités :* $e_a \leq o_{A2}$*,* $e_b \leq o_{A1} + o_{A3}$ *et* $e_d \leq o_{A4}$*. Ces dernières nous permettent d'obtenir l'inégalité*

$$o_{A1} + o_{A2} + o_{A3} + o_{A4} \geq e_a + e_b + e_d$$

L'application du Lemme 1 aux fluents a*,* b *et* d *produit, après un réarrangement, les inégalités :* $e_a \geq 1 + e_{\neg a}$*,* $e_b \geq 1 + e_{\neg b}$ *et* $e_d \geq e_{\neg d}$ *qui nous permettent d'obtenir l'inégalité*

$$e_a + e_b + e_d \geq 2 + e_{\neg a} + e_{\neg b} + e_{\neg d}$$

Comme les fluents $\neg a$*,* $\neg b$ *et* $\neg d$ *sont effets-stricts, l'application du Lemme 2 produit les égalités :* $e_{\neg a} = o_{A1}$*,* $e_{\neg b} = o_{A4}$ *et* $e_{\neg d} = o_{A3}$ *qui nous permettent d'obtenir l'égalité*

$$2 + e_{\neg a} + e_{\neg b} + e_{\neg d} = 2 + o_{A1} + o_{A4} + o_{A3}$$

Comme $e_b \leq o_{A1} + o_{A3}$ *et* $e_b \geq 1 + e_{\neg b}$*, nous obtenons ensuite*

$$2 + o_{A1} + o_{A4} + o_{A3} \geq 3 + e_{\neg b} + o_{A4}$$

Enfin, comme $e_{\neg b} = o_{A4}$*, nous obtenons l'inégalité*

$$o_{A1} + o_{A2} + o_{A3} + o_{A4} \geq 3 + 2o_{A4}$$

Le coût de chaque action étant de 1, nous pouvons en déduire une borne inférieure de 3 au coût d'un plan-solution. Le coût du plan optimal $\langle A2, A1, A4, A3, A2\rangle$ *est de 5.*

De manière générale, nous cherchons une borne inférieure au coût d'un plan-solution optimal P. Ceci revient à résoudre le problème suivant :

$$\text{minimiser} \sum_{a \in A} o_a cout(a)$$

tel que les inégalités obtenues par les lemmes précédents soient satisfaites et que chaque variable o_a, e_f soit un entier qui ne soit pas négatif. Si nous résolvons la relaxation linéaire résultant du programme linéaire en nombres entiers, nous pouvons obtenir des valeurs non entières pour les variables o_a mais cette résolution fournira néanmoins une borne inférieure utile de $cout(P)$. Si le nombre d'actions et de fluents n'est pas excessif, nous pouvons également envisager une recherche exhaustive pour résoudre le programme linéaire en nombres entiers.

4. Transformation d'un problème de planification

La transformation d'un problème de planification en un problème de planification plus facile à résoudre doit permettre de conserver les mêmes solutions avec les mêmes coûts. Nous posons donc la définition suivante :

Définition 14 (Transformation préservant le coût, TPC) *Une transformation qui préserve le coût (TPC, Transformation Préservant les Coûts) d'un problème* Π *de planification optimale est une opération qui transforme* Π *en un problème* Π' *de planification optimale tel que* Π *et* Π' *ont le même ensemble de plans-solutions et tel que pour chaque plan-solution* P, $cout_{\Pi}(P) = cout_{\Pi'}(P)$.

Nous dirons que Π *et* Π' *sont équivalents en coût.*

Il est important de noter qu'une telle transformation ne préserve pas, en général, le coût des actions ou des plans partiels. La définition 14 implique en effet que tous les problèmes qui n'ont aucun plan-solution sont équivalents en coût.

Nous utilisons ici principalement les TPC pour transformer un problème Π en un autre problème Π' dans lequel la borne inférieure du coût d'un plan-solution optimal est plus explicite. Mais ces transformations peuvent être utilisées pour d'autres applications : égaliser le coût des actions afin de trouver une meilleure borne supérieure au nombre d'actions dans un plan optimal ou, au contraire, augmenter le coût d'une action particulière afin de montrer qu'elle est trop onéreuse pour faire partie d'un plan optimal. Nous illustrons ces idées par un exemple. Soit un problème de transport Π dans lequel, pour aller de A à B, nous avons deux choix :

1) prendre un taxi en A jusqu'à l'aéroport le plus proche (coût 20 euros) ; prendre un avion jusqu'à l'aéroport le plus proche de B (coût 140 euros) ; prendre un taxi de cet aéroport jusqu'à B (coût 20 euros).

2) prendre un bus en A jusqu'à la gare la plus proche (coût 2 euros) ; prendre un train jusqu'à la gare la plus proche de B (coût 152 euros) ; prendre un bus de cette gare jusqu'à B (coût 2 euros).

Ce problème est clairement équivalent en coût à un problème identique Π' dans lequel le coût d'un trajet en taxi est de 60 euros, un vol en avion est de 60 euros, un trajet en bus est de 52 euros et un trajet en train coûte 52 euros. Dans cette version, le coût minimum d'une action est de 52 euros alors qu'il est de 2 euros dans le problème original. Une fois que le plan-solution (bus-train-bus) de coût total 156 euros a été trouvé, nous pouvons déduire qu'il n'existe pas de meilleur plan comprenant plus de deux actions, car dans Π' chaque action a un coût d'au moins 52 euros. Dans une autre version Π'' équivalente en coût au même problème, les taxis sont gratuits mais un trajet en avion coûte 180 euros, nous pouvons en déduire immédiatement que cette action ne peut pas faire partie d'un plan optimal, car nous avons déjà un plan coûtant seulement 156 euros. Π est également équivalent en coût au problème Π_{opt} qui est identique à Π excepté que le coût d'un taxi à partir de A vers l'aéroport est de 24

euros, toutes les autres actions ont un coût de 0, mais il y a un coût de 156 euros à payer en arrivant à B (quel que soit le moyen de transport utilisé).

Par rapport aux approches similaires, la nouveauté de l'approche que nous présentons est qu'elle permet de calculer une borne inférieure au coût (ici 156 euros) en transformant le problème originel en un problème équivalent dans lequel cette borne est rendue explicite comme étant le coût d'une unique action de réalisation du but. L'avantage de notre méthode est que les coûts présents dans le problème transformé peuvent, en plus, être utilisés pour guider la recherche (par exemple en choisissant prioritairement les actions de coût nul ou en éliminant de la recherche les actions dont le coût, cumulé à celui de la borne inférieure, dépasse le coût du meilleur plan-solution déjà trouvé).

L'opération de tranformation locale que nous définirons par la suite nécessite que le problème de planification soit effet-strict pour garantir la préservation des coûts lors de son application. Avant de présenter cette opération, nous montrons, par deux lemmes qui décrivent deux méthodes distinctes, comment la plupart des problèmes de planification peuvent être transformés en des problèmes optimaux équivalents et effets-stricts.

Définition 15 (Arité d'une action, Problème de planification k-local) *L'arité d'une action a est $|\mathit{eff}(a)|$, le nombre de fluents positifs ou négatifs appartenant à $\mathit{eff}(a)$. Un problème de planification optimale $\Pi = \langle A, I, B\rangle$ est k-local si l'arité de chaque action $a \in A$ est bornée par k.*

Lemme 3 *Il existe une réduction polynomiale de la classe des problèmes de planification optimale k-locaux ayant un état initial complet, où k est une constante, vers la classe des problèmes de planification optimale effets-stricts.*

Preuve : Soit $\Pi = \langle A, I, B\rangle$ un problème de planification optimale dans lequel I est un état complet et tel que, $\forall a \in A, |\mathit{eff}(a)| \leq k$. Pour chaque fluent $f \in F_A$ et une action $a \in A$ tels que $\neg f \in \mathit{eff}(a)$ et $f \notin \mathit{prec}(a)$, nous pouvons remplacer a par deux nouvelles actions a_1 et a_2 telles que

$$\begin{array}{lll} \mathit{prec}(a_1) = \mathit{prec}(a) \cup \{f\} & \mathit{eff}(a_1) = \mathit{eff}(a) & \mathit{cout}(a_1) = \mathit{cout}(a) \\ \mathit{prec}(a_2) = \mathit{prec}(a) \cup \{\neg f\} & \mathit{eff}(a_2) = \mathit{eff}(a) - \{\neg f\} & \mathit{cout}(a_2) = \mathit{cout}(a) \end{array}$$

L'action a_1 (a_2) est une version de a qui peut être appliquée dans un état E si f est vrai (faux) dans E ; a_1 supprime f tandis que a_2 laisse la valeur de f inchangée. Nous devons prouver que le problème résultant $\Pi_{(f,a)}$ est équivalent à Π. Soit P un plan-solution pour Π qui contient l'action a. Soit E un état qui est atteint durant l'application de P juste avant l'application de l'action a. Comme l'état I est complet, l'état E l'est aussi. Nous pouvons alors simplement remplacer a par a_1 ou a_2 selon que $f \in E$ ou que $\neg f \in E$. Ainsi, un plan-solution pour Π peut être converti en temps polynomial en un plan-solution pour $\Pi_{(f,a)}$. Inversement, si P' est un plan-solution

pour $\Pi_{(f,a)}$, alors il est trivial d'obtenir un plan-solution P pour Π, en remplaçant toutes les actions a_1 ou a_2 appartenant à P' par a.

Nous pouvons clairement appliquer la transformation ci-dessus pour chaque fluent f (positif ou négatif) et pour chaque action a tels que $\neg f \in \mathit{eff}(a)$ et $f \notin prec(a)$. Le problème de planification Π' qui en résulte est effet-strict. La transformation de Π à Π' introduit 2^{k_a} copies de l'action a, où k_a est le nombre de fluents f pour lesquels $\neg f \in \mathit{eff}(a)$ et $f \notin prec(a)$. Comme $k_a \leq k$ et k est une constante, la transformation est une réduction polynomiale. ∎

Le Lemme 3 fournit seulement une réduction polynomiale de la classe des problèmes de planification k-locaux à la classe des problèmes de planification effets-stricts. Le lemme suivant nous montre que si le problème original est retrait-strict et positif, nous n'avons pas besoin de borner l'arité des actions.

Définition 16 (Problème de planification positif) *Un problème de planification est positif si tous les fluents du but et toutes les préconditions de toutes les actions sont positifs.*

Notons que, dans un problème de planification positif, les actions peuvent avoir des effets négatifs. Il est bien connu qu'un problème de planification peut être transformé en un problème de planification positif de taille comparable en introduisant un nouveau fluent $notf$ pour chaque fluent positif f (Ghallab *et al.*, 2004).

Lemme 4 *Il existe une réduction polynomiale de la classe des problèmes de planification optimale positifs et retraits-stricts vers la classe des problèmes de planification optimale effets-stricts.*

Preuve : soit $\Pi = \langle A, I, B\rangle$ un problème de planification optimale positif et retrait-strict. Par hypothèse, toutes les actions $a \in A$ sont retraits-stricts. Pour chaque action $a \in A$, nous définissons l'action a' ainsi :

$$\begin{aligned} prec(a') &= prec(a) \cup \{\neg f | f \in \mathit{eff}^+(a)\} \\ \mathit{eff}(a') &= \mathit{eff}(a) \\ cout(a') &= cout(a) \end{aligned}$$

Nous définissons également, pour chaque fluent positif $f \in F_A^+$, une nouvelle action factice d_f qui supprime simplement le fluent f, i.e.

$$\begin{aligned} prec(d_f) &= \{f\} \\ \mathit{eff}(d_f) &= \{\neg f\} \\ cout(d_f) &= 0 \end{aligned}$$

Toutes les actions a' (pour $a \in A$) et d_f (pour $f \in F_A^+$) sont effets-stricts. Soit $\Pi' = \langle A', I, B\rangle$, où

$$A' = \{a' | a \in A\} \cup \{d_f | f \in F_A^+\}$$

Π' est effet-strict. Il nous reste à montrer qu'il est équivalent à Π. Un plan-solution P pour Π peut être transformé en un plan-solution pour Π'

1) en remplaçant chaque action a appartenant à P par l'action a',

2) et en ajoutant une action factice d_f juste avant l'action a' quand $\neg f$ est une précondition de a' et qu'elle n'est pas satisfaite.

Inversement, un plan-solution P' pour Π' peut être transformé en un plan-solution pour Π par la suppression des actions factices d_f de P', et en remplaçant chaque action a' dans P' par l'action correspondante a. Comme le but et les préconditions des actions de Π ne contiennent aucun fluent négatif, le plan résultant est valide. Dans les deux sens, les plans-solutions ont le même coût puisque les actions factices ont un coût nul. De plus, la réduction est clairement polynomiale. ■

Nous avons étudié tous les problèmes des 26 domaines non temporels STRIPS proposés par les compétitions de IPC'2000[1] (Bacchus, 2001), IPC'2002[2] (Long *et al.*, 2003), IPC'2004[3] (Hoffmann *et al.*, 2006) et IPC'2006[4] (Dimopoulos *et al.*, 2006) et IPC'2008[5]. Les 26 domaines étudiés sont : *Schedule*, *Logistics*, *Freecell*, *Elevator*, *Blocks*, *Depots*, *Driver*, *Zeno*, *Rovers*, *Satellite*, *Airport*, *Pipesworld*, *Promela optical telegraph*, *Promela philosophers*, *Psr*, *Pathways*, *Storage*, *Tpp*, *Trucks*, *Openstacks*, *ParcPrinter*, *Pegsol*, *Scanalyser*, *Sokoban*, *Transport* et *Woodworking*.

– Pour 19 de ces domaines (*Schedule*, *Logistics*, *Freecell*, *Elevator*, *Blocks*, *Depots*, *Driver*, *Zeno*, *Rovers*, *Psr*, *Pathways*, *Tpp*, *Trucks*, *ParcPrinter*, *Openstacks*, *Pegsol*, *Scanalyser*, *Sokoban*, *Transport*), toutes les instances sont à la fois positives et retraits-stricts, le Lemme 4 peut donc être appliqué. Nous avons calculé (*cf.* tableau 1) les moyennes des pourcentages des augmentations du nombre d'actions obtenues en appliquant la transformation du Lemme 4 à des problèmes positifs et retraits-stricts. Nous avons seulement pris en compte les actions et fluents accessibles à partir de l'état initial (en utilisant l'analyse du graphe de planification relaxé avec le planificateur hsp_0^* (Haslum, 2008)). Le pourcentage moyen de l'augmentation du nombre d'actions est de 34,10 %. Cette valeur tend à diminuer avec la taille du problème considéré et tombe à 16,56 % si l'on considère uniquement les plus gros problèmes de chaque domaine. Ces résultats montrent que la plupart des problèmes positifs et retraits-stricts peuvent être transformés en problèmes équivalents et effets-stricts avec seulement une petite augmentation de la taille du problème.

– Dans 4 des 7 domaines restants, la valeur maximum de k_a (pour $a \in A$) est relativement petite : 2 pour *Satellite* ; 3 pour *Storage* ; 6 pour *Pipesworld* ; 7 pour

1. http ://www.cs.toronto.edu/aips2000/
2. http ://planning.cis.strath.ac.uk/competition/
3. http ://ls5-web.cs.uni-dortmund.de/ edelkamp/ipc-4/
4. http ://zeus.ing.unibs.it/ipc-5/
5. http ://ipc.informatik.uni-freiburg.de/

	% des augmentations du nombre d'actions	
domaine	en moyenne	plus gros problème
depots	22,14	7,04
driverlog	22,42	8,26
logistics	26,96	13,57
zenotravel	9,67	2,60
rovers	40,23	12,12
freecell	5,28	1,78
blocks	65,00	55,13
psr	68,53	6,29
elevator	17,99	5,00
openstack	20,07	5,90
parcPrinter	47,49	32,89
pegsol	57,44	54,05
scanalyzer	4,10	0,69
sokoban	62,71	39,15
transport	6,92	3,91
pathways	43,82	27,78
tpp	75,26	11,51
trucks	17,74	10,38

Tableau 1. *Moyenne des pourcentages des augmentations du nombre d'actions obtenues en appliquant la transformation du Lemme 4 à des problèmes positifs et retraits-stricts. La moyenne est calculée sur les 20 problèmes (ou plus) de chaque domaine. Pour chaque domaine, la colonne "plus gros problème" donne, en pourcentage, l'augmentation du nombre d'actions pour le problème qui en comporte le plus grand nombre*

Woodworking. Nous pouvons donc envisager, pour ces 4 domaines, d'appliquer la transformation décrite dans la preuve du Lemme 3.

– La valeur maximale de k_a est atteinte dans les domaines *Airport*, *Promela optical telegraph* et *Promela philosophers* avec des valeurs comprises entre 12 (pour *Airport*) et 178 (pour le problème $pfile7$ de *Promela optical telegraph*).

Nous pouvons en conclure que, dans la majorité des benchmarks, nous pouvons transformer le problème en un problème équivalent et effet-strict sans augmenter de manière excessive la taille de l'ensemble des actions.

L'opération **Arc-Mouvement** (définie dans la section 5) ne garantit la préservation des coûts que si elle est appliquée aux fluents qui ont une valeur identique avant et après l'application de tous les plans-solutions. Comme nous le montrons ci-dessous, il est toujours possible de normaliser un problème de planification pour que cette condition soit satisfaite.

Rappelons que F_A^+ représente l'ensemble des fluents dans leur forme positive associés à l'ensemble des actions A, i.e. l'ensemble des littéraux non négatifs f tel que f ou $\neg f$ appartiennent à une précondition ou un effet d'au moins une action de A.

Définition 17 (Problème de planification normalisé) *Un problème de planification optimale $\Pi = \langle A, I, B\rangle$ est normalisé si l'état initial est $\{init\} \cup \{\neg f | f \in F_A^+ - \{init\}\}$ et l'état but est $\{but\} \cup \{\neg f | f \in F_A^+ - \{but\}\}$.*

Lemme 5 *Il existe une réduction polynomiale de la classe des problèmes de planification optimale effets-stricts avec un état initial complet vers la classe des problèmes de planification optimale effets-stricts normalisés.*

Preuve : soit $\Pi = \langle A, I, B\rangle$ un problème de planification optimale effet-strict, avec I un état complet. Nous introduisons deux nouveaux fluents $init$ et but pour lesquels on peut considérer, sans perte de généralité, qu'ils ne sont pas présents dans F_A^+. Soit $F = F_A^+ \cup \{init, but\}$. Nous définissons les actions action-initialisation a_I et action-but a_B ainsi :

$$\begin{array}{l} prec(a_I) = \{init\} \cup \{\neg f | f \in F - \{init\}\} \\ \mathit{eff}(a_I) \ = I^+ \cup \{\neg init\} \\ cout(a_I) = 0 \end{array}$$

$$\begin{array}{l} prec(a_B) = B \cup \{\neg f | f \in F - B^+\} \\ \mathit{eff}(a_B) \ = \{\neg f | f \in B^+\} \cup \{but\} \\ cout(a_B) = 0 \end{array}$$

Le résultat de l'application de l'action a_I est un état complet $I \cup \{\neg init, \neg but\}$. La précondition de l'action a_B est également un état complet dans lequel tous les fluents n'appartenant pas à B prennent la valeur faux. Afin de pouvoir affecter les fluents de F_A^+ qui n'appartiennent pas au but à faux, nous définissons, pour chaque fluent $f \in F_A^+ - B^+$, une action factice d_f^B telle que :

$$\begin{array}{rcl} prec(d_f^B) & = & B \cup \{f\} \\ \mathit{eff}(d_f^B) & = & \{\neg f\} \\ cout(d_f^B) & = & 0 \end{array}$$

Les actions a_I, a_B et d_f^B (pour $f \in F_A^+ - B^+$) sont toutes effets-stricts. L'action d_f^B a simplement pour effet de retirer f : elle joue le même rôle que l'action factice d_f dans la preuve du Lemme 4 (excepté que d_f^B ne peut être appliquée qu'après la satisfaction de B).

Soit $\Pi' = \langle A', I', B'\rangle$, tel que

$$\begin{array}{l} A' = A \cup \{a_I, a_B\} \cup \{d_f^B | f \in F_A^+ - B^+\} \\ I' \ = \{init\} \cup \{\neg f | f \in F - \{init\}\} \\ B' = \{but\} \cup \{\neg f | f \in F - \{but\}\} \end{array}$$

Π' est effet-strict et normalisé. Il reste à prouver qu'il est équivalent à Π. Un plan-solution P de Π peut être transformé en un plan-solution de Π' en préfixant P par une action-initialisation a_I, en terminant P par une action-but a_B et en insérant une action d_f^B juste avant a_B pour tous les fluents positifs $f \in F_A^+ - B^+$ qui sont vrais à la fin de l'application de P. Inversement, un plan-solution P' pour Π' peut être transformé en un plan-solution pour Π en supprimant dans P' toutes les actions a_I, a_B et d_f^B (pour $f \in F_A^+ - B^+$). Dans les deux sens, les plans-solutions ont les mêmes coûts car l'action-initialisation, l'action-but et les actions factices ont un coût nul. Cette réduction est clairement polynomiale. ∎

Dans la suite de cet article, nous considérons des problèmes de planification optimale effets-stricts et normalisés. Nous supposons que l'une des transformations décrites dans les Lemmes 3 ou 4 a été appliquée, suivie par la transformation décrite dans le Lemme 5. Notons que si le problème original est positif et retrait-strict, il n'est pas nécessaire d'introduire les actions factices d_f^B car les actions factices d_f (décrites dans la preuve du Lemme 4) remplissent déjà ce rôle.

Considérons la transformation **Arc-Mouvement**(f, δ), décrite dans l'algorithme 1, qui permet de déplacer un coût δ entre les actions liées à un fluent f. Sans perte de généralité, nous supposons que f est un fluent positif et $\delta \in \mathbb{R}$. Comme le montre le lemme suivant, cette transformation préserve les coûts dans un problème de planification optimale Π effet-strict et normalisé.

Algorithme 1 Opération de transformation préservant les coûts des problèmes de planification optimale effets-stricts et normalisés

Arc-Mouvement(f, δ) :
  pour toutes les actions $a \in \mathit{eff}_f$ faire
    $cout(a) \leftarrow cout(a) - \delta$;
  pour toutes les actions $a \in \mathit{eff}_{\neg f}$ faire
    $cout(a) \leftarrow cout(a) + \delta$;

Lemme 6 *Soit Π un problème de planification optimale normalisé dans lequel f est un fluent positif tel que $f \notin \{init, but\}$. Si f et $\neg f$ sont des fluents effets-stricts, alors **Arc-Mouvement**(f, δ) est une transformation qui préserve les coûts.*

Preuve : soit P un plan-solution pour Π. Soit e_f ($e_{\neg f}$) le nombre de fois que f (respectivement $\neg f$) est établi durant l'application de P et o_a le nombre de fois que l'action a apparaît dans P. Comme Π est normalisé et que f est un fluent positif tel que $f \notin \{init, but\}$, $\neg f$ apparaît à la fois dans l'état initial et dans l'état but. En appliquant le Corollaire 1, nous obtenons

$$0 \leq e_f - e_{\neg f} \leq 0$$

et donc $e_f = e_{\neg f}$. Si f et $\neg f$ sont effets-stricts, par le Lemme 2, nous avons

$$e_f \;=\; \sum_{a \in \mathit{eff}_f} o_a, \qquad e_{\neg f} \;=\; \sum_{a \in \mathit{eff}_{\neg f}} o_a$$

Par conséquent, le coût de P dans le problème transformé Π' est

$$\mathit{cout}_{\Pi'}(P) \;=\; \mathit{cout}_{\Pi}(P) \;-\; e_f\delta \;+\; e_{\neg f}\delta \;=\; \mathit{cout}_{\Pi}(P)$$

■

5. Transformation optimale d'un problème de planification

L'opération **Arc-Mouvement**(f, δ) permet de transformer un problème de planification effet-strict et normalisé en un autre problème équivalent en coût. Mais cette opération est juste une opération locale et il peut être beaucoup plus productif d'appliquer simultanément un ensemble d'opérations de transformations par arc-mouvement. En effet, nous avons vu qu'un des objectifs communs aux problèmes combinatoires de minimisation consiste à trouver une borne inférieure à la valeur d'une solution optimale. Cette borne inférieure est également essentielle dans des algorithmes de recherche intelligents qui élaguent l'arbre de recherche. Nous cherchons donc à maximiser cette borne inférieure. Si a_B est l'action-but, alors nous devons maximiser $\mathit{cout}(a_B)$. Bien sûr, ceci est une borne inférieure au coût du plan-solution optimal seulement si tous les autres coûts ne sont pas négatifs.

Un ensemble d'arc-mouvements est clairement équivalent à un ensemble d'appels de l'opération **Arc-Mouvement**(f, δ_f) où f varie entre tous les fluents de $F_A^+ - \{\mathit{init}, \mathit{but}\}$. Dans cette section, et pour simplifier la notation, nous supposerons que toutes les sommes où les quantifications de fluents portent sur tous les fluents positifs de $F_A^+ - \{\mathit{init}, \mathit{but}\}$.

Pour des coûts réels finis, les appels à **Arc-Mouvement** commutent. Notons que les coûts originaux des actions peuvent être négatifs (représentant un bénéfice gagné par l'application de l'action), à condition que dans la version transformée Π', le coût de chaque action a ne soit pas négatif :

$$\mathit{cout}(a) \;-\; \sum_{f \in \mathit{eff}(a)} \delta_f \;+\; \sum_{\neg f \in \mathit{eff}(a)} \delta_f \;\geq\; 0$$

Par conséquent, le problème qui consiste à trouver la meilleure borne inférieure peut être exprimé par le programme linéaire suivant :

LP1 : maximiser $\sum_{f \in B} \delta_f$ tel que
pour toutes les actions $a \in A - \{a_B\}$,
$$\textstyle\sum_{f \in \mathit{eff}(a)} \delta_f \;-\; \sum_{\neg f \in \mathit{eff}(a)} \delta_f \;\leq\; \mathit{cout}(a)$$

Nous n'imposons pas la contrainte que $cout(a_B)$ ne soit pas négatif dans Π', car c'est exactement cette valeur que nous essayons de maximiser. En effet, si l'on autorise les profits, la valeur maximum de $\sum_{f \in B} \delta_f$ peut être négative.

Définition 18 (Transformation arc-optimale) *Un problème de planification optimale effet-strict et normalisé avec une action-but a_B est transformé de manière arc-optimale si aucun ensemble d'opérations d'arc-mouvement ne peut augmenter $cout(a_B)$ tout en conservant des coûts non négatifs pour toutes les autres actions.*

Théorème 1 *Si Π est un problème de planification optimale effet-strict et normalisé, alors un problème Π', transformé de manière arc-optimale et équivalent en côut à Π, peut être trouvé en un temps polynomial.*

Preuve : ceci découle immédiatement de la discussion ci-dessus et du fait que la programmation linéaire peut être résolue en un temps polynomial (Karmarkar, 1984; Schrijver, 1998). ∎

Exemple 2 *Considérons un problème de planification contenant les dix actions suivantes :*

$$A1 = \langle \{a, \neg b\}, \{b, \neg a\}, 1 \rangle$$
$$A2 = \langle \{c, \neg a\}, \{a, \neg c\}, 1 \rangle$$
$$A3 = \langle \{d, \neg b\}, \{b, \neg d\}, 1 \rangle$$
$$A4 = \langle \{b, \neg c, \neg d\}, \{c, d, \neg b\}, 1 \rangle$$
$$a_I = \langle \{\neg a, \neg b, \neg c, \neg d, \neg but, init\}, \{c, \neg init\}, 0 \rangle$$
$$a_B = \langle \{a, b, \neg c, \neg d, \neg init, \neg but\}, \{\neg a, \neg b, but\}, 0 \rangle$$
$$d_f = \langle \{f\}, \{\neg f\}, 0 \rangle \text{ pour chaque } f \in \{a, b, c, d\}$$

Ce problème Π est équivalent au problème de l'exemple 1. Il a été transformé selon les preuves des lemmes 4 et 5. Il est effet-strict et normalisé.

En appliquant l'approche précédente pour trouver la transformation de Π qui maximise la somme des coûts sur les fluents de but, nous obtenons le programme linéaire suivant :

$$\begin{aligned}
&\textit{maximiser } \delta_a + \delta_b \\
&\textit{tel que } \delta_b - \delta_a \leq 1 \\
&\qquad \delta_a - \delta_c \leq 1 \\
&\qquad \delta_b - \delta_d \leq 1 \\
&\qquad \delta_c + \delta_d - \delta_b \leq 1 \\
&\qquad \delta_c \leq 0 \\
&\qquad -\delta_f \leq 0 \text{ pour chaque } f \in \{a, b, c, d\}
\end{aligned}$$

Ce programme linéaire a une solution optimale $\delta_a = 1$, $\delta_b = 2$, $\delta_c = 0$, $\delta_d = 1$. Nous pouvons en déduire que Π est équivalent en coût à un problème identique dans lequel les coûts des actions sont :

$$\begin{array}{lll} cout(\mathrm{A1}) = 0 & cout(a_I) = 0 & cout(d_a^B) = 1 \\ cout(\mathrm{A2}) = 0 & cout(a_B) = 3 & cout(d_b^B) = 2 \\ cout(\mathrm{A3}) = 0 & & cout(d_c^B) = 0 \\ cout(\mathrm{A4}) = 2 & & cout(d_d^B) = 1 \end{array}$$

Ainsi la borne inférieure au coût de tous les plans-solutions optimaux a une valeur de 3, car le coût de l'action action-but a_B a un coût de 3.

Considérons un problème de planification Π effet-strict et normalisé qui contient une action-but a_B comme donnée dans la preuve du Lemme 5. Par les arguments donnés dans la preuve du Lemme 6, pour chaque fluent positif $f \notin \{init, but\}$,

$$\sum_{a \in \mathit{eff}_f} o_a \; = \; \sum_{a \in \mathit{eff}_{\neg f}} o_a$$

que nous pouvons réécrire :

$$\sum_{a \in \mathit{eff}_f - \{a_B\}} o_a - \sum_{a \in \mathit{eff}_{\neg f} - \{a_B\}} o_a = b_f$$

où $b_f = 1$ si $f \in B$ (et 0 sinon), comme le fluent positif f appartient à B ssi $\neg f \in \mathit{eff}(a_B)$, il n'y a aucun fluent positif $f \notin \{init, but\}$ dans $\mathit{eff}(a_B)$ et nous avons clairement dans le plan-solution $o_{a_B} = 1$. Nous pouvons alors obtenir une borne inférieure au coût de la solution optimale à partir des techniques de la section 3 et en résolvant le programme linéaire suivant :

LP2 : minimiser $\sum_{a \in A - \{a_B\}} o_a cout(a)$
tel que $\forall f \in F_A^+ - \{init, but\}$,
$\sum_{a \in \mathit{eff}_f - \{a_B\}} o_a \; - \; \sum_{a \in \mathit{eff}_{\neg f} - \{a_B\}} o_a = b_f$
et $\forall a, \; o_a \geq 0$

Théorème 2 *Les bornes inférieures obtenues en résolvant les programmes linéaires **LP1** et **LP2** sont identiques à condition qu'au moins un des problèmes soit faisable.*

Preuve : LP2 est le dual de **LP1**. Le résultat est une conséquence directe du théorème fondamental de dualité (Trustrum, 1971). ∎

Les expérimentations réalisées dans (van den Briel *et al.*, 2007) utilisent **LP2** pour obtenir une borne inférieure au coût des plans optimaux dans des problèmes issus de

Domaine	nombre de problèmes	borne inférieure / Optimal
logistics	7	0,73
freecell	5	1,00
driverlog	6	0,66
zenotravel	6	0,71
tpp	5	0,76
blocksworld	4	0,72

Tableau 2. *Moyenne des bornes inférieures (exprimée par rapport au coût de la solution optimale) obtenues en utilisant* **LP2** *sur différents problèmes*

six domaines énumérés dans le tableau 2. Dans ces expérimentations, toutes les actions ont un coût de 1. La colonne "nombre de problèmes" contient le nombre de problèmes pour lesquels ils ont réussi à trouver le coût d'un plan optimal (et pour lesquels le coût optimal a été retourné par Satplanner (Rintanen *et al.*, 2005)). Pour les problèmes dont le coût d'un plan optimal est fourni par Satplanner, nous avons calculé le rapport entre la valeur des bornes inférieures obtenues par **LP2** et le coût d'une solution optimale. La colonne "Borne inférieure/Optimal" présente la moyenne des rapports obtenus pour chaque domaine. Dans chaque cas, la borne inférieure correspond à au moins 66 % du coût d'une solution optimale. Pour chacun des problèmes, la transformation trouvée par **LP1** produit un problème équivalent Π' avec la même borne inférieure λ rendu explicite en étant la valeur du coût de l'action a_B qui atteint le but. Soit Π'' le problème identique à Π' excepté que le coût de l'action qui atteint le but est 0. Une heuristique admissible calculée pour Π'', obtenue par n'importe quelle autre méthode, peut alors être cumulée à λ.

Nous pouvons clairement résoudre un programme linéaire similaire à **LP1** pour déterminer si, par exemple, il existe un ensemble d'opérations d'**Arc-Mouvement** qui transforment Π en un problème équivalent dans lequel le coût de toutes les actions soit borné par une certaine constante C. En particulier, dans le cas où le problème original contient des actions avec un coût négatif (i.e. des actions qui produisent des profits), nous pouvons chercher un problème équivalent avec des coûts strictement positifs. De même, nous pouvons résoudre un programme linéaire pour déterminer s'il existe une version de Π, transformée par arc-mouvement, dans laquelle $cout(a) > M$ où M est le coût de la meilleure solution trouvée jusque là. Si tel est le cas, alors nous pouvons en déduire que l'action a ne peut appartenir à aucune solution optimale.

Nous avons vu que pour utiliser la transformation donnée par **LP1** il faut normaliser le problème de planification, ce qui peut en augmenter la taille. Il n'est cependant pas indispensable de résoudre cette version normalisée : il est possible de l'utiliser uniquement pour calculer une meilleure borne inférieure, puis de se servir de cette borne pour résoudre le problème originel. Considérons ainsi le problème de l'exemple 2 (version normalisée et transformée du problème de l'exemple 1). Nous pouvons cumuler la borne inférieure que nous avons pu expliciter (puisque le coût de a_B=3)

avec n'importe quelle autre borne inférieure déduite de la version transformée de notre problème par une autre méthode.

6. Conclusion

L'analyse des problèmes de planification optimale nécessite de nouveaux outils par rapport à la planification classique. Un programme linéaire qui ignore l'ordre des actions peut être employé pour déterminer dans un temps polynomial une borne inférieure au coût d'un plan-solution (van den Briel *et al.*, 2007). Nous avons montré que le dual de ce programme linéaire fournit une transformation du problème en un problème équivalent dans lequel le coût de l'action qui atteint le but est exactement égal à cette borne inférieure. Cette transformation est une méthode indépendante qui peut être combinée avec n'importe quelle autre technique pour résoudre des problèmes de planification optimale.

La transformation d'un CSP valué réalisée en résolvant un programme linéaire pour déterminer les déplacements optimaux de coûts entre les fonctions de coût unaires ou d'ordre supérieur est une technique connue sous le nom de consistance optimale d'arc (OSAC) (Cooper *et al.*, 2007). La transformation de problèmes que nous avons décrite dans cet article est clairement analogue à OSAC. Pour obtenir rapidement une approximation de OSAC, plusieurs techniques appliquées aux CSP valués ont été développées. Elles évitent de résoudre un programme linéaire en recherchant les inconsistances locales du CSP obtenu en ne considérant que les valeurs et tuples de coût nul (Cooper *et al.*, 2008; de Givry *et al.*, 2005). D'autres recherches sont encore nécessaires pour déterminer dans quelle mesure des approximations similaires peuvent être appliquées à un problème de planification optimale lorsque le problème ne comportant que des actions de coût nul n'a pas de solution. Si c'est le cas (comme cela l'est pour les CSP valués) nous pourrions en dériver de nombreuses techniques de transformation de problèmes permettant d'arriver à de bons compromis entre le temps de recherche et la qualité de la borne inférieure obtenue.

D'un point de vue théorique, il pourrait être intéressant de voir s'il est possible de déterminer une classe traitable de problèmes de planification optimale pour lesquels la borne inférieure obtenue à partir du programme linéaire est toujours égale au coût réel d'un plan-solution optimal.

7. Bibliographie

Bacchus F., « The AIPS'00 Planning Competition », *AI Magazine*, vol. 22, n° 3, p. 47-56, 2001.

Benton J., van den Briel M., Kambhampati S., « A Hybrid Linear Programming and Relaxed Plan Heuristic for Partial Satisfaction Planning Problems », *Proceedings of 17th International Conference on Automated Planning and Scheduling, (ICAPS'07)*, p. 34-41, 2007.

Bonet B., Geffner H., « Planning as heuristic search », *Artificial Intelligence*, vol. 129, n° 1-2, p. 5-33, 2001.

Bylander T., « The Computational Complexity of Propositional STRIPS Planning », *Artificial Intelligence*, vol. 69, n° 1-2, p. 165-204, 1994.

Bylander T., « A Linear Programming heuristic for optimal planning », *Proceedings of 14th National Conference on Artificial Intelligence (AAAI'97)*, p. 694-699, 1997.

Cooper M. C., de Givry S., Sánchez M., Schiex T., Zytnicki M., « Virtual Arc Consistency for Weighted CSP », *Proceedings of 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, AAAI Press, p. 253-258, 2008.

Cooper M. C., de Givry S., Schiex T., « Optimal Soft Arc Consistency », *Proceedings of 20th International Joint Conference on Artificial Intelligence, (IJCAI'07)*, p. 68-73, 2007.

Cooper M. C., Schiex T., « Arc consistency for soft constraints », *Artificial Intelligence*, vol. 154, n° 1-2, p. 199-227, 2004.

de Givry S., Heras F., Zytnicki M., Larrosa J., « Existential arc consistency : Getting closer to full arc consistency in weighted CSPs », *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, p. 84-89, 2005.

Dechter R., *Constraint Processing*, Morgan Kaufmann, 2003.

Dimopoulos Y., Gerevini A., Haslum P., Saetti A., « The Benchmark Domains of the Deterministic Part of IPC-5 », *Proceedings of 5th International Planning Competition (IPC'2006)*, 2006.

Ghallab M., Nau D., Traverso P., *Automated Planning : Theory and Practice*, Morgan Kaufmann, 2004.

Haslum P., « Additive and Reversed Relaxed Reachability Heuristics Revisited », *Proceedings of 6th International Planning Competition (IPC'2008)*, 2008.

Hoffmann J., Edelkamp S., Thiébaux S., Englert R., dos S. Liporace F., Trüg S., « Engineering Benchmarks for Planning : the Domains Used in the Deterministic Part of IPC-4 », *Journal of Artificial Intelligence Research*, vol. 26, p. 453-541, 2006.

Hoffmann J., Nebel B., « The FF Planning System : Fast Plan Generation Through Heuristic Search », *Journal of Artificial Intelligence Research*, vol. 14, p. 253-302, 2001.

Karmarkar N., « A new polynomial-time algorithm for linear programming », *Combinatorica*, vol. 4, n° 4, p. 373-396, 1984.

Long D., Fox M., « The 3rd International Planning Competition : Results and Analysis », *Journal of Artificial Intelligence Research*, vol. 20, p. 1-59, 2003.

McDermott D., « Using regression-match graphs to control search in planning », *Artificial Intelligence*, vol. 109, n° 1-2, p. 111-159, 1999.

Rintanen J., Heljanko K., Niemelä I., Planning as Satisfiability : Parallel Plans and Algorithms for Plan Search, Technical report n° 216, Institute of Computer Science at Freiburg University, 2005.

Schrijver A., *The Theory of Linear and Integer Programming*, John Wiley & Sons, 1998.

Trustrum K., *Linear Programming*, Routledge & Kegan Paul, 1971.

van den Briel M., Benton J., Kambhampati S., Vossen T., « An LP-Based Heuristic for Optimal Planning », *Proceedings of 13th International Conference, Principles and Practice of Constraint Programming (CP'07)*, p. 651-665, 2007.

van den Briel M., Vossen T., Kambhampati S., « Loosely coupled formulations for Automated Planning : An Integer Programming Perspective », *Journal of Artificial Intelligence Research*, vol. 31, p. 217-257, 2008.

ANNEXE POUR LE SERVICE FABRICATION

A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   RSTI - RIA. Volume 24 – n° 4/2010. Transformation en planification optimale

2. AUTEURS :
   Martin C. Cooper — Marie de Roquemaurel — Pierre Régnier

3. TITRE DE L'ARTICLE :
   Transformation de problèmes de planification optimale

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Transformation en planification optimale

5. DATE DE CETTE VERSION :
   30 avril 2010

6. COORDONNÉES DES AUTEURS :

   – adresse postale :

   IRIT - Université Paul Sabatier
   180, route de Narbonne, 31062 Toulouse, cedex 4

   {cooper,deroquemaurel,regnier}@irit.fr

   – téléphone : 05 62 44 85 25

   – télécopie : 05 61 55 62 39

   – e-mail : cooper@irit.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes.cls`,
   version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com