THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
Simon VERNHES

le vendredi 12 décembre 2014

Titre :

Décomposition des problèmes de planification de tâches basée sur les landmarks

---

École doctorale et discipline ou spécialité :
EDSYS : Informatique

Unité de recherche :
Équipe d'accueil ISAE-ONERA CSDV

Directeur(s) de Thèse :
M. Gérard VERFAILLIE (directeur de thèse)
M. Vincent VIDAL (co-directeur de thèse)

Jury :

M. Héctor GEFFNER, Professeur Universitat Pompeu Fabra - Rapporteur
M. Jörg HOFFMANN, Professeur Universität des Saarlandes - Rapporteur
M. Pierre MARQUIS, Professeur d'université Université d'Artois
M. Pierre SAVÉANT, Ingénieur de recherche Thales Research and Technology
M. Gérard VERFAILLIE, Directeur de recherche Onera - Directeur de thèse
M. Vincent VIDAL, Ingénieur de recherche Onera - Co-directeur de thèse

Résumé

Les algorithmes permettant la création de stratégies efficaces pour la résolution d'ensemble de problèmes hétéroclites ont toujours été un des piliers de la recherche en Intelligence Artificielle. Dans cette optique, la planification de tâches a pour objectif de fournir à un système la capacité de raisonner pour interagir avec son environnement de façon autonome afin d'atteindre les buts qui lui ont été assignés. À partir d'une description de l'état initial du monde, des actions que le système peut exécuter, et des buts qu'il doit atteindre, un *planificateur* calcule une séquence d'actions dont l'exécution permet de faire passer l'état du monde dans lequel évolue le système vers un état qui satisfait les buts qu'on lui a fixés. Le problème de planification est en général difficile à résoudre (PSPACE-difficile), cependant certaines propriétés des problèmes peuvent être automatiquement extraites permettant ainsi une résolution efficace.

Dans un premier temps, nous avons développé l'algorithme LMBFS *(Landmark-based Meta Best-First Search)*. À contre-courant des planificateurs *state-of-the-art*, basés sur la recherche heuristique dans l'espace d'états, LMBFS est un algorithme qui réactualise la technique de décomposition des problèmes de planification basés sur les landmarks. Un landmark est un fluent qui doit être vrai à un certain moment durant l'exécution de n'importe quel plan solution. L'algorithme LMBFS découpe le problème principal en un ensemble de sous-problèmes et essaie de trouver une solution globale grâce aux solutions trouvées pour ces sous-problèmes. Dans un second temps, nous avons adapté un ensemble de techniques pour améliorer les performances de l'algorithme. Enfin, nous avons testé et comparé chacune de ces méthodes permettant ainsi la création d'un planificateur efficace.

Mots-Clés : planification de tâches, landmarks, algorithmes de recherche, intelligence artificielle.

Abstract

The algorithms allowing on-the-fly computation of efficient strategies solving a heterogeneous set of problems has always been one of the greatest challenges faced by research in Artificial Intelligence. To this end, classical planning provides to a system reasoning capacities, in order to help it to interact with its environment autonomously. Given a description of the world current state, the actions the system is able to perform, and the goal it is supposed to reach, a *planner* can compute an action sequence yielding a state satisfying the predefined goal. The planning problem is usually intractable (PSPACE-hard), however some properties of the problems can be automatically extracted allowing the design of efficient solvers.

Firstly, we have developed the Landmark-based Meta Best-First Search (LMBFS) algorithm. Unlike state-of-the-art planners, usually based on state-space heuristic search, LMBFS reenacts landmark-based planning problem decomposition. A landmark is a fluent appearing in each and every solution plan. The LMBFS algorithm splits the global problem in a set of subproblems and tries to find a global solution using the solutions found for these subproblems. Secondly, we have adapted classical planning techniques to enhance the performance of our base algorithm, making LMBFS a competitive planner. Finally, we have tested and compared these methods.

Keywords : classical planning, automated planning, landmarks, search algorithm, artificial intelligence.

À ma famille.

Remerciements

Une thèse est loin d'être un travail solitaire, et je n'aurais jamais pu y arriver sans le soutien d'un grand nombre de personnes que je tiens à remercier, ici, chaleureusement. Tout d'abord, un grand merci à Vincent Vidal et Guillaume Infantes pour tous leurs conseils avisés et leurs disponibilités pendant ces trois ans. Ce fut une expérience riche et agréable de travailler avec eux.

Merci aussi à Gérard Verfaillie, pour son avis critique sur ma thèse et les discussions que nous avons eues qui m'ont permis d'élargir ma vision de la recherche.

Je suis également reconnaissant envers les membres de mon jury : Hector Geffner, et Jörg Hoffmann qui ont accepté d'être rapporteurs, Pierre Marquis qui a présidé le jury, et Pierre Savéant. Merci pour toutes vos remarques critiques à la fois pertinentes et encourageantes.

Malik Ghallab et Pierre Reignier ont participé à mon comité de mi-thèse, et je les en remercie pour leurs commentaires et leurs conseils utiles.

Merci aussi à mes collègues de l'Onera de Toulouse. Et en particulier tous ceux qui sont venus dans mon bureau profitant des pauses cafés pour débattre de sujets plus ou moins sérieux (ou aériens). Merci à Thibault, Pierre, Damien, Alex, Sergio, Caroline, Mehdi et Jack. Et à ceux qui sont arrivés un peu plus tard, notamment Jeremy, Guillaume, Nicolas, Patrick, Jâcques, Francis, Élodie, Sylvain, Pascal, Nicolas, Igor, Florian, Clément, Gaëtan, et tous les nombreux autres que je ne peux citer.

Merci enfin à ma famille qui m'a toujours soutenu pour l'ensemble de mes projets.

Merci à tous.

Table des matières

Table des définitions

Table des théorèmes

Introduction

Un système possédant des capacités intrinsèques d'interaction avec son environnement, peut être défini comme autonome (au moins en partie) s'il n'est pas totalement contrôlé par un humain. Ces systèmes sont étudiés, dans la recherche et dans l'industrie, à différents niveaux d'abstraction : planification de trajectoires, interaction entre plusieurs systèmes, sélection des objectifs à atteindre ou encore modélisation de connaissance.

La planification de tâches propose à de tels systèmes la capacité d'utiliser leurs différentes facultés pour atteindre un ensemble d'objectifs qui leurs seraient fixé. Elle se place entre l'autorité de supervision (humaine ou autonome), définissant des objectifs, et les systèmes automatiques contrôlant ces facultés. Un planificateur a donc l'objectif de trouver un ensemble de tâches à effectuer pour arriver à un but (ou un ensemble de buts) donné. Plusieurs approches ont été, et sont toujours, étudiées.

Une première approche consiste à calculer un plan complet (une séquence d'actions) hors-ligne, en fonction de l'état de départ, de l'ensemble des buts à atteindre, et des possibilités offertes dans le monde modélisé ; en considérant un monde complètement déterministe. Une autre approche, considérant des capacités d'observation complète mais des actions non déterministes, consiste à calculer une politique : pour tous les états du système une action à appliquer est déterminée de manière à optimiser une fonction objectif. Contrairement à l'approche précédente, celle-ci peut permettre au système de se récupérer en cas d'échec d'une action, puisque la politique s'applique à tous les états du système [1]. Enfin, d'autres approches s'intéressent à relâcher plus d'hypothèses, telles que la capacité d'observation complète du système. Chacune de ces techniques possède ses avantages et ses inconvénients, et propose un compromis entre capacité d'adaptation des

1. Une politique peut être partielle, c'est-à-dire définie pour un sous-ensemble d'états du système. Les méthodes de génération de ces politiques partielles s'intéressent aux états les plus problables auquels le système peut accéder, permettant ainsi un compromis entre le temps d'exploration et l'exhaustivité de la politique.

solutions trouvées face des évènements imprévus ou des observations imparfaites et temps de calcul. Dans cette thèse, nous nous interessons au problème de planification de tâches déterministe, c'est-à-dire qu'on fait deux hypothèses : celle de l'observabilité totale du monde, et celle de l'effet déterministe des actionneurs du système.

La planification de tâches déterministe fait l'hypothèse de l'observabilité totale du monde et de l'exécution déterministe des actions, où les problèmes qui peuvent survenir lors de l'exécution d'un plan sont reportés aux autres couches du système, grâce à des méthodes de replanification. Une autre hypothèse, moins visible au premier abord, est de choisir de chercher un plan depuis un état initial vers un but représenté par un ensemble d'objectifs à atteindre. D'autres types de planification, comme ceux basés sur les Processus Décisionnels de Markov (MDP), cherchent à optimiser une fonction de récompense. Dans les deux cas, on peut essayer d'atteindre ces deux objectifs : un but et une optimisation de récompense. D'un côté on peut optimiser la qualité du plan calculé vers le but, et de l'autre on peut intégrer les objectifs dans la fonction à minimiser.

La planification de tâches a débuté avec l'envie de créer des solveurs capables de résoudre les problèmes, notamment avec le General Problem Solver (GPS) [Newell et Simon, 1963] et QA3 [Green, 1969]. Ce dernier propose une représentation du problème de planification en utilisant la logique du premier ordre. Ainsi aujourd'hui, la planification classique conserve un formalisme proche de la théorie de la logique du premier ordre même si elle s'en est détaché depuis le planificateur STRIPS [Fikes et Nilsson, 1972] que nous présenterons dans le premier chapitre.

Dans une première partie, nous présenterons les bases sur lesquelles repose notre travail. Le premier chapitre présente les différents modèles de la planification classique et la manière dont ils sont utilisés. Le deuxième chapitre décrit une technique de résolution du problème de planification grâce à l'exploration en avant dans l'espace d'états et la conception d'heuristiques génériques. Ensuite, le troisième chapitre introduit les landmarks, des points-clés présents dans tous les plans solutions d'un problème de planification, permettant ainsi la conception de nouvelles heuristiques et méthodes de recherche. Enfin, le chapitre quatre présente différents planificateurs et les compare grâce aux résultats obtenus lors des dernières compétitions académiques de planification.

Dans un second temps, nous présentons les apports de cette thèses. Avec le chapitre cinq, nous commencerons par introduire l'algorithme de planification que nous avons conçu. L'idée générale est de constuire une séquence de sous-buts, les landmarks, à atteindre en séquence avant d'atteindre le but global du problème et donc de chercher un plan entre ces buts. Le chapitre six affine cet algorithme en ajoutant et testant diverses techniques. Le chapitre sept présente des travaux préliminaires de parallélisation de l'algorithme. Enfin, le chapitre huit s'intéresse à la recherche de landmarks conjonctifs afin d'améliorer les buts intermédiaires de notre algorithme.

L'ensemble de nos travaux, excepté le travail sur les landmarks conjonctifs, a été publié :

Premières réflexions sur un algorithme de décomposition d'un problème de planification en un ensemble de sous-problèmes :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [août 2012].
« The Landmark-based Meta Best-First Search Algorithm for Classical Planning ». Dans : *Proceedings of the 5th European Starting AI Researcher Symposium (STAIRS-2012)*. T. 241. Frontiers in Artificial Intelligence and Applications. IOS Press, p. 336–347

Présentation de l'algorithme *Landmark-based Meta Best-First Search* :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [août 2013b].
« Problem Splitting using Heuristic Search in Landmark Orderings ». Dans : *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-2013)*. AAAI Press

Première parallélisation de l'algorithme :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [juin 2013a].
« Landmark-based Meta Best-First Search Algorithm : First Parallelization Attempt and Evaluation ». Dans : *Proceedings of the 5th ICAPS Workshop on Heuristics and Search for Domain-independent Planning (HSDIP-2013)*, p. 44–52

Première partie

Contexte

1

La planification classique

Une première étape nécessaire à tout algorithme générique de résolution de problèmes de planification est un langage de description représentant l'ensemble des problèmes que l'on souhaite résoudre. Ce chapitre introduit donc les différentes représentations utilisées aujourd'hui par les méthodes de planification classique. Tout d'abord, on détaillera le modèle théorique de planification classique, basé sur un ensemble fini d'états atteignables. On présentera ensuite deux représentations utilisées au sein des planificateurs : la représentation STRIPS, définissant les objets du monde grâce à des variables booléennes et la représentation SAS$^+$, définissant les objets du monde grâce à des variables multivaluées. Enfin, on introduira PDDL, un langage de description synthétique permettant une description de haut niveau.

Les chapitres suivants évoqueront quelques résultats classiques, tels que les algorithmes de recherche standards associés à des heuristiques spécifiques à la planification, ainsi qu'un ensemble d'autres techniques telles que les *landmarks*. Enfin, on s'attachera à décrire différents planificateurs existants et à les comparer en se basant sur les résultats des dernières compétitions académiques.

Sommaire

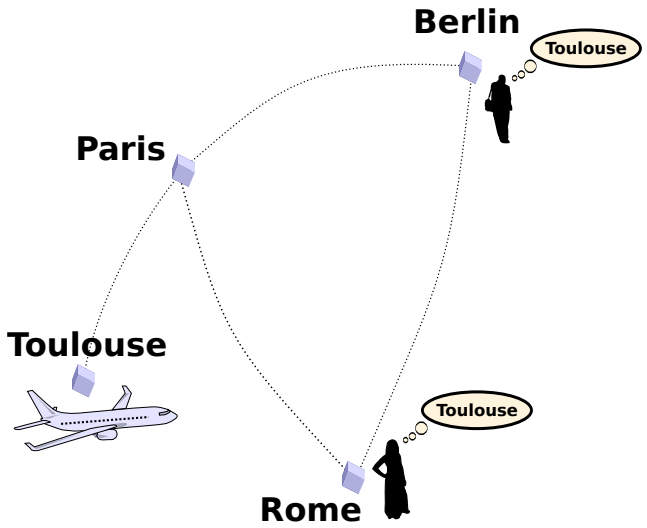


FIGURE 1.1 : Un problème simple du domaine Zenotravel

1.1 Exemple de problème de planification

Pour modéliser un problème de planification de tâches déterministe, on doit d'abord définir le monde qui est accessible au système que l'on veut représenter. Afin de présenter les différentes notations et éléments que nous allons manipuler, nous utiliserons un exemple du domaine Zenotravel qui modélise un problème de partage de ressources pour le transport de passagers entre plusieurs villes. La figure 1.1 nous montre un problème avec quatre villes (Berlin, Rome, Paris et Toulouse), un avion capable de se déplacer entre ces quatre villes et enfin deux passagers qui souhaiteraient se déplacer de Berlin et Rome vers Toulouse.

Pour commencer à modéliser ce problème, on peut tout d'abord identifier les éléments immuables du problème, comme l'avion ou les passagers. Ceci nous permet de déterminer les *prédicats statiques* [2] :

- $(\mathit{avion}\ a)$ représente l'avion ;
- $(\mathit{passager}\ p_1), (\mathit{passager}\ p_2)$ sont les deux passagers ;
- $(\mathit{ville}\ b), (\mathit{ville}\ r), (\mathit{ville}\ p), (\mathit{ville}\ t)$ sont les lieux (respectivement Berlin, Rome, Paris et Toulouse) ;
- $(\mathit{route}\ t\ p)$ précise qu'il existe une route de Toulouse vers Berlin, et il faut donc y ajouter la route inverse $(\mathit{route}\ p\ t)$. Et de même pour les 6 autres routes.

2. Tout au long de ce mémoire, nous garderons une notation proche de la représentation PDDL (que nous détaillerons plus tard). Un prédicat du premier ordre se note habituellement $\mathit{avion}(a)$.

Ensuite, il faut modéliser les éléments dynamiques du système, tels que la position de l'avion ou des passagers, nous permettant ainsi de définir les *prédicats dynamiques* (ou *fluents*) :

- $(\texttt{dans}\ p_1\ a)$ si le passager p_1 est dans l'avion a ;
- $(\texttt{à}\ a\ t)$ si l'avion a est à Toulouse ;
- $(\texttt{à}\ p_1\ b)$ si le passager p_1 est à Berlin.

Partant de ces prédicats, on peut maintenant décrire les différents états du monde. Pour cela, on décrit le monde courant avec une conjonction de l'ensemble des prédicats vrais à cet état. Les fluents non présents dans cette conjonction sont considérés comme faux dans cet état[3]. Par exemple, l'état défini par la figure 1.1 est la conjonction des prédicats[4] :

$$\begin{gathered}(\texttt{avion}\ a) \wedge (\texttt{passager}\ p_1) \wedge (\texttt{passager}\ p_2) \wedge \\ (\texttt{ville}\ b) \wedge (\texttt{ville}\ r) \wedge (\texttt{ville}\ p) \wedge (\texttt{ville}\ t) \wedge \\ (\texttt{à}\ a\ t) \wedge (\texttt{à}\ p_1\ b) \wedge (\texttt{à}\ p_2\ r)\end{gathered}$$

Enfin, le système est capable de modifier l'état du monde grâce à ses *actionneurs*. Ces capacités sont modélisées sous forme d'*actions* qui vont modifier les fluents du problème :

- $(\texttt{déplacer}\ a\ t\ p)$ permet de déplacer l'avion a de Toulouse à Paris ;
- $(\texttt{embarquer}\ p_1\ a\ t)$ permet d'embarquer le passager p_1 dans l'avion a se trouvant tous les deux dans la ville de Toulouse ;
- $(\texttt{débarquer}\ p_1\ a\ t)$ permet l'opération inverse.

Ces actions sont définies sous la forme de triplets. D'une part, les préconditions d'une action, définies par un ensemble de prédicats, précisent si l'action est applicable dans l'état courant du monde, c'est-à-dire si les conditions d'exécution de l'action sont réunies. D'autre part, les effets sont constitués de deux ensembles regroupant les ajouts et les suppressions des fluents à appliquer si l'action est effectuée. Par exemple, pour embarquer le passager p_1 à Toulouse $(\texttt{embarquer}\ p_1\ a\ t)$, il faut que l'avion et le passager se trouvent tous les deux dans la bonne ville $(\texttt{à}\ a\ t) \wedge (\texttt{à}\ p_1\ t)$. Si cette action est appliquée, il faut alors retirer le fluent $(\texttt{à}\ p_1\ t)$ et ajouter $(\texttt{dans}\ p_1\ a)$.

3. Hypothèse du monde clos

4. L'ensemble des prédicats n'est pas représenté, il faudrait aussi ajouter les prédicats statiques représentant les différents liens entre les villes tel que $(\texttt{route}\ t\ p)$. Les prédicats statiques sont présents dans tous les états du monde.

1.2 Modèle de la planification classique basé sur les états

De manière formelle, la planification classique peut être définie de la façon suivante :

Définition 1 : ***Modèle du problème de planification de tâches***

On peut définir un problème de planification de tâches comme un 5-uplet $\Pi = \langle S, I, S_G, A, f \rangle$ *tel que :*

- S *est un ensemble fini d'états (l'espace d'états) ;*
- $I \in S$ *représente l'unique état initial ;*
- $S_G \subseteq S$ *est un ensemble d'états buts ;*
- A *est un ensemble d'actions ;*
- f *est une fonction partielle de transition entre états* $f : S \times A \longrightarrow S$.

Dans le modèle initial les actions ont un coût unitaire, mais on peut y ajouter une fonction qui associe à chaque action de l'ensemble A une fonction coût définie par $cost : A \longrightarrow \mathbb{R}$. Dans le cas où la fonction $cost$ n'est pas précisée on considérera simplement que les actions ont un coût unitaire : $(\forall a \in A)\ cost(a) = 1$.

Ce modèle, étudié comme problème de recherche d'une séquence d'actions (éventuellement optimale) permettant depuis l'état initial I d'atteindre un des états buts de l'ensemble S_G, peut être représenté sous la forme d'un graphe orienté où les états sont les sommets et les arcs sont déterminés par la fonction partielle f. Le plan (séquence d'actions) recherché est alors simplement le chemin depuis l'état initial I, jusqu'à un des buts du problème.

Cette première représentation conduit donc à de premières pistes de réflexion pour sa résolution : typiquement, une implication forte avec la théorie des graphes. Cependant, on ne pourra expliciter le graphe complet des problèmes de planification ni en mémoire pour l'exploration ni en format d'entrée car, en pratique, les problèmes de grande taille ne sont pas représentables de cette manière. En effet, si on considère le petit problème de type Zenotravel présenté à la section 1.1, on obtient déjà 16384 états potentiels. Certes, une partie d'entre eux sont impossibles : en effet, certains fluents peuvent être *mutuellement exclusifs*, tel que le fait qu'un passager ne peut être dans plusieurs villes à la fois. On peut donc calculer une meilleure estimation en considérant les fluents mutuellement exclusifs, limitant à 100 le nombre d'états atteignables. Cependant, même si dans notre exemple jouet le nombre d'états est faible, il reste exponentiel en nombre d'avions et de passagers. Un résumé de l'explosion combinatoire pour ce problème est présenté dans le tableau 1.1. Donc, cette représentation théorique n'est pas utilisable en pratique.

5. Il suffit de compter le nombre de positions possibles pour chaque avion (toutes les villes v) et chaque passager (toutes les villes et tous les avions $a + v$). De plus, comme toutes les entités sont

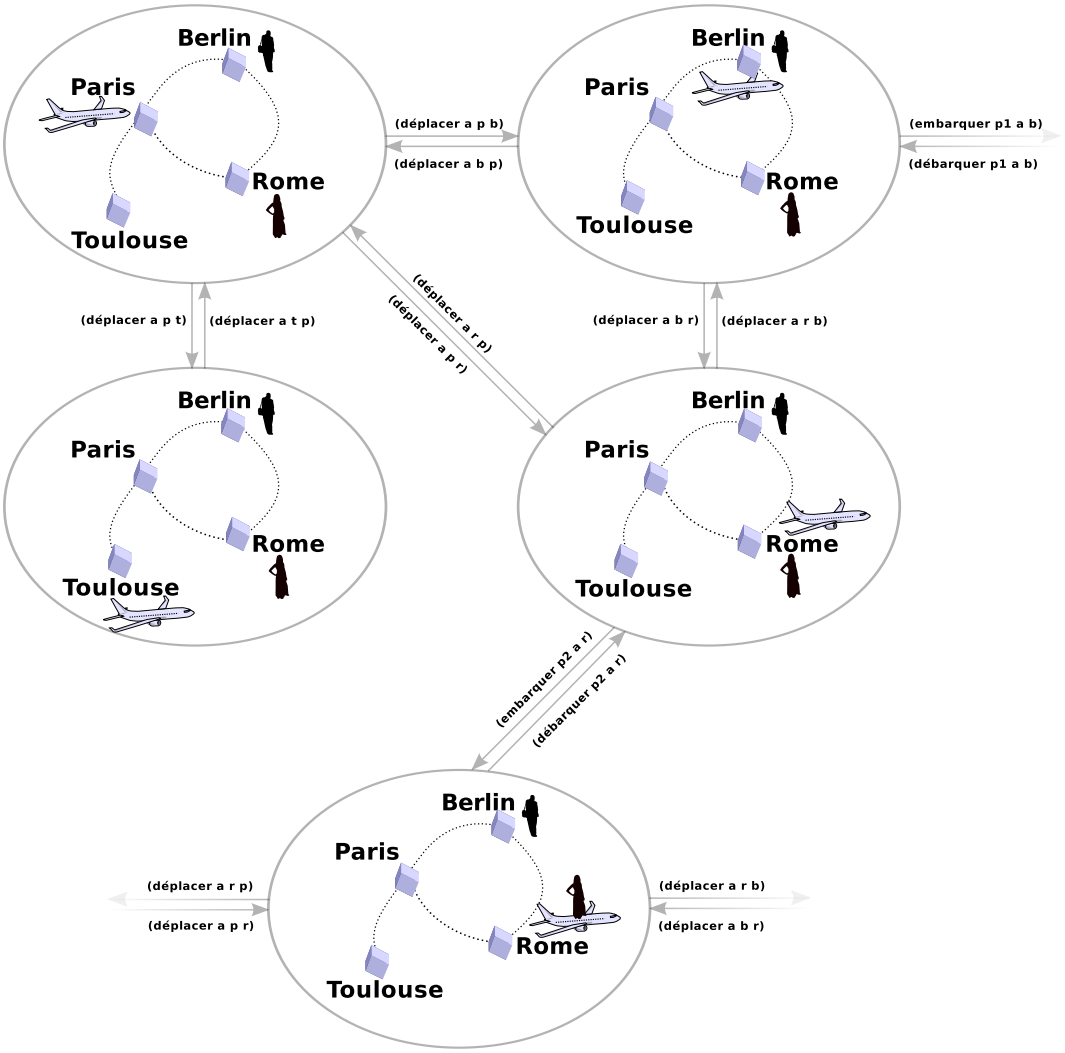


FIGURE 1.2 : Graphe représentant certains états d'un problème de type Zeno-travel.

p	a	v	États possibles : $2^{pa+av+pv}$	États atteignables : $v^a(a+v)^p$
2	1	4	$2^{14} = 16384$	100
3	1	4	$2^{19} = 524288$	500
4	1	4	$2^{24} = 16777216$	2500
10	5	20	$2^{350} \approx 10^{106}$	$\approx 10^{21}$

TABLEAU 1.1: Nombre d'états possibles pour le problème Zenotravel où p est le nombre de passagers, a le nombre d'avions et v le nombre de villes. Si on ne considère pas les fluents mutuellement exclusifs, le nombre d'états possibles est $2^{pa+av+pv}$ (c'est-à-dire $2^{\text{nombre de fluents}}$). Si on prend en compte l'exclusion mutuelle, on a alors $v^a(a+v)^p$ états atteignables[5].

1.3 Le modèle STRIPS

STRIPS [Fikes et Nilsson, 1972] est un planificateur qui a introduit une représentation des problèmes de planification de tâches toujours très utilisée. Par habitude, et par abus de langage, on appelle maintenant STRIPS (ou le modèle STRIPS) cette représentation. Ici, un sous-ensemble des fluents décrit *l'état* du monde, ce qui rend cette représentation proche de celle présentée en section 1.1.

Définition 2 : *Problème de planificiation STRIPS*

On peut définir un problème de planification de tâches comme un 4-uplet $\Pi = \langle F, A, I, G \rangle$ *tel que :*

- F *est l'ensemble des fluents ;*
- A *est l'ensemble des actions ;*
- $I \subseteq F$ *représente l'unique état initial ;*
- $G \subseteq F$ *est un ensemble de fluents représentant le but à atteindre.*

L'ensemble G représentant le but ne définit pas nécessairement un état complet. Il peut donc y avoir plusieurs états $G_i \subseteq F$ satisfaisant le but (tels que $G \subseteq G_i$).

De manière équivalente, on pourrait définir la représentation STRIPS grâce à un ensemble de variables booléennes représentant les fluents. Un état serait une assignation de valeurs à cet ensemble et les actions modifieraient les valeurs de certaines de ces variables.

Définition 3 : *État*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On dit que $s \subseteq F$ *est un état du monde. Un état du monde est donc un ensemble de fluents de* F.

indépendantes les unes des autres, il suffit de multiplier le nombre de positions de chaque entité : $v^a(a+v)^p$

Les actions, permettant les transitions entre les états, sont définies comme :

Définition 4 : *Action*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
Une action a *est un triplet* $\langle \text{pre}(a), \text{add}(a), \text{del}(a) \rangle$ *où :*

préconditions	$\text{pre}(a) \subseteq F$	*l'ensemble des fluents* nécessaires
ajouts	$\text{add}(a) \subseteq F$	*l'ensemble des fluents* ajoutés
retraits	$\text{del}(a) \subseteq F$	*l'ensemble des fluents* supprimés

Dans notre exemple fil rouge Zenotravel, l'action permettant de déplacer un avion a de la ville v_1 à la ville v_2 peut-être définie comme :

Exemple 1 : *Action* $\alpha = (\texttt{déplacer a } v_1\ v_2)$

$$\begin{aligned} \text{pre}(\alpha) &= \{(\grave{a}\ a\ v_1), (\texttt{route}\ v_1\ v_2)\} \\ \text{add}(\alpha) &= \{(\grave{a}\ a\ v_2)\} \\ \text{del}(\alpha) &= \{(\grave{a}\ a\ v_1)\} \end{aligned}$$

Les actions, exécutées uniquement si les préconditions sont remplies (c'est-à-dire si $\text{pre}(a) \subseteq s$), sont appliquées en ôtant les fluents appartenant aux retraits de l'actions *puis* en adjoignant les fluents de l'ensemble des ajouts. Plus formellement :

Définition 5 : *Application des effets d'une action*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* s *un état du monde.*
On note $\text{apply}(s, a)$ *la fonction appliquant l'action* a *à l'état* s *définie tel que :*

$$\text{apply} : \left| \begin{array}{ccc} E \subseteq 2^F \times A & \longrightarrow & 2^F \\ (s, a) & \longmapsto & (s \setminus \text{del}(a)) \cup \text{add}(a) \end{array} \right.$$

où E *est l'ensemble des couples état-action dont l'action est applicable pour cet état :*

$$E = \left\{ (s, a) \in 2^F \times A \;\middle|\; \text{pre}(a) \subseteq s \right\}$$

Pour plus d'aisance, on s'autorise à noter :

$$s[a] = \text{apply}(s, a)$$

La résolution d'un problème de planification de tâches consiste à trouver une suite d'actions applicables depuis l'initial et conduisant au but. On définit donc une séquence d'actions :

Définition 6 : *Séquence d'actions*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On note $\langle a_1, a_2, \ldots, a_n \rangle$*, avec* $\big(\forall i \in [\![1, n]\!]\big)\, a_i \in A$*, une séquence d'actions qui est une suite ordonnée d'actions avec répétition possible.*
On note $\langle\rangle$ *la séquence d'actions vide (ne comportant aucune action).*

Définition 7 : *Plan*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et* $\rho = \langle a_1, a_2, \ldots, a_n \rangle$*, une séquence d'actions.*
On dit que ρ *est un plan si et seulement si l'application successive des actions de ladite séquence respecte les préconditions de ces actions. C'est-à-dire :*

$$\begin{cases} \mathrm{pre}(a_1) \subseteq I \\ \big(\forall i \in [\![2, n]\!]\big)\, \mathrm{pre}(a_i) \subseteq I[a_1][a_2]\ldots[a_{i-1}] \end{cases}$$

On note $\mathcal{P}(\Pi)$*, l'ensemble des plans.*
De plus, on étend l'application apply *pour un plan* $\rho \in \mathcal{P}(\Pi)$ *:*

$$\begin{cases} s[\rho] = \mathrm{apply}(s, \rho) = \mathrm{apply}(\ldots \mathrm{apply}(\mathrm{apply}(s, a_1), a_2), \ldots, a_n) \\ s[\langle\rangle] = s \end{cases}$$

On s'autorise à noter ρ_i *l'action* a_i *du plan* $\rho = \langle a_1, a_2, \ldots, a_i, \ldots, a_n \rangle$*, et* $\rho_{i \to i'} = \langle a_i \ldots, a_{i'} \rangle$*, une sous-séquence d'actions de* ρ*. La première action d'un plan est toujours indexée par* 1*.*
Finalement, on note aussi $|\rho|$*, le nombre d'actions du plan* ρ*.*

Une solution d'un problème de planification est un plan dont l'application depuis l'état initial mène jusqu'à un des états buts. On appelle une telle séquence d'actions un plan solution :

Définition 8 : *Plan solution*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et* $\rho \in \mathcal{P}(\Pi)$ *un plan.*
On dit que ρ *est un plan solution si et seulement si* $G \subseteq I[\rho]$*.*
On note $\mathcal{P}^\star(\Pi) = \big\{\rho \in \mathcal{P}(\Pi) \;\big|\; G \subseteq I[\rho]\big\}$*, l'ensemble des plans solutions (ou* $\mathcal{P}^\star$ *quand non ambigu).*

Définition 9 : *Concaténation de plans*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et* $(\rho, \mu) \in \mathcal{P}(\Pi)^2$ *deux plans. On note* $\rho \circ \mu$ *la concaténation des plans* ρ *et* μ *telle que :*

$$\rho \circ \mu = \begin{cases} \langle\rangle & \text{Si } \rho = \mu = \langle\rangle \\ \rho & \text{Si } \mu = \langle\rangle \\ \mu & \text{Si } \rho = \langle\rangle \\ \langle \rho_1, \ldots, \rho_{|\rho|}, \mu_1, \ldots, \mu_{|\mu|} \rangle & \text{Sinon} \end{cases}$$

Si on associe une fonction de coût aux actions, alors on notera $cost(\rho) = \sum_{i=1}^{|\rho|} cost(\rho_i)$.

La complexité du problème de décision associé à la représentation STRIPS a été étudiée et prouvée difficile.

Théorème 1 : *Complexité du modèle STRIPS*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
Le problème de décision défini par la question « Existe-t-il un plan solution pour le problème Π ? *» est PSPACE-complet [*Bylander, *1994].*

De plus, prouver l'existence d'un plan ayant moins de k actions (problème d'optimisation associé) est aussi PSPACE-complet [Bylander, 1994] ; [Erol, Nau et Subrahmanian, 1995].

1.4 Le modèle SAS⁺

Plus récemment, une représentation différente basée sur des variables à valeurs dans des ensembles finis a pris de l'importance. Ce formalisme, SAS⁺ [Bäckström et Nebel, 1995], permet une vision un peu plus naturelle des problèmes de planification. Par ailleurs, les domaines de valeur des variables permettent de s'affranchir, intrinsèquement, de certaines exclusions mutuelles entre fluents (typiquement celles précisées précédemment, comme l'identification de la position des avions ou des passagers dans le domaine Zenotravel).

Définition 10 : *Problème de planificiation SAS⁺*

On peut définir un problème de planification de tâches SAS⁺ comme un 4-uplet $\Pi = \langle V, A, I, G, \rangle$ *tel que :*

- V *est l'ensemble des variables (chaque variable a un domaine fini de valeurs* D_V *associé) ;*
- A *est l'ensemble des actions ;*
- I *représente l'état initial ;*
- G *représente l'ensemble des états buts.*

Un état est une assignation (complète) de toutes les variables à une valeur de leur domaine. Le but G *est une assignation partielle de variables.*

De la même manière qu'en STRIPS, les actions ont des préconditions et des effets. Les préconditions précisent les valeurs attendues des variables, et les effets changent les valeurs des variables.

La complexité du modèle SAS⁺ a été étudiée et prouvée PSPACE-complète [Bäckström et Nebel, 1995] ; [Bäckström et al., 2012]. D'autre part, les modèles SAS⁺ et STRIPS sont tout aussi expressifs [Bäckström, 1995].

1.5 PDDL — Planning Domain Definition Language

Historiquement, le *Action Description Language* (ADL) [Pednault, 1989] s'est proposé comme extension au langage STRIPS. Plus tard, le *Planning Domain Definition Language* (PDDL) [McDermott et al., 1998] a amené une standardisation du language de description utilisé par la plupart des planificateurs grâce à son utilisation dans la compétition de planification académique [6].

PDDL permet une représentation claire et succincte d'un problème de planification. En effet, il donne la possibilité de représenter les tâches de planification grâce à des variables, des prédicats du premier ordre et des connecteurs logiques. Le problème de planification est séparé en deux parties, la description du *domaine* et la description du *problème*. Le domaine consiste en une description générale d'un monde : un ensemble de types, éventuellement quelques objets (constantes) communs à tous les problèmes associés et toutes les actions possibles dans ce monde, représentées notamment grâce à des prédicats du premier ordre. Le problème peut être vu comme une instanciation du domaine, où l'on vient présenter un ensemble d'objets (constantes) spécifiques au problème, l'état initial, et le but à atteindre. Un exemple de couple domaine et problème pour l'exemple Zenotravel se trouve en annexe B.

La représentation factorisée des prédicats et des actions à l'aide de variables correspond, de fait, à un regroupement de prédicats ou d'actions. Les variables, contrairement aux valeurs (a, p_1, etc.), sont précédées par le marqueur « ? ». On

6. *International Planning Competition* (IPC) — `http://ipc.icaps-conference.org`.

peut instancier le fluent $(dans\ ?p\ ?a)$ avec des constantes : par exemple, le passager p_1 et l'avion a, donnant un fluent instancié $(dans\ p_1\ a)$. De la même manière, l'action de déplacement de l'avion peut être factorisée : $(déplacer\ ?a\ ?v_1\ ?v_2)$.

Plusieurs extensions [Fox et Long, 2003] ; [Edelkamp et Hoffmann, 2004] ; [Gerevini et Long, 2006] ont été ajoutées lors des différentes IPC. Certaines augmentent l'expressivité du langage telles que les actions temporelles, les *timed initial literals*, les axiomes, ou encore différentes métriques pour une évaluation quantitative de la qualité d'un plan. D'autres extensions ajoutées à PDDL[7], telles que les préconditions négatives[8], les préconditions disjonctives, les effets conditionnels ou encore les quantificateurs ne sont finalement que du sucre syntaxique rendant la modélisation des problèmes plus aisé : elles n'augmentent pas l'expressivité de la représentation STRIPS au sens où les problèmes PDDL peuvent être modélisés en STRIPS, et inversement[9]. Cependant, il est important de noter que même si la puissance expressive des deux langages reste la même, une action composée de prédicats du premier ordre en PDDL utilisant des effets conditionnels amène à la création d'un ensemble exponentiel d'actions lors de la compilation en STRIPS. L'intérêt de la puissance expressive et de la puissance expressive *perçue* des différentes extensions PDDL est plus amplement discutée dans [Thiébaux, Hoffmann et Nebel, 2005] ; [Nebel, 2011].

Ce langage donne une représentation unifiée des problèmes de planification, notamment pour la constitution d'un ensemble de benchmarks permettant de comparer les performances des différentes méthodes de résolution des problèmes de planification. Cependant, même si depuis les premières IPC, les planificateurs considèrent PDDL comme un fichier de description standard des problèmes, ils utilisent dans le cœur du programme plutôt des représentations proche de STRIPS ou, plus récemment, de SAS$^+$.

1.6 Motivation

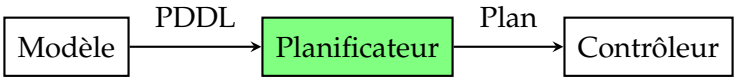
Finalement, on peut dire que le modèle classique peut être vu comme un graphe complet, inutilisable tel quel pour la résolution de problème de planification à cause de l'explosion combinatoire à laquelle il est sujet. Les algorithmes classiques utilisent donc comme représentation interne une modélisation plus concise telle que STRIPS ou SAS$^+$. D'autre part, de manière à être facilement compréhensible et modélisable par un humain, le langage PDDL a été proposé et est devenu un standard pour la plupart des planificateurs. Il est cependant utilisé en tant que modèle d'entrée qui est ensuite instancié vers une représentation interne efficace.

Maintenant que nous avons défini le modèle de la planification classique et ses différentes représentations, on peut se demander quelle est la motivation derrière

7. En fait, il s'agit de toutes les extensions provenant d'ADL.
8. Par exemple, ne pas avoir un passage dans l'avion lors de la maintenance de l'appareil.
9. Il existe d'ailleurs un outil, *adl2strips*, permettant la compilation d'un problème ADL en problème STRIPS [Hoffmann et al., 2006].

ce domaine de recherche. Comme tout domaine de l'intelligence artificielle, la motivation initiale de la planification de tâches est de participer au développement de systèmes ou d'agents capables de prendre des décisions par eux-mêmes. Le problème clef est de sélectionner quelle est la prochaine action à exécuter [10]. Plusieurs approches tentent de résoudre ce problème, pouvant être utilisées de manière complémentaire ou non, comme les solutions basées sur de l'apprentissage [Sutton et Barto, 1998]. Comme vu précédemment, nous nous intéresserons à la planification via la synthèse de plans, qui est une approche basée sur les modèles où les problèmes sont décrits par un expert (un humain, ou un autre module autonome) et où le système de contrôle se base sur un plan dérivé par un planificateur.

Un planificateur est un solveur capable de résoudre un ensemble de problèmes. Il prend en entrée un modèle décrivant le problème (en PDDL) et calcule un contrôleur capable de résoudre ce problème (un plan).



Il existe différents types de planificateurs, caractérisés à différents niveaux outre leur capacité à respecter les différentes possibilités offertes par PDDL. Il convient en premier lieu de distinguer les planificateurs temporels des planificateurs classiques. Bien que ces deux domaines aient une base commune et utilisent des techniques communes, les algorithmes de recherche dans l'espace d'états, très efficaces en planification classique ne sont pas directement adaptable à l'aspect de concurrence des actions présent en planification temporelle [Cushing et al., 2007] ; [Coles et al., 2008]. Ensuite, on peut classer les planificateurs en fonction de la qualité de la solution recherchée : nous avons tout d'abord la classe des planificateurs cherchant un plan *optimal* (et prouvant l'optimalité de celui-ci), puis ceux cherchant la solution de meilleure qualité dans un temps imparti *(satisficing)* [11], et enfin ceux cherchant une solution le plus rapidement possible *(agile)* au détriment de la qualité du plan. Dans une autre catégorie, certains s'intéressent aussi à retourner un plan satisfaisant le plus de sous-buts possibles (ou maximiser une utilité sur un sous-ensemble du but) dans les problèmes où aucune solution ne peut satisfaire tous les buts existants *(over-subscribring)*.

Dans notre cas, nous nous intéresserons aux approches de type *satisficing* et *agile*.

10. On peut calculer à l'avance une séquence d'actions complète éventuellement optimale. De manière plus réactive, on peut s'intéresser uniquement aux prochaines actions à exécuter [Korf, 1990].

11. Ces planificateurs sont souvent des planificateurs *anytime*, capable de retourner une solution à n'importe quel moment. En réalité, il s'agit souvent de chercher une première solution rapidement puis d'utiliser le temps restant pour trouver un plan de meilleure qualité. Cependant, certains s'intéressent à utiliser le temps imparti directement en tant que critère de recherche [Ruml et Do, 2007].

2

PLANIFICATION PAR RECHERCHE HEURISTIQUE

La recherche dans un espace guidée par une heuristique est une méthode couramment utilisée pour résoudre les problèmes de plus court chemin. De plus, des méthodes d'analyse des problèmes de planification permettant l'extraction automatique d'heuristiques ont permis à la recherche heuristique de devenir un des axes majeurs pour la résolution des problèmes de planification de tâches. On appelle ces techniques *la recherche heuristique dans l'espace des états*.

Nous allons donc voir comment sont extraites ces heuristiques génériques, indépendantes du domaine, quels algorithmes de recherche dans un graphe sont couramment utilisés en planification, et enfin quelques améliorations apportées à ces algorithmes spécifiquement pour la planification.

Sommaire

2.1 Graphe de planification

2.1.1 Graphplan

Avant de détailler les principales méthodes de recherche dans l'espace d'états, nous allons tout d'abord introduire une technique qui a mené à la création d'heuristiques génériques toujours utilisées actuellement. Au milieu des années 1990, Graphplan [Blum et Furst, 1997] montre une nouvelle manière de résoudre les problèmes de planification de tâches via la création d'un graphe de planification. Ce dernier est une façon condensée d'exprimer l'ensemble des solutions d'une taille donnée d'un problème de planification sous forme d'un graphe orienté découpé en niveaux.

Ce graphe est constitué d'une structure récursive à deux niveaux, alternant niveau de fluents et niveau d'actions : $\langle \lambda_0^f, \lambda_0^a, \lambda_1^f, \lambda_1^a, \ldots, \lambda_n^a, \lambda_n^f \rangle$. Chaque élément de chaque niveau est un nœud du graphe.

Les arcs relient uniquement des nœuds de niveaux successifs. Pour chaque niveau d'action λ_i^a, et pour chaque action $a \in \lambda_i^a$, il existe un arc reliant chaque précondition de a du niveau λ_i^f à l'action a du niveau λ_i^a. Il existe aussi un arc pour chaque effet de l'action a : entre cette action et le fluent du niveau λ_{i+1}^f.

Le premier niveau de fluent est simplement l'ensemble des fluents de l'état initial. Ensuite, niveau par niveau, on peut générer le graphe complet jusqu'à atteindre le but (ou un point fixe). Pour les niveaux d'actions, on ajoute toutes les actions exécutables (et les actions Noop [12]) grâce aux fluents du niveau précédent en tenant compte de l'exclusion mutuelle, c'est-à-dire qu'on n'ajoute pas d'actions dont deux préconditions sont mutuellement exclusives. Deux fluents sont considérés comme mutuellement exclusifs (*mutex*) à un niveau du graphe si tous les couples d'actions qui produisent ces deux fluents et qui sont applicables au niveau d'action précédent sont mutuellement exclusives. Deux actions sont considérées comme mutuellement exclusives si leurs préconditions sont *mutex* au niveau précédent, ou si les effets d'une action interfèrent négativement avec les préconditions de l'autre. Pour les niveaux de fluents, on ajoute simplement les fluents ajoutés par les actions du niveau précédent.

À chaque création d'un niveau d'actions ou de fluents, on calcule donc les *mutexes* en fonction du niveau précédent. Les *mutexes* peuvent apparaître et disparaître au fil de la construction des niveaux du graphe.

Pour extraire un plan à partir du graphe, il faut partir du dernier niveau de fluents λ_n^f. On choisit un ensemble d'actions du niveau λ_n^a non mutex permettant d'accomplir l'ensemble des fluents du but. On répète ensuite ce processus au niveau λ_{n-1}^f sur l'ensemble des préconditions des actions choisies. On itère ainsi jusqu'à niveau de fluents initial. Si à un niveau i, il n'existe pas d'ensemble d'actions non mutex permettant d'obtenir l'ensemble des sous-buts courants, il faut faire un *backtrack* et choisir un ensemble d'actions différent à niveau supérieur à i. Il existe

12. Noop : action nulle associée à un fluent f (ayant comme précondition f et ajoutant f)

plusieurs manières de sélectionner un ensemble d'actions, on peut par exemple privilégier les actions Noop.

2.1.2 Graphe de Planification Relaxé (RPG)

Un graphe de planification relaxé est un graphe de planification appliqué au problème relaxé. Ce problème relaxé est obtenu en retirant du problème initial les effets négatifs des actions.

Définition 11 : *Problème relaxé* Π^+

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On dit que $\Pi^+ = \langle F, A^+, I, G \rangle$ *est le problème relaxé de* Π*, avec* A^+ *l'ensemble des actions de* A *sans les retraits des fluents, c'est-à-dire :*

$$A^+ = \left\{ \langle pre(a), add(a), \emptyset \rangle \mid \langle pre(a), add(a), del(a) \rangle \in A \right\}$$

Exemple 2 : *Zenotravel – Problème relaxé*

La version relaxée du problème présenté dans la figure 1.1 consisterait à pouvoir avoir l'avion et les passagers à plusieurs endroits en même temps. Par exemple, si on déplace l'avion de Toulouse à Rome, dans le problème relaxé, l'avion se trouverait à la fois à Toulouse et à Rome.

Trouver une solution optimale à un problème de planification relaxé Π^+ est toujours difficile (en fait, NP-dur [Bylander, 1994]). Il est cependant possible de trouver des solutions sous-optimales en temps polynomial. Pour cela, on construit un *graphe de planification* du problème relaxé suivi d'une recherche de plan gloutonne (cf. h^{ff}, section 2.2.4). On peut simplement définir le graphe de planification du problème relaxé comme :

Définition 12 : *Graphe de Planification Relaxé (RPG)*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification. On note alors* $GP(\Pi^+)$ *le graphe de planification relaxé (ou graphe de planification appliqué au problème relaxé* Π^+*).*

Le RPG se compose de la même manière que le graphe de planification mais sans tenir compte des *mutexes*, puisqu'il n'y en a simplement pas dans le problème relaxé. La figure 2.1 présente un RPG complet pour un problème très simple du domaine Zenotravel composé d'un seul passager et de trois villes.

D'une certaine manière, le graphe de planification est une vision non déterministe du problème de planification où toutes les transitions possibles d'un niveau de fluents du graphe sont activées. C'est une manière de représenter les problèmes de planification sous forme d'un graphe de taille abordable, et ainsi d'en extraire certaines propriétés telle que l'estimation de la distance par rapport au but (heuristiques), la détection de fluents clés pour la résolution du problème (landmarks) ou encore la sélection de l'action la plus adéquate pour atteindre un fluent *(best-supporter)*.

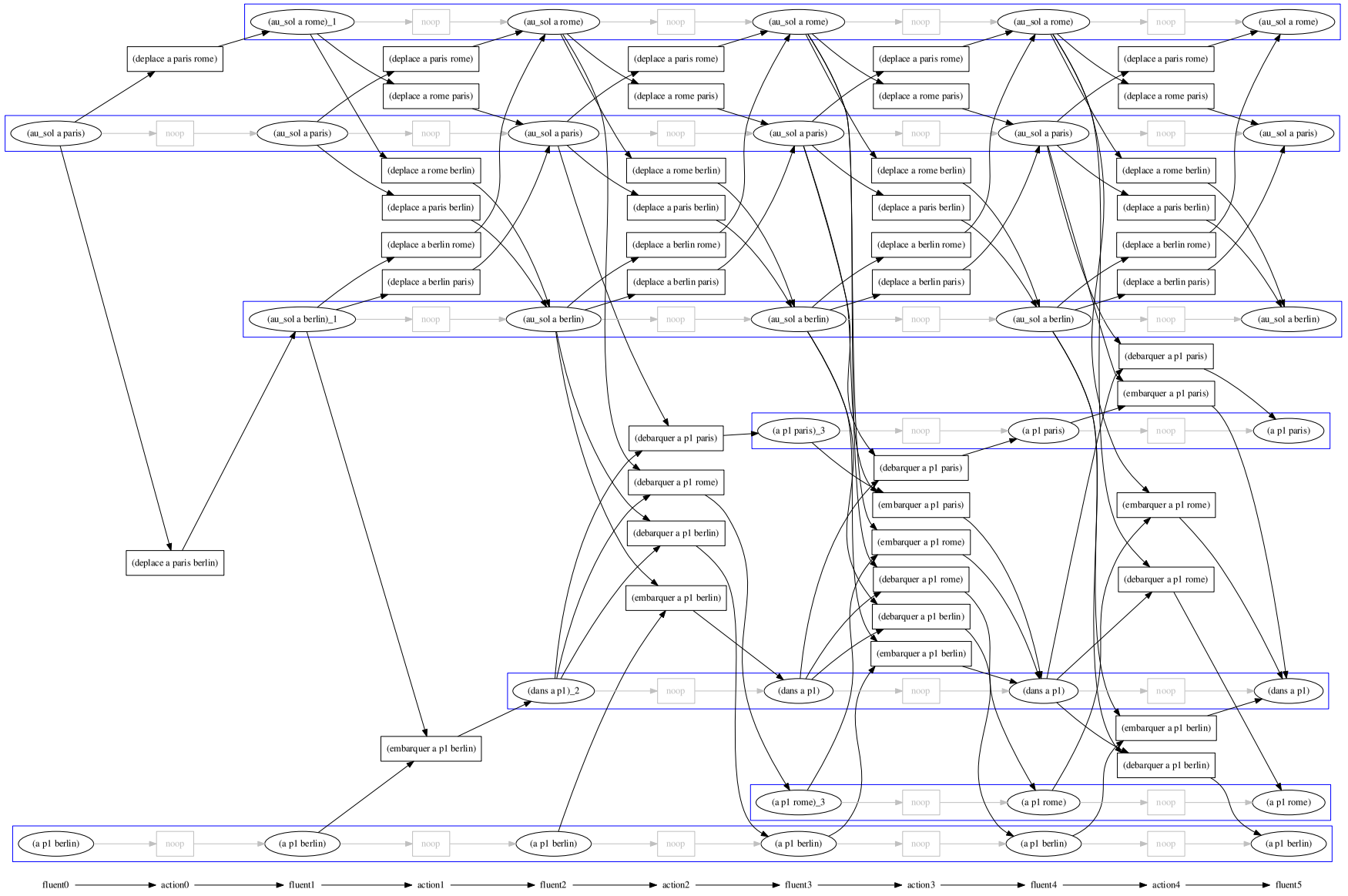


FIGURE 2.1 : Exemple d'un RPG pour un problème simple du domaine Zenotravel.

2.2 Heuristiques issues du graphe de planification

Les heuristiques sont des estimateurs capables d'évaluer une distance : le nombre d'actions nécessaires entre deux états. En pratique, on évalue la distance entre un état et le but du problème. Ces heuristiques essaient donc d'estimer la taille ou le coût d'un plan solution du problème de planification et, grâce aux algorithmes de recherche dans un graphe (détaillé dans la section 2.3), vont guider la recherche dans l'espace d'états. Ces heuristiques, habituellement définies par $h : 2^F \longrightarrow \mathbb{R}^+$ dépendent uniquement de l'état courant. Cependant, certaines heuristiques [13] sont de nature différente et dépendent du chemin suivi depuis l'état initial. Partant d'hypothèses différentes, elles n'ont donc pas les mêmes propriétés.

Graphplan utilise le graphe de planification pour faire une recherche en arrière dans cet espace. Des méthodes plus récentes se basent sur le cœur de Graphplan, le graphe de planification, pour calculer des heuristiques indépendantes du domaine. Plus précisément, ces heuristiques se basent sur le RPG [Bonet et Geffner, 2001].

2.2.1 Heuristique optimale $h^\star$

On peut tout d'abord présenter une heuristique théorique qui informerait parfaitement de la distance entre un état et le but du problème.

Définition 13 : *Heuristique optimale* $h^\star$

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On note $h^\star_\Pi(s)$*, l'heuristique donnant le coût optimal entre l'état* s *et le but* G*, définie comme :*

$$h^\star_\Pi : \left| \begin{array}{ccc} 2^F & \longrightarrow & \mathbb{R}^+ \\ s & \longmapsto & \begin{cases} \min\limits_{\rho \in \mathcal{P}^\star(\langle F,A,s,G \rangle)} |\rho| & \text{si } \mathcal{P}^\star(\langle F, A, s, G \rangle) \neq \emptyset \\ \infty & \text{sinon} \end{cases} \end{array} \right.$$

Lorsque le contexte n'est pas ambigu, $h^\star$ *dénotera* $h^\star_\Pi$*.*

À partir de l'heuristique $h^\star$, on peut définir une autre heuristique donnant la distance optimale pour le problème relaxé.

Définition 14 : *Heuristique optimale du problème relaxé* h^+

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On note h^+*, l'heuristique donnant le coût optimal entre l'état* s *et le but pour le* problème relaxé*, définie de la manière suivante :*

$$h^+_\Pi = h^\star_{\Pi^+}$$

Lorsque le contexte n'est pas ambigu, h^+ *dénotera* h^+_Π*.*

13. *Par exemple, l'heuristique* h^{LM} *est dépendante du chemin parcouru jusqu'à un nœud (cf. section 3.5).*

Le calcul exact de ces deux heuristiques est aussi difficile que la résolution exacte du problème de planification [14], mais elles vont cependant permettre de caractériser les autres heuristiques.

2.2.2 Admissibilité et consistance

Nous allons tout d'abord préciser quelques propriétés qui vont caractériser les heuristiques et qui permettront d'avoir certaines hypothèses lors de leur utilisation dans des algorithmes de recherche (section 2.3).

Définition 15 : *Admissibilité d'une heuristique*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
Une heuristique h *est dite* admissible *si et seulement si*

$$\left(\forall s \in 2^F\right) h(s) \leqslant h^\star(s)$$

C'est-à-dire, une heuristique admissible ne surestime jamais la distance par rapport au but.

Exemple 3 : h^+ *est admissible*

Par définition, on voit directement que l'heuristique h^+ *est admissible. En effet, le plan relaxé optimal est nécessairement plus court que le plan optimal puisqu'il n'a pas à prendre en compte le rétablissement des fluents.*

Une propriété plus forte que l'admissibilité est la consistance. Cette propriété garantit un comportement monotone de l'heuristique, c'est-à-dire que la différence entre les valeurs $h(n)$ de deux nœuds consécutifs est plus petite que le coût réel de l'arc entre ces deux nœuds. Un des effets de la monotonie est la décroissance de la valeur de l'estimation de l'heuristique le long du meilleur plan solution [15]. Enfin, la consistance implique l'admissibilité.

Définition 16 : *Consistance d'une heuristique*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
Une heuristique h *est dite* consistante *si et seulement si*

$$\left(\forall s \in 2^F\right) \left(\forall a \in A, \mathrm{pre}(a) \subseteq s\right) h(s) \leqslant \mathrm{cost}(a) + h(s[a])$$

14. Le calcul exact de h^+ a cependant été étudié et peut être calculé de façon efficace dans certains cas [Bonet et Geffner, 2008].

15. Décroissance de la fonction $f(s) = g(s) + h(s)$ où $g(s)$ est le coût du meilleur chemin de l'état initial vers l'état s le long du meilleur plan.

Exemple 4 : *Consistance de* h^+

De même, on peut voir que h^+ *est consistante, puisque (grossièrement) :*

$$(\forall s \in 2^F) \left(\forall a \in A, pre(a) \subseteq s\right)$$

Si a *est une action optimale pour atteindre le but* G *alors on a simplement*

$$h^+(s) = h^+(s[a]) + cost(a)$$

Sinon le nouveau plan est nécessairement plus long que le plan optimal, donc

$$h^+(s) < h^+(s[a]) + cost(a)$$

Donc, $h^+(s) \leqslant h^+(s[a]) + cost(a)$.

Les heuristiques sont fortement impactées par ces propriétés. En effet, elles permettent en particulier de garantir l'optimalité des solutions trouvées par des algorithmes de recherche dans des graphes, et même de réduire le nombre de nœuds explorés par ces algorithmes (cf. section 2.3.1.2). Si ces propriétés sont très importantes pour la résolution des problèmes de façon optimale, elles le sont moins pour la planification sous-optimale. En outre, ces propriétés ne garantissent en rien le fait qu'une heuristique soit informative[16]. Par exemple, l'heuristique $(\forall s \in 2^F)\ h(s) = 0$ est consistante donc admissible, mais non informative.

2.2.3 Heuristique basée sur la relaxation du problème : h^{add}

Une première heuristique basée sur le RPG est h^{add} [Bonet et Geffner, 2001]. Cette heuristique fait une hypothèse d'indépendance entre les fluents ; c'est-à-dire que le coût pour atteindre un ensemble de fluents est la somme des coûts pour atteindre chacun de ces fluents indépendamment. On suppose ainsi qu'il n'y aurait pas d'actions permettant l'obtention de plusieurs fluents utiles à la résolution du problème en même temps.

16. C'est-à-dire qu'elle permette une recherche efficace.

Définition 17 : *Heuristique* h^{add}

Soit $\Pi = \langle F, A, I, G \rangle$ un problème de planification.
On note l'heuristique h^{add}, définie récursivement comme :

$$h^{add} : \begin{array}{rcl} 2^F & \longrightarrow & \mathbb{R}^+ \\ s & \longmapsto & h^{add}(G, s) \end{array}$$

$$h^{add} : \begin{array}{rcl} 2^F \times 2^F & \longrightarrow & \mathbb{R}^+ \\ (E, s) & \longmapsto & \sum_{f \in E} h^{add}(f, s) \end{array}$$

$$h^{add} : \begin{array}{rcl} F \times 2^F & \longrightarrow & \mathbb{R}^+ \\ (f, s) & \longmapsto & \begin{cases} 0 & si\ f \in s \\ \min_{a \in A, f \in add(a)} h^{add}(a, s) & sinon \end{cases} \end{array}$$

$$h^{add} : \begin{array}{rcl} A \times 2^F & \longrightarrow & \mathbb{R}^+ \\ (a, s) & \longmapsto & cost(a) + h^{add}(pre(a), s) \end{array}$$

Cette heuristique calcule, pour un état s, $h^{add}(G, s)$:

- pour un ensemble de fluents F, la somme de h^{add} pour chacun des fluents de F ;
- pour un fluent f, le coût de l'action qui minimise (récursivement) h^{add} pour l'ensemble de ses préconditions (cette action est aussi appelé *best supporter*).

Il existe des méthodes polynomiales efficaces pour calculer h^{add}, en particulier on peut utiliser un algorithme de plus court chemin comme détaillé par l'algorithme 1.

Propriété 1 : *Non-admissibilité de* h^{add}

L'heuristique h^{add} n'est pas admissible.

Démonstration (contre-exemple) :

Soit $\Pi = \langle F, A, I, G \rangle$. Avec, $F = \{a, b\}$, $A = \{\langle \emptyset, \{a, b\}, \emptyset \rangle\}$, $I = \emptyset$ et $G = \{a, b\}$.
On a : $h^{add}(I) = h^{add}(\{a, b\}, \emptyset) = h^{add}(a, \emptyset) + h^{add}(b, \emptyset) = 2 \cdot cost(a)$
Or, on voit facilement que $h^\star(I) = cost(a)$ donc $h^{add}(I) > h^\star(I)$. ■

2.2.4 Heuristique h^{ff}

L'heuristique h^{ff} [Hoffmann et Nebel, 2001] estime la distance jusqu'au but en calculant un plan (non nécessairement optimal) pour Π^+, puis utilise la longueur de ce plan comme valeur heuristique. Le processus se passe en deux phases : dans un premier temps, le RPG est construit, puis, un plan relaxé ρ^+ est construit par

Algorithme 1: Calcul de h^{add}

Entrées : $\Pi = \langle F, A, I, G \rangle$ un problème de planification, s l'état courant
Sorties : $h^{add}(s) \in \mathbb{R}^+$ la distance évaluée de s jusqu'au but

pour chaque $f \in F$ **faire**
  $h'(f) \leftarrow \begin{cases} 0 & \text{si } f \in s \\ \infty & \text{sinon} \end{cases}$
répéter
  $h \leftarrow h'$
  pour chaque $f \in F$ **faire**
    $c \leftarrow \min_{a \in A, f \in add(a)} cost(a) + \sum_{f' \in pre(a)} h(f')$
    $h'(f) \leftarrow \min(h(f), c)$
jusqu'à $h = h'$
retourner $\sum_{f \in s} h(f)$

recherche arrière (de la même manière que Graphplan). En partant des fluents du but, des actions sont ajoutées au plan ρ^+ pour satisfaire ces buts. Il faut ensuite satisfaire les préconditions des actions ajoutées à ρ^+, et récursivement ajouter des actions jusqu'à remonter au niveau de fluents initiaux λ_0^f.

Différentes techniques ont été étudiées pour sélectionner les actions à ajouter à ρ^+ notamment privilégier les actions Noop [Hoffmann et Nebel, 2001] et les *best supporters* selon h^{max} [Keyder et Geffner, 2008].

2.2.5 Famille des heuristiques basées sur les chemins critiques : h^m

Contrairement aux deux heuristiques précédentes qui sont directement basées sur la relaxation des retraits des actions, la famille d'heuristiques h^m [Bonet et Geffner, 2001] ; [Haslum et Geffner, 2000] sont des heuristiques de chemins critiques, c'est-à-dire qu'elles essaient d'approximer le coût de réalisation du but au regard du fluent (ou de l'ensemble de fluents) le(s) plus coûteux à établir. Cette famille d'heuristiques a été créée dans le but d'avoir des heuristiques admissibles donc utilisables pour la planification optimale.

On peut définir h^m pour $m = 1$:

Définition 18 : *Heuristique* h^1/h^{max}

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On définit l'heuristique h^1 *de la même manière que* h^{add} *en remplaçant simplement la somme par un maximum pour calculer la valeur heuristique d'un ensemble de fluents, tel que :*

$$h^1 : \left| \begin{array}{rcl} 2^F \times 2^F & \longrightarrow & \mathbb{R}^+ \\ (F, s) & \longmapsto & \max\limits_{f \in F} h^1(f, s) \end{array} \right.$$

Propriété 2 : *Niveau d'appartion dans le RPG*

La valeur de h^1 *pour un fluent donne exactement son niveau d'apparition dans le graphe de planification relaxé.*

L'heuristique h^1 peut être étendue pour former la famille des heuristiques h^m qui calculent les coûts pour des paires (ou des n-uplets plus grands, de taille m). Ainsi :

Définition 19 : *Heuristique* h^m

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et* $m \in \mathbb{N}^*$.

$$h^m : \left| \begin{array}{rcl} 2^F & \longrightarrow & \mathbb{R}^+ \\ s & \longmapsto & h^m(G, s) \end{array} \right.$$

$$h^m : \left| \begin{array}{rcl} 2^F \times 2^F & \longrightarrow & \mathbb{R}^+ \\ (E, s) & \longmapsto & \begin{cases} 0 & s\ E \subseteq s \\ \min\limits_{(E', a) \in R(E)} (\text{cost}(a) + h^m(E', s)) & si\ |E| \leqslant m \\ \max\limits_{E' \in 2^E, |E'| \leqslant m} h^m(E', s) & sinon \end{cases} \end{array} \right.$$

$$R : \left| \begin{array}{rcl} 2^F & \longrightarrow & 2^F \times A \\ E & \longmapsto & \left\{ \left((E \setminus \text{add}(a)) \cup \text{pre}(a), a \right) \;\middle|\; a \in A\ et \right. \\ & & \left. \text{add}(a) \cap E \neq \emptyset\ et\ \text{del}(a) \cap E = \emptyset \right\} \end{array} \right.$$

Propriété 3 : *Admissibilité de* h^m

Les heuristiques de la famille h^m *sont admissibles [*Haslum *et* Geffner*, 2000].*

Propriété 4 : *Niveau d'apparition dans le graphe de planification*

La valeur de h^2 *pour un fluent donne exactement son niveau d'apparition dans le graphe de planification.*

Intuitivement, on peut dire que h^1 estime la distance d'un état à un ensemble de fluents via son fluent le plus coûteux. h^2 estime le coût de la paire de fluents

la plus coûteuse. Et ainsi de suite, jusqu'à atteindre un m suffisant grand pour devenir égal à $h^\star$ [17].

Avec l'augmentation de m, les heuristiques de la famille h^m sont de plus en plus informatives, mais deviennent cependant de plus en plus difficiles à calculer [18]. En pratique, on dépasse rarement h^2 car le temps de calcul de l'heuristique devient trop important et ne réduit plus le temps global de recherche dans le graphe [Haslum, 2006]. De plus, on peut noter que l'heuristique h^2 calcule exactement le niveau d'apparition des fluents dans le graphe de planification avec coûts unitaires. Enfin, même si ces heuristiques sont admissibles, h^1 n'est que très peu informative. En effet, si on prend un petit exemple où le but du problème est une conjonction de trois fluents productibles uniquement par trois actions complètement indépendantes, alors on aura $h^1 = 1$ alors que $h^{add} = 3$. En somme, si les buts d'un problème sont indépendants alors h^{add} sera informatif et proche de h^+ alors que h^1 sous-estimera énormément la distance par rapport au but.

2.2.6 Caractérisation et limitations de h^+

Grâce à une analyse des caractéristiques des domaines, comme la présence d'impasses (*dead-ends*) ou de plateaux, l'heuristique h^+ a pu être caractérisée [Hoffmann, 2005] ; [Hoffmann, 2011]. Comme les planificateurs modernes utilisent des heuristiques basées sur la relaxation Π^+, ce travail permet de mieux cerner leurs performances. De plus, certains domaines où, d'une part, la recherche grâce à des heuristiques basées sur la relaxation Π^+ sera « facile », et d'autre part, où ces heuristiques ne sont plus capables de repérer des impasses ont pu être déterminés. En particulier, l'absence de minimum local de la fonction h^+ et lorsque l'évasion des plateaux peut se faire grâce à un nombre d'actions borné par une constante implique qu'un plan solution peut être trouvé en temps polynomial en utilisant une méthode de recherche locale et h^+.

La figure 2.2 relate les résultats de cette analyse en classant un ensemble de domaines selon ces critères d'impasse et de sortie de plateaux. Comme on peut le voir, pour notre domaine exemple Zenotravel [19], h^+ a une bonne topologie. Ces analyses ont été automatisées [Hoffmann, 2011] mais encore peu ou pas utilisées directement dans un planificateur pour l'aider dans sa recherche heuristique.

17. Il faut que m soit suffisamment grand pour que le dernier cas de la définition de h^m, $|E| > m$, ne soit jamais activé. Il s'agit en fait du cas de relaxation pour cette famille d'heuristiques.
18. Calculable en $\mathcal{O}(|F|^m)$
19. De plus, notre version de Zenotravel est une version simplifiée, ne prenant pas en compte le carburant. Elle serait donc classée dans la colonne *undirected*, où les fluents peuvent être achevés immédiatement.

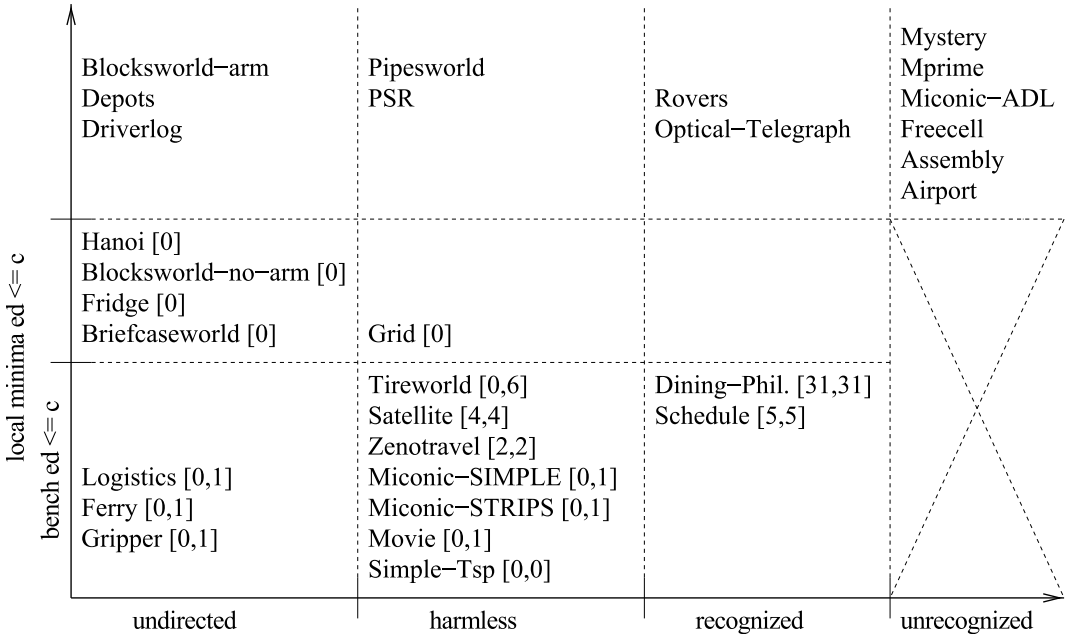


FIGURE 2.2 : Taxinomie des problèmes de planification. *Undirected* et *Harmless* sont des domaines sans impasses. *Recognized* sont les domaines avec impasses mais où Π^+ ne perd pas cette spécificité. *Unrecognized* sont les domaines avec impasses où Π^+ n'a pas d'impasses. Figure extraite de [HOFFMANN, 2005].

2.2.7 D'autres manières de relaxer le problème de planification

2.2.7.1 Compilation des conjonctions dans le problème de planificaiton

Afin de pallier les problèmes liés aux heuristiques basées sur le problème relaxé, comme le non-comptage des allers-retours dans les problèmes de transports, certains auteurs s'intéressent à ajouter des informations de retraits de fluents dans le problème relaxé.

La compilation Π^m transforme le problème original Π, en y rajoutant des fluents représentant les conjonctions de fluents de taille au plus m et en modifiant les actions pour prendre en compte ces fluents-conjonction [HASLUM, 2009]. Cette compilation permet de calculer h^m comme $h^{max}(\Pi^m)$.

S'inspirant de cette compilation, KEYDER, HOFFMANN et HASLUM [2012] ont défini la compilation Π^C_{ce}, qui permet de s'affranchir de l'intractabilité du problème Π^m grâce d'une part à l'utilisation des effets conditionnels et d'autre part d'une sélection de conjonctions de fluents $C \in 2^F$. Le problème Π^C_{ce} possède autant d'actions que Π avec un nombre d'effets conditionnels linéaire en fonction de $|C|$. De plus, contrairement à $h^+(\Pi^m)$, $h^+(\Pi^C_{ce})$ est consistante et de plus domine $h^+(\Pi)$.

2.2.7.2 Algorithme de planification rouge-noir

Une autre manière d'intégrer l'information de retrait des actions dans le problème relaxé peut aussi se faire en tenant compte des retrait de variables seulement pour certains variables (dans le modèle SAS$^+$). Les variables rouges sont les variables où les retraits sont relaxés. Il est possible de trouver un plan pour un problème de ce type de manière efficace si l'ensemble des variables noires (non relaxées) sont réversibles [Katz, Hoffmann et Domshlak, 2013] ; [Katz et Hoffmann, 2013].

2.3 Algorithme de recherche dans un graphe

Les méthodes de recherche dans les espaces d'états, basées sur des méthodes de graphe de recherche en avant, font partie d'un des axes de recherche les plus étudiés en planification classique (en particulier grâce à l'étude d'heuristiques basées sur l'analyse du problème). Les algorithmes de recherche *best-first*, *Enforced Hill-Climbing*, et *weighted*-A* sont des méthodes privilégiées en planification sous-optimale. En planification optimale, l'algorithme A* est favorisé. Ces algorithmes vont utiliser les heuristiques de façon à sélectionner les zones intéressantes du graphe pour les algorithmes gloutons, ou alors pour élaguer des zones de recherche. Les nœuds du graphe d'exploration sont une combinaison d'un état de planification et de l'action qui a conduit à cet état depuis l'état initial. Très souvent, les algorithmes n'ont qu'un nœud pour chaque état du problème de planification, nous utiliseront donc les termes nœud et état indistinctement.

2.3.1 Algorithmes de recherche *Best-First*

Les algorithmes de type recherche *best-first* [Pearl, 1984] parcourent un graphe en explorant le nœud qui semble être le plus prometteur (meilleur d'abord). Pour évaluer la qualité d'un nœud ou plus précisément sa capacité à aider la recherche du but, ces algorithmes utilisent des heuristiques vues comme des fonctions d'évaluation de cette qualité (cf. section 2.2).

Ces algorithmes ont une base commune et se différencient au niveau de la sélection du prochain nœud à explorer. En partant de l'état initial du problème, ces algorithmes vont donc itérativement sélectionner le prochain nœud (état) à explorer dans la liste des nœuds atteints mais encore non développés (liste ouverte). Une fois le nœud choisi, il est ajouté dans la liste fermée et les états voisins (un pour chaque action applicable) sont ajoutés dans la liste ouverte (s'ils ne sont pas déjà dans la liste ouverte ou la liste fermée). L'algorithme 2 en présente une version détaillée.

Un des points-clés de cet algorithme est la sélection du prochain nœud à explorer (ligne 4). Cette ligne sélectionne le nœud ayant la plus petite valeur f calculée par la fonction *eval*. Cette fonction combine la distance *réelle* du nœud

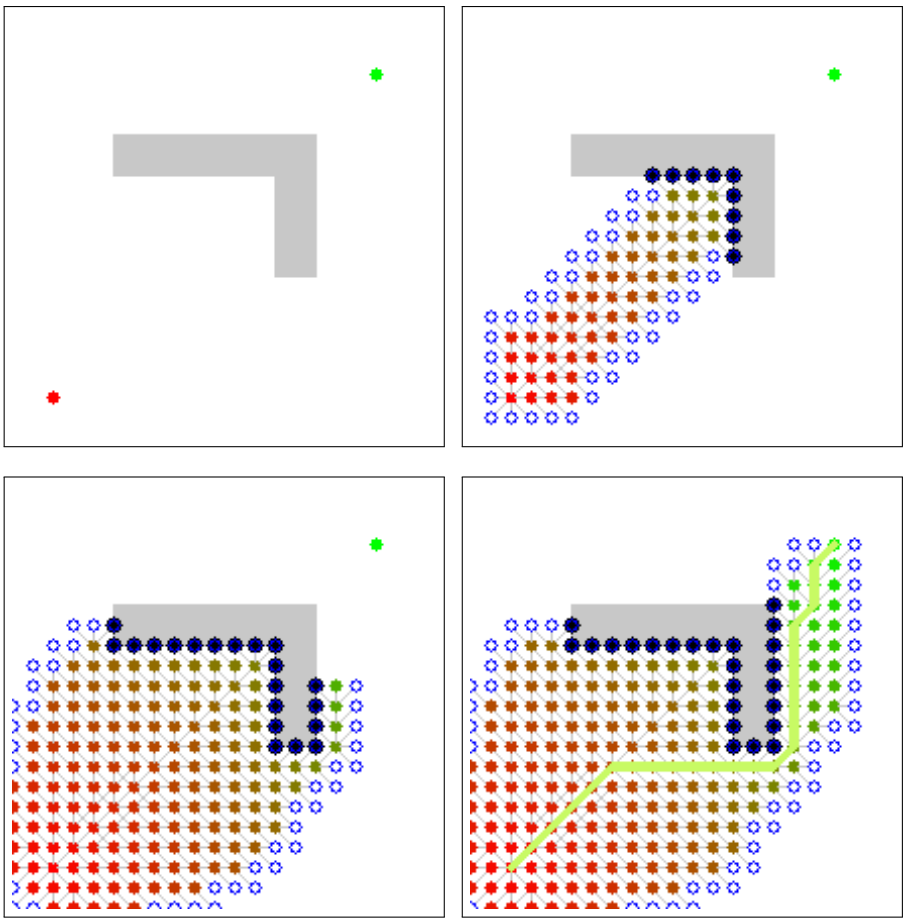


FIGURE 2.3 : Évolution d'un algorithme de recherche *best-first*. Les nœuds dans la liste fermée sont symbolisés par un cercle plein (allant du rouge vers le vert en fonction la valeur heuristique). Les cercles bleus sont les nœuds se trouvant dans la liste ouverte. Enfin, on peut voir sur la dernière image en vert, le chemin trouvé. (CC-BY 3.0 Subh83)

Algorithme 2: Recherche *Best-First* appliquée aux problèmes de planification

Entrées : $\Pi = \langle F, A, I, G \rangle$ un problème de planification,
eval(état, but, coût) une fonction d'évaluation
Sorties : Un plan solution $\rho \in \mathcal{P}^\star$ ou $\perp$

```
/* Un nœud du graphe: ⟨état,évaluation,plan partiel⟩                    */
/* Initialisation de la liste ouverte 𝒪 avec l'état initial, et
   de la liste fermée 𝒞 à vide                                          */
```

1 $\mathcal{O} \leftarrow \{\langle I, 0, \text{eval}(I, G, 0), \emptyset\rangle\}$
2 $\mathcal{C} \leftarrow \emptyset$
3 **tant que** $\mathcal{O} \neq \emptyset$ **faire**
4 $\quad \langle s, f, \rho\rangle \leftarrow \underset{\langle s', f', \rho'\rangle \in \mathcal{O}}{\text{argmin}}\ f'$ `/* Sélection du prochain nœud à développer */`
5 $\quad \mathcal{O} \leftarrow \mathcal{O} \setminus \{\langle s, f, \rho\rangle\}$
6 $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\langle s, f, \rho\rangle\}$
7 $\quad$ **si** $G \subseteq s$ **alors**
8 $\quad\quad$ **retourner** ρ
9 $\quad$ **pour chaque** $a \in A$ **tel que** $\text{pre}(a) \subseteq s$ **faire**
10 $\quad\quad s' \leftarrow s[a]$
11 $\quad\quad$ **si** $s' \notin \mathcal{C} \cup \mathcal{O}$ **alors**
12 $\quad\quad\quad \rho' \leftarrow \rho \circ \langle a\rangle$
13 $\quad\quad\quad f' \leftarrow \text{eval}(s', G, |\rho'|)$
14 $\quad\quad\quad \mathcal{O} \leftarrow \mathcal{O} \cup \{\langle s', f', \rho'\rangle\}$
15 $\quad\quad$ **sinon**
16 $\quad\quad\quad$ MiseAJourNoeud$(s', \langle s, f, \rho\rangle)$
17 **retourner** $\perp$

depuis l'état initial et la distance *évaluée* (heuristiquement) jusqu'au but. Cette fonction va conditionner la vitesse de la recherche et l'optimalité de la première solution trouvée. Parmi les nœuds qui ont une évaluation égale, les algorithmes préféreront les états avec une valeur heuristique plus faible.

La fonction MiseAJourNoeud met à jour les distances et les parents du nœud fils que l'on vient d'atteindre s'il se trouve déjà dans la liste ouverte ou la liste fermée et si le coût depuis l'état initial est inférieur à la première insertion de ce nœud dans la liste ouverte. En fonction des propriétés des fonctions heuristiques et d'évaluation utilisées, et des conditions d'optimalité recherchées, il faudra éventuellement sortir ce nœud de la liste fermée pour le réintroduire dans la liste ouverte.

Pour détailler les fonctions d'évaluation, on notera *coût*, le coût du chemin entre l'état initial I et l'état courant et *h*, la valeur heuristique entre l'état courant et le but du problème.

2.3.1.1 Algorithme *Best-First* glouton

Dans le cas de l'algorithme *Best-First* glouton, on définit la fonction d'évaluation comme étant exactement la valeur retournée par la fonction heuristique :

$$eval(\text{état}, G, \text{coût}) = h(\text{état}, G)$$

On obtient donc un algorithme qui va tenter de s'approcher le plus rapidement possible d'un état but au détriment de la qualité du chemin choisi. De plus, une fois qu'un nœud est développé et ajouté à la liste fermée, il ne sera plus jamais reconsidéré. Cet algorithme est complet puisqu'il épuisera complètement son espace de recherche (tous les nœuds du graphe seront explorés). Cependant, on ne peut avoir aucune garantie sur l'optimalité de la solution trouvée.

2.3.1.2 Algorithme $A^\star$ [Hart, Nilsson et Raphael, 1968]

Contrairement à la recherche gloutonne, l'algorithme $A^\star$ propose de minimiser la somme de la distance par rapport à l'état initial plus l'évaluation du nœud par rapport au but :

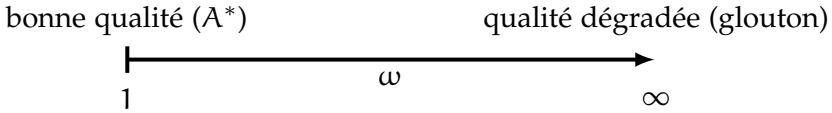
$$eval(\text{état}, G, \text{coût}) = \text{coût} + h(\text{état}, G)$$

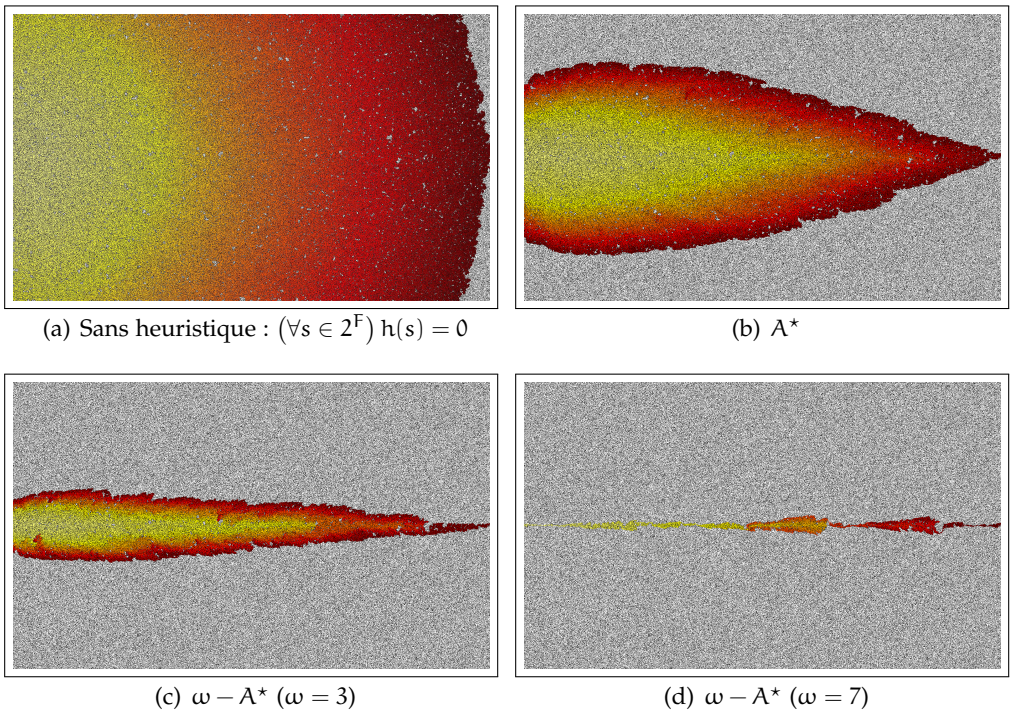
L'algorithme $A^\star$ combiné à une heuristique consistante (définition 16) donne la garantie qu'un nœud développé (c'est-à-dire, ajouté à la liste fermée) une fois n'aura pas besoin d'être reconsidéré. Si l'heuristique est seulement admissible, il est quand même possible de préserver l'optimalité de la solution trouvée, mais pour cela, il faudra réouvrir les nœuds atteints plusieurs fois avec des coûts inférieurs [Dechter et Pearl, 1985]. C'est-à-dire, les sortir de la liste fermée et les réinsérer dans la liste ouverte avec les nouvelles valeurs d'évaluation.

L'algorithme $A^\star$ peut être modifié vers une version plus gloutonne présentant un compromis sur la vitesse de convergence vers une solution au détriment de sa qualité, en appliquant un poids à la fonction d'évaluation :

$$eval(\text{état}, G, \text{coût}) = \text{coût} + \omega * h(\text{état}, G) \text{ avec } \omega \geqslant 1$$

Cet algorithme est appelé *weighted*-$A^\star$ [Pohl, 1970] (ou $\omega - A^\star$) et plus on augmente ω, plus la recherche devient gloutonne :





(a) Sans heuristique : $(\forall s \in 2^F)\ h(s) = 0$ (b) $A^\star$

(c) $\omega - A^\star$ $(\omega = 3)$ (d) $\omega - A^\star$ $(\omega = 7)$

FIGURE 2.4 : Exploration de différents algorithmes de recherche de type *best-first* sur une grille. Les nœuds vont du jaune vers le rouge en fonction de l'ordre dans lequel ils ont été développés. Les murs sont symbolisés en noir. (CC-BY 3.0 Jordan Thayer)

2.3.2 Enforced Hill-Climbing

Contrairement aux algorithmes de type *Best-First* qui sont des algorithmes de recherche d'optimum global, l'algorithme *Hill-Climbing* est un algorithme glouton de recherche locale (ou de recherche d'optimums locaux). Cet algorithme sélectionne le meilleur nœud successeur ayant (strictement) une meilleure évaluation heuristique que le nœud courant. *Enforced Hill-Climbing* [HOFFMANN et NEBEL, 2001] adapte ce premier algorithme dans le cas où un nœud améliorant l'heuristique n'est pas dans le voisinage immédiat du nœud courant. Dans ce cas, l'algorithme recherche ce nouveau nœud grâce à une recherche en largeur d'abord. Cet algorithme est complet seulement si l'espace de recherche ne comporte pas d'impasses, c'est-à-dire si de n'importe quel état du problème de planification, il est toujours possible d'atteindre le but.

2.4 Autres techniques améliorant les algorithmes de recherche

Comme nous le détaillerons plus amplement dans le chapitre 4 sur les planificateurs, ceux-ci ne se limitent pas à des algorithmes de recherche dans un graphe classique, mais y incorporent des techniques avancées spécifiques à la planification classique. Ces techniques s'intéressent par exemple au problème de *tie-breaking* [20], à la combinaison d'heuristiques ou encore à la définition de macro-actions.

2.4.1 Évaluation heuristique différée *(deferred heuristic evaluation)*

L'évaluation heuristique différée [Helmert, 2006] ; [Richter et Helmert, 2009] est une variante de recherche visant à réduire le nombre d'évaluation des nœuds. Dans un algorithme de recherche *Best-First* classique, la liste ouverte garde ordonné un ensemble d'états à développer plus tard, dans l'ordre de leur valeur heuristique. Cela implique qu'à chaque fois qu'un état est ajouté dans la liste ouverte il faut calculer sa valeur heuristique. Or, même si les complexités de calcul des heuristiques sont polynomiales, les temps de calculs ne sont pas négligeable face au temps de calcul global et peut réduire significativement le nombre de nœuds développés par seconde. En particulier, lorsque qu'un état a beaucoup de successeurs.

Au lieu de calculer l'heuristique au moment de l'ajout dans la liste ouverte, on utilise plutôt la valeur heuristique du parent. De cette manière, tous les fils d'un nœud partagent la même valeur heuristique (et par conséquent on ne calcule la valeur qu'une fois). Ainsi, la valeur heuristique d'un nœud est calculée uniquement si ce nœud est développé.

Le gain potentiel n'est pas nécessairement évident à percevoir ; cependant, cette technique peut réduire le nombre de nœuds explorés lorsqu'un bon candidat est rapidement sélectionné parmi les fils d'un nœud grâce à d'autres techniques (comme les *preferred operators*) [Helmert, 2006].

Les performances du planificateur LAMA [Richter et Westphal, 2010], et l'ensemble des expériences faites lors de la conception de ce planificateur [Richter, 2010] montrent que l'évaluation différée permet de faire un compromis entre temps de recherche et qualité de la solution. Moins de temps est passé à calculer des heuristiques, mais les valeurs heuristiques sont moins informatives pour chaque nœud de la recherche.

2.4.2 *Preferred operators* et *helpful actions*

Les *preferred operators* sont un sous-ensemble des actions applicables pour un état donné. L'idée est de donner priorité à un ensemble d'actions qui semblent

20. Lorsque plusieurs nœuds de la liste ouverte ont la même évaluation heuristique (les plateaux)

plus utiles (dans le sens où elles rapprocheront du but du problème) que d'autres actions qui sont applicables, mais potentiellement inutiles. On peut par exemple imaginer qu'une fois avoir déplacé un avion d'une destination à une autre, il est probablement inutile de faire le trajet retour directement sans avoir effectué d'autres actions entre-temps (embarquer des passagers, etc.).

Dans le planificateur FF [Hoffmann et Nebel, 2001], les *preferred operators* (appelés *helpful actions* dans ce cas-là) correspondent aux actions applicables dans l'état courant générant au moins un fluent produit par une action du premier niveau du RPG se trouvant aussi dans le plan relaxé trouvé lors du calcul de l'heuristique h^{ff} (cf. section 2.2.4). Le planificateur FF utilise uniquement les *helpful actions* pour se déplacer dans l'espace d'états. L'intention est de couper une partie de l'espace de recherche, et ne s'intéresser qu'aux actions jugées utiles. Les actions qui contribuent à la résolution du problème relaxé aideront a priori aussi à résoudre le problème original.

Fast-Downward [Helmert, 2006] reprend ce concept de sélection d'un sous-ensemble d'états successeurs qui semblent prometteurs, en se basant sur l'heuristique des graphes causaux h^{CG} [Helmert, 2004]. Pour cela, en utilisant la modélisation SAS^+, on peut calculer des graphes de transition des variables. Il suffit ensuite de sélectionner les actions qui mènent à une solution en ne tenant compte que d'une seule variable (en fait, trouver un chemin solution dans le graphe de transition d'une variable).

Les *preferred operators* peuvent être utilisés pour classer les nœuds donné ex-aequo par l'heuristique utilisée [Richter, 2010]. À l'opposé, on peut choisir de développer d'abord les états provenant des *preferred operators* [Vidal, 2004].

Ces méthodes ont été largement étudiées et évaluées, en particulier lors de la conception du planificateur LAMA [Richter, 2010]. Elles ont la plupart du temps permis d'améliorer les performances des planificateurs qui les ont utilisées.

D'autres travaux [Wehrle, Kupferschmid et Podelski, 2008] se sont aussi intéressés à analyser le contexte pour trouver les actions qui sont inutiles à la résolution du problème, et ainsi ne plus les considérer lors de la recherche (et donc couper des parties de l'espace de recherche).

2.4.3 Listes ouvertes multiples (*Multi-queue*)

Pour pouvoir utiliser plusieurs heuristiques au sein d'un même algorithme de recherche, [Helmert, 2006] propose une extension des algorithmes de type *Best-First* en utilisant plusieurs listes ouvertes, une pour chaque heuristique utilisée. L'intention est de pouvoir contrebalancer les faiblesses d'une heuristique grâce au support d'autres heuristiques, sans cependant utiliser de technique pour combiner ces heuristiques en une unique fonction.

2.4.4 *Lookaheads*

Avec l'intention de faire des explorations profondes très rapides du graphe de recherche, les *lookahead* [Vidal, 2004] ; [Vidal, 2011] appliquent itérativement les actions du plan relaxé de l'état courant. Cette application gloutonne des plans relaxés considère que ces plans sont souvent de bonne qualité au regard du problème original et peuvent permettre de se rapprocher rapidement du but du problème.

Les sondes (*probes*) [Lipovetzky et Geffner, 2011] ; [Lipovetzky, 2012] sont des séquences d'actions. Pour construire sa séquence d'action, une sonde va construire à la volée une séquence de sous-buts (les landmarks, que nous définirons au chapitre suivant). Le prochain sous-but à atteindre est choisi parmi l'ensemble des landmarks qu'il reste à atteindre. Les actions remplissant les séquences entre les sous-buts sont sélectionnées grâce aux *helpful actions* et l'heuristique h^{add} biaisée. Le biais introduit dans h^{add} pénalise les actions menaçant les engagements pris lors de la sélection de précédentes actions de la séquence. Un engagement correspond au fait qu'une action a été ajouté à la séquence avec pour but indirect d'ajouter certains fluents du plan relaxé.

Ces deux méthodes ont été conçues de manière a être capable de résoudre des problèmes simples. Utilisées en combinaison avec un algorithme de recherche en avant en l'appliquant à chaque développent de nœud, elles permettent d'aller rapidement explorer l'espace d'états en profondeur.

3

LANDMARKS

Un landmark est un élément d'un problème de planification qui doit être obtenu à un moment ou à un autre lors de l'exécution de n'importe quel plan solution. Si on reprend l'exemple de la figure 1.1, on peut voir que les passagers doivent se trouver dans l'avion à un moment donné pour que le but puisse être atteint. Donc le fluent $(\mathtt{dans}\ \mathtt{p}_1\ \mathtt{a})$ est bien un landmark. Dans ce chapitre, nous introduisons les différentes notions associées aux landmarks, et les différentes manières de les utiliser.

Sommaire

3.1 Différents types de landmarks

On peut donner deux définitions sur les landmarks :

Définition 20 : *Landmark factuel* [Porteous, Sebastia *et* Hoffmann, *2001*]

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On dit qu'un fluent $l \in F$ *est un* landmark *factuel pour* Π *si et seulement s'il respecte une de ces deux conditions :*

$$\begin{cases} l \in I \\ (\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l \in \mathtt{add}(a). \end{cases}$$

On notera $\mathcal{L}_f(\Pi)$ *(ou* $\mathcal{L}$ *quand non ambigu), l'ensemble des landmarks du problème* Π*.*

Définition 21 : *Landmark causal* [Zhu *et* Givan, *2003*]

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On dit que qu'un fluent $l \in F$ *est un* landmark causal *pour* Π *si et seulement s'il respecte une de ces deux conditions :*

$$\begin{cases} l \in G \\ (\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l \in \mathtt{pre}(a) \end{cases}$$

On notera $\mathcal{L}_c(\Pi)$ *(ou* $\mathcal{L}$ *quand non ambigu), l'ensemble des landmarks causaux du problème* Π*.*

Ces deux définitions de landmarks ont une subtile différence. En effet, les landmarks causaux ne donnent que des landmarks utiles pour l'obtention du but tandis que les landmarks factuels rajoutent en plus quelques landmarks rencontrés lors de l'exécution de n'importe quel plan solution mais ne servant pas directement à la production du but. Par exemple, pour un problème simple avec un état initial vide $I = \emptyset$, un but $G = \{\alpha\}$ et une seule action sans précondition et ajoutant deux fluents $\{\alpha, \beta\}$, on obtient deux landmarks factuels $\{\alpha, \beta\}$ mais un seul landmark causal $\{\alpha\}$. De plus, les landmarks non causaux ne sont pas très informatifs, et peuvent même être nuisibles dans certains cas. On peut par exemple imaginer un problème où un agent doit marcher sous la pluie pour atteindre sa destination. Le fluent $(\mathtt{agent\_mouillé})$ serait donc un landmark non-causal (ajouté par une action mais dans aucune précondition menant au but). Utiliser ce fluent comme sous-but serait trompeur pour essayer de se rapprocher du but (il ne nous indique pas la direction à suivre pour aller vers la destination voulue). Ces landmarks supplémentaires peuvent être vus comme des effets de bord dûs à la résolution du problème : une ressource n'est peut-être pas directement utile à la résolution du problème, mais sera quand même altérée dans tous les cas. Cependant, même si ces landmarks semblent néfastes si on les utilise directement comme sous-buts,

ils peuvent, dans certains cas, être bénéfiques lorsque qu'ils sont utilisés pour du calcul d'heuristiques [Richter, Helmert et Westphal, 2008].

Théorème 2 : ***Relation entre landmark factuel et landmark causal***

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*

$$\mathcal{L}_c(\Pi) \subseteq \mathcal{L}_f(\Pi)$$

Démonstration :

Soit $\Pi = \langle F, A, I, G \rangle$ un problème de planification.
Si $l \in \mathcal{L}_c(\Pi)$, alors (par définition 21) on a $(\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l \in pre(a)$, ou $l \in G$.
Dans les deux cas, on peut avoir directement l appartient à l'état initial. En effet, si l'action a est la première action du plan alors $pre(a) \subseteq I$. Et, si $l \in G$ mais n'a été ajouté par aucune action alors il appartient directement à l'état initial. Donc, $l \in I$ **(1)**
Sinon l a été ajouté par une action des plans solutions. En effet, on sait que :

$$(\forall \rho = \langle a_1, \dots, a_n \rangle \in \mathcal{P}^\star)(\exists i \in [\![2, n]\!])\ l \in pre(a_i)$$

Donc, comme $l \notin I$,

$$(\exists i' \in [\![1, i-1]\!])l \in I[\rho_{1 \to i'}] \text{ et } l \notin I[\rho_{1 \to (i'-1)}]$$

C'est-à-dire $l \in add(a_{i'})$
Donc $(\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho) l \in add(a)$ **(2)**
Donc, d'après **(1)** et **(2)**, on a bien $l \in \mathcal{L}_f(\Pi)$
Donc $\mathcal{L}_c(\Pi) \subseteq \mathcal{L}_f(\Pi)$ ■

3.2 Ordres sur les landmarks

À partir d'un ensemble de landmarks, on peut essayer de caractériser quels landmarks doivent être obtenus avant d'autres, dans tous les plans solutions. Il s'agit de trouver quels sont les landmarks nécessaires à l'établissement d'autres landmarks, permettant de définir des relations d'ordre entre les landmarks. Une première relation naturelle est la suivante :

Définition 22 : ***Relation d'ordre strict partiel $\to$ (ordre naturel)***

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification.*
On note $\to$*, la relation d'ordre strict partiel* naturelle *sur un ensemble de landmarks* $\mathcal{L}$*.*
$(\forall (l, l') \in \mathcal{L}^2)$*, on a* $l \to l'$ *si et seulement si* l *doit être obtenu avant* l' *dans tous les plans solutions. C'est-à-dire*

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l' \in I[\rho_{1 \to j}] \implies (\exists i < j)\ l \in I[\rho_{1 \to i}]$$

À partir d'un ensemble de landmarks et d'un ordre (ou un ensemble d'ordres) sur cet ensemble, on peut créer un graphe représentant cette relation. Le graphe de landmarks est exactement le *diagramme de Hasse* représentant l'ensemble $\mathcal{L}$ partiellement ordonné par $\rightarrow$:

Définition 23 : *Graphe de landmarks* Γ

Soient $\mathcal{L}$, un ensemble de landmarks et $\rightarrow$, une relation d'ordre strict partiel associée à $\mathcal{L}$.
On dit que $\Gamma = \langle \mathcal{V}, \mathcal{E} \rangle$ est le graphe orienté de landmark associé à $(\mathcal{L}, \rightarrow)$ où :

- *$\mathcal{V} = \mathcal{L}$ est l'ensemble des nœuds ;*
- *l'ensemble des arcs $\mathcal{E}$ provient de la réduction transitive du graphe $\langle \mathcal{L}, \{(l, l') \in \mathcal{L}^2 \mid l \rightarrow l'\} \rangle$.*

On notera abusivement $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$, le graphe de landmarks, construit de la manière décrite ci-dessus.
Soit $E \subseteq \mathcal{L}$, on note $\Gamma \setminus E = \langle \mathcal{L} \setminus E, \rightarrow \rangle$, le graphe de landmarks Γ privé du sous-ensemble de landmarks E.

Définition 24 : *Landmarks racines d'un graphe de landmarks* Γ

Soit $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$ un graphe de landmarks.
On dit que les landmarks racines de Γ sont les landmarks sans arc incident. C'est-à-dire, tous les landmarks qui n'ont pas de landmark à satisfaire avant de pouvoir être obtenus. On note cet ensemble $\mathcal{R}(\Gamma)$.

$$\mathcal{R}(\langle \mathcal{L}, \rightarrow \rangle) = \left\{ l \in \mathcal{L} \;\middle|\; (\nexists\, l' \in \mathcal{L})\; l' \rightarrow l \right\}$$

Définition 25 : *Parents et fils d'un landmark*

Soient $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$ un graphe de landmarks et $l \in \mathcal{L}$ un landmark.
On dit que les landmarks parents du landmark l sont l'ensemble des prédécesseurs du nœud l dans la fermeture transitive du graphe Γ, défini tel que :

$$\mathcal{P} : \left| \begin{array}{ccc} \mathcal{L} & \longrightarrow & 2^{\mathcal{L}} \\ l & \longmapsto & \left\{ l' \in \mathcal{L} \;\middle|\; l' \rightarrow l \right\} \end{array} \right.$$

On dit que les landmarks fils du landmark l sont l'ensemble des successeurs directs du nœud l dans le graphe Γ, défini tel que :

$$\mathrm{Ch} : \left| \begin{array}{ccc} \mathcal{L} & \longrightarrow & 2^{\mathcal{L}} \\ l & \longmapsto & \left\{ l' \in \mathcal{L} \;\middle|\; (l, l') \in \mathcal{E} \right\} \end{array} \right.$$

où $\mathcal{E}$ est l'ensemble des arcs du graphe Γ (qui par définition est réduit transitivement).

Il existe d'autres ordres entre les landmarks, plus ou moins restrictifs que la relation d'ordre naturelle. En particulier, certains de ces ordres ne respectent pas

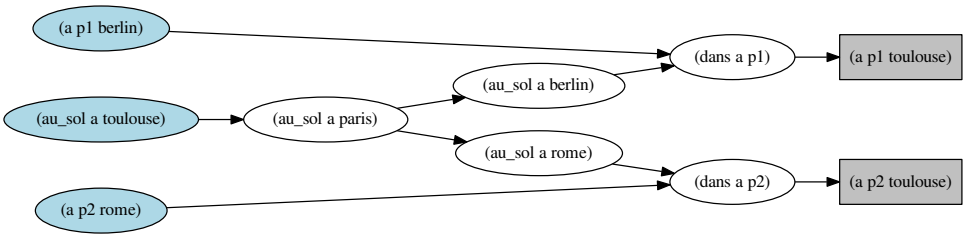


FIGURE 3.1 : Graphe de landmarks de l'exemple Zenotravel (en bleu les fluents de l'état initial, et en gris ceux du but)

nécessairement le fait qu'un landmark est obtenu avant un autre dans tous les plans solutions, et de ce fait ne seront pas considérées comme correctes [21]. Dans ces travaux, nous nous sommes uniquement intéressés aux relations correctes, en particulier parce que le graphe de landmarks est *acyclique* si la relation d'ordre partiel associé est *correcte*.

Définition 26 : *Relation d'ordre correcte entre landmarks*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\mathcal{L}$ *un ensemble de landmarks. On dit que* $\rightarrow_x$ *est une relation d'ordre* correcte *si et seulement si :*

$$(\forall (l, l') \in \mathcal{L}^2) l \rightarrow_x l' \implies l \rightarrow l'$$

Exemple 5 : *Example de graphe de landmarks pour Zenotravel*

La figure 3.1 nous montre un graphe de landmarks causaux généré à partir du problème exemple Zenotravel. On voit bien ici les points-clés nécessaires à la résolution du problème : l'avion doit aller à Paris avant d'aller à Rome et Berlin. Les passagers doivent nécessairement se trouver dans l'avion avant de pouvoir se trouver à Toulouse. L'exécution de tous les plans solutions passe par ces points.

D'autres définitions d'ordres, toujours corrects, avec des propriétés plus fortes sur la relation entre landmarks existent. Elles précisent une relation plus directe entre deux landmarks, telles que l'obligation de présence d'un landmark à l'état précédent l'obtention d'un autre. Dans notre exemple, il faut par exemple qu'un passager se trouve dans l'avion à l'état précédent l'apparition du passager à Toulouse, ainsi on a :

$$(\mathtt{dans\ a\ p1}) \rightarrow_n (\mathtt{à\ p_1\ toulouse})$$

21. au sens algorithmique *(soundness)*

Définition 27 : *Relation d'ordre impératif* $\rightarrow_n$ (necessary)

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et un ensemble de landmark* $\mathcal{L}$. *On dit qu'il y a un ordre* impératif *entre un couple* $(l, l') \in \mathcal{L}^2$, *noté* $l \rightarrow_n l'$ *si et seulement si* l *doit être vrai à l'état précédent chaque apparition de* l'. *C'est-à-dire :*

$$(\forall \rho \in \mathcal{P}^\star)(\forall j)\ l' \in I[\rho_{1\rightarrow j}] \implies l \in I[\rho_{1\rightarrow j-1}]$$

Définition 28 : *Relation d'ordre glouton impératif* $\rightarrow_{gn}$ (greedy necessary)

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification et un ensemble de landmark* $\mathcal{L}$. *On dit qu'il y a un ordre* glouton impératif *entre un couple* $(l, l') \in \mathcal{L}^2$, *noté* $l \rightarrow_{gn} l'$ *si et seulement si* $l \rightarrow_n l'$ *et* l *doit apparaitre pour la première fois à l'état précédent la première apparition de* l'. *C'est-à-dire :*

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l' \in I[\rho_{1\rightarrow j}]\ \textit{et}\ (\forall i < j-1)\ l' \notin I[\rho_{1\rightarrow i}] \implies l \in I[\rho_{1\rightarrow (j-1)}]$$

Par ailleurs, on peut aussi définir d'autres ordres, ne respectant pas la condition d'obtention d'un landmark avant un autre *dans tous les plans solutions*, tel que les ordres raisonnables *(reasonable orders)* [Hoffmann, Porteous et Sebastia, 2004]. L'intuition étant que si un landmark l doit être supprimé dans le but d'obtenir un landmark l' mais si l doit être réobtenu par la suite, alors il semble raisonnable de vouloir atteindre l' avant l. Cependant, dans certains problèmes, il peut être nécessaire d'atteindre le landmark l plusieurs fois. La relation d'ordre raisonnable n'est donc pas correcte.

3.3 Complexité

Théorème 3 : *Décider si un fluent est un landmark*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification. Le problème de décision défini par la question «* Est-ce qu'un fluent est un landmark ? *» est PSPACE-complet [Hoffmann, Porteous et Sebastia, 2004].*

Théorème 4 : *Décider si un fluent est un landmark pour* Π^+

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification. Le problème de décision défini par la question «* Est-ce qu'un fluent est un Landmark pour le problème Π^+ *» est dans P.*

Démonstration :

La démonstration vient directement de la démonstration du théorème 3. Décider si un fluent est un landmark de Π^+ revient à décider si une solution existe pour Π^+ sans ce fluent (Problème de décision PLANSAT$^+_+$ [Bylander, 1994]). ■

3.4 Algorithmes de recherche des landmarks

La méthode naïve pour décider si un fluent est un landmark provient de la démonstration précédente. Il s'agit donc de tester pour chaque fluent s'il existe une solution au problème sans ce fluent. Cette méthode est cependant trop coûteuse et ne permet pas de faire apparaître des ordres entre les landmarks.

3.4.1 Propagation en *avant* dans le RPG

Un parcours complet du graphe relaxé du problème de planification permet de trouver l'ensemble des landmarks causaux de Π^+ [Zhu et Givan, 2003]. La première étape consiste à calculer le graphe de planification relaxé (RPG) de Π : $GP(\Pi^+) = \langle \lambda^a, \lambda^f, \mathcal{E} \rangle$. Pour chaque fluent f, à chaque niveau de fluents λ^f_i du RPG, associons $\Delta_{\lambda^f_i}(f)$, un ensemble d'atomes. De la même manière, associons $\Delta_{\lambda^a_i}(a)$ à chaque action de chaque niveau d'actions λ^a_i. $\Delta_{\lambda^f_i}(f)$ (respectivement $\Delta_{\lambda^a_i}(a)$) représente l'ensemble des landmarks causaux du fluent f (respectivement de l'action a) au niveau λ_i.

Pour le premier niveau, chaque fluent est labellisé par lui-même : $(\forall f \in \lambda^f_1)\Delta_{\lambda^f_1}(f) = \{f\}$. Pour tout les autres niveaux :

$$\text{niveau d'actions :} \quad (\forall i > 1, pair)(\forall a \in \lambda^a_i) \quad \Delta_{\lambda^a_i}(a) = \bigcup_{f \in \lambda^f_{i-1}, f \in pre(a)} \Delta_{\lambda^f_{i-1}}(f)$$

$$\text{niveau de fluents :} \quad (\forall i > 1, impair)(\forall f \in \lambda^f_i) \quad \Delta_{\lambda^f_i}(f) = \{f\} \cup \bigcap_{a \in \lambda^a_{i-1}, f \in add(a)} \Delta_{\lambda^a_{i-1}}(a)$$

Théorème 5 : ***Recherche de landmarks causaux [Zhu et Givan, 2003]***

Si le RPG est calculé jusqu'à un point fixe, l'union des labels du dernier niveau n des fluents du but (c'est-à-dire $\bigcup_{f \in G} \Delta_{\lambda^f_n}(f)$) correspond exactement aux landmarks causaux du problème relaxé Π^+.

3.4.2 Propagation en *arrière*

Hoffmann, Porteous et Sebastia [2004] ont proposé un algorithme d'extraction de landmarks basé sur une propagation en arrière dans le RPG. Cet algorithme se décompose en plusieurs étapes. Une première partie consiste à établir un ensemble de fluents qui va être un ensemble de landmarks candidats. Cet ensemble va ensuite être filtré pour évaluer si chaque candidat est bien un landmark. Enfin une troisième étape ajoute des ordres raisonnables.

Pour générer l'ensemble des candidats, on part de landmarks connus : les fluents du but. On peut ensuite, par propagations en arrière itératives, trouver d'autres landmarks en appliquant des règles, telles que : « Si toutes les actions qui

ajoutent un landmark l partagent une même précondition, alors cette précondition est un landmark » :

$$(\forall l \in \mathcal{L})\ \text{Si}\ \exists f \in \bigcap_{a \in \left\{ a \in A \,\middle|\, l \in \text{add}(a) \right\}} \text{pre}(a)\ \text{alors}\ f\ \text{est un landmark}$$

Cette règle génère finalement assez peu de landmarks, ce qui amène à l'utilisation d'une deuxième où l'on utilise seulement les actions qui apparaissent au niveau d'action précédent la première apparition du landmark considéré. Cette nouvelle règle amène cependant à la création de faux landmarks, c'est pour cela qu'une étape de filtrage est nécessaire. Ce filtrage consiste à tester s'il existe une solution au problème relaxé en retirant toutes les actions qui produisent un landmark.

Cette méthode a aussi inspiré une méthode d'extraction de landmarks appliquée au modèle SAS^+ [Richter, Helmert et Westphal, 2008] et améliorée grâce à l'utilisation des graphes de transition de domaines spécifique à SAS^+.

3.4.3 Graphe ET-OU

Une généralisation possible de la méthode de propagation en avant est de représenter le problème de recherche d'un landmark sous forme d'un graphe ET-OU [Keyder, Richter et Helmert, 2010]. Un graphe ET-OU est un graphe orienté $G = \langle V_I, V_{AND}, V_{OR}, E \rangle$ où, V_I est un ensemble de nœud initiaux, V_{AND} et V_{OR} sont l'ensemble des nœuds ET et des nœuds OU. Enfin, E est simplement l'ensemble des arcs. Le problème relaxé peut être vu comme un graphe ET-OU, où les fluents de l'état initial sont chacun un nœud de l'ensemble V_I, les autres fluents sont des nœuds *OU* et les actions sont des nœuds *ET*. Les arcs représentent les préconditions et les additions de chaque action. La figure 3.2 présente le graphe ET-OU de notre problème-exemple Zenotravel. Un plan solution est une *justification* du but G dans ce graphe. C'est-à-dire, un graphe $J = \langle V_J, E_J \rangle$ tel que :

- tout les fluents du but G appartiennent à V_J ;
- les actions du plan relaxé sont l'ensemble des nœuds OU présents dans V_J, et doivent respecter le fait que l'ensemble des préconditions (des arcs arrivant sur les nœuds ET) soient présentes dans l'ensemble E_J ;
- de la même manière, l'ensemble des fluents doivent avoir été ajoutés par une action, c'est-à-dire qu'il doit exister au moins un arc entrant présent dans E_J pour chaque nœud OU de V_J ;
- enfin, J doit être acyclique.

Un landmark est simplement un nœud OU présent dans toutes les justifications du graphe G.

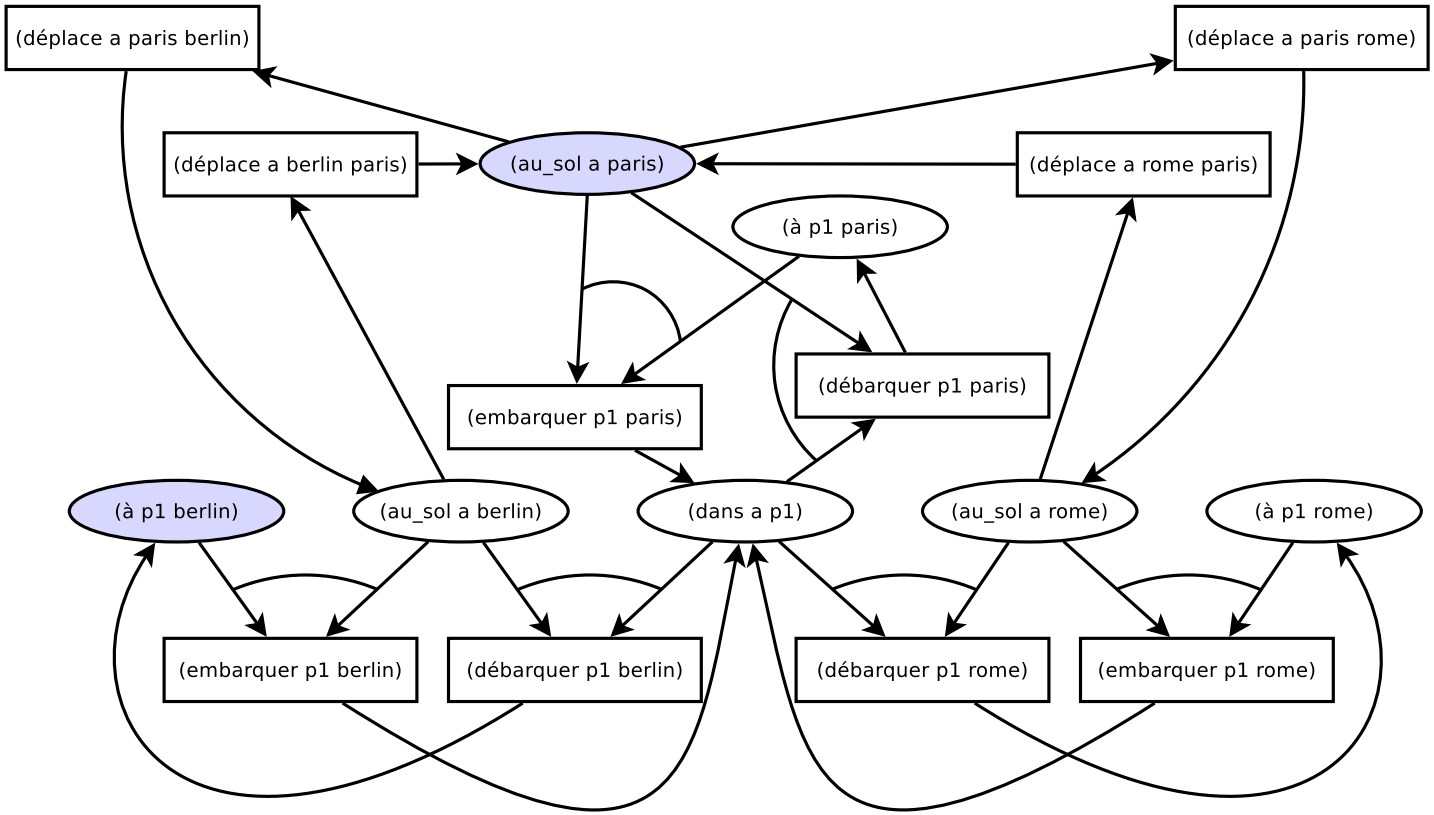


FIGURE 3.2 : Exemple d'un graphe ET-OU pour un problème simple du domaine Zenotravel. Les nœuds initiaux sont en bleu, les nœuds *OU* sont les ellipses et les nœuds *ET* sont les rectangles.

Théorème 6 : ***Ensemble des landmarks dans un graphe ET-OU***

Soit $G = \langle V_I, V_{AND}, V_{OR}, E \rangle$ *un graphe ET-OU :*

$$\mathcal{L} = \bigcap_{J=\langle V_J, E_J \rangle, \textit{ justification de } G} V_J$$

On peut trouver ces landmarks de la même manière que l'algorithme de propagation en avant dans le RPG. En appliquant cette méthode au graphe ET-OU du problème relaxé, on obtient exactement l'ensemble des landmarks causaux. Elle est donc identique à la méthode de Zhu et Givan [2003].

De la même manière que la famille d'heuristiques h^m peut être calculée par h^1 sur le problème Π^m, on peut calculer l'ensemble des landmarks causaux de Π^m permettant ainsi de capturer d'autres landmarks simples et des landmarks conjonctifs de taille inférieure ou égale à m.

3.5 Heuristique basée sur les landmarks h^{LM}

Une heuristique basée sur les landmarks qui s'est avérée très efficace consiste simplement à compter le nombre de landmarks qu'il reste à atteindre depuis l'état courant [Richter, Helmert et Westphal, 2008]. Pour cela, il faut pré-calculer un graphe de landmarks depuis l'état initial. On peut ensuite estimer la distance au but comme le nombre de landmarks totaux moins ceux que l'on a déjà atteints et plus ceux qu'il faut produire de nouveau. Un landmark est considéré comme admis *(accepted)* s'il est vrai dans l'état courant et si tous les landmarks ordonnés avant lui sont considérés comme admis dans l'état précédent. Un landmark admis à un état reste admis dans tous les états successeurs. Un *landmark* l est à nouveau nécessaire *(required again)* s'il n'est plus vrai dans l'état courant et s'il existe un landmark l' tel qu'il existe un ordre *glouton impératif* entre l et l'.

Définition 29 : h^{LM} — *Landmark heuristic* [Richter *et* Westphal, *2010]*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$ *un graphe de landmarks :*

$$h^{LM} : \left| \begin{array}{rcl} 2^F \times \mathcal{P} & \longrightarrow & \mathbb{N} \\ (s, \rho) & \longmapsto & \left| \left(\mathcal{L} \setminus A_{cc}(s, \rho) \right) \bigcup R_{eq}(s, \rho) \right| \end{array} \right.$$

où A_{cc} *est l'ensemble des landmarks admis et* R_{eq} *est l'ensemble des landmarks à nouveau nécessaires :*

$$A_{cc} : \left| \begin{array}{rcl} 2^F \times \mathcal{P} & \longrightarrow & 2^{\mathcal{L}} \\ (s, \rho) & \longmapsto & \begin{cases} \left\{ l \in \mathcal{L} \;\middle|\; l \in s \textit{ et } (\nexists l' \in \mathcal{L})\ l' \rightarrow l \right\} & \textit{si } \rho = \langle\rangle \\ A_{cc}\left(I\left[\rho'\right], \rho'\right) \bigcup \left\{ l \in \mathcal{L} \;\middle|\; l \in s \right. & \\ \quad \left. \textit{et } (\forall l' \in \mathcal{L}, l' \rightarrow l)\ l' \in A_{cc}\left(I\left[\rho'\right], \rho'\right) \right\} & \textit{sinon} \end{cases} \end{array} \right.$$

$$\textit{où } \rho' = \rho_{1 \rightarrow |\rho| - 1}$$

$$R_{eq} : \left| \begin{array}{rcl} 2^F \times \mathcal{P} & \longrightarrow & 2^{\mathcal{L}} \\ (s, \rho) & \longmapsto & \left\{ l \in A_{cc}(s, \rho) \;\middle|\; l \notin s \textit{ et} \right. \\ & & \quad \left. (l \in G \textit{ ou } (\exists l' \in \mathcal{L}, l' \rightarrow_{gn} l)\ l' \notin A_{cc}(s, \rho)) \right\} \end{array} \right.$$

Cet heuristique n'est pas uniquement basée sur l'état courant mais est dépendante du chemin (plan) qui a mené à cet état. De plus cette heuristique n'est pas admissible puisqu'une action peut produire plusieurs landmarks.

3.6 Utilisation des landmarks pour décomposer et résoudre des problèmes

3.6.1 Disjunctive Search Control

L'algorithme *Disjunctive Search Control* (DSC) [Porteous, Sebastia et Hoffmann, 2001] ; [Hoffmann, Porteous et Sebastia, 2004] est un algorithme basé sur l'utilisation des landmarks comme sous-buts. Cet algorithme calcule un graphe de landmarks basé sur la définition 20 et un ensemble d'ordres non nécessairement corrects. Cet algorithme décompose le problème en une suite de sous-problèmes ayant des landmarks en tant que buts intermédiaires. Pour cela, l'algorithme construit à chaque étape un nouveau sous-but pour le problème. Ce nouveau but est disjonctif, il est composé d'une disjonction du but global du problème G, et des landmarks racines du graphe de landmarks. Ce sous-problème est ensuite donné à résoudre à un planificateur sous-jacent (dans ce cas précis FF et LPG). Lorsqu'un landmark est atteint, il est enlevé du graphe de landmarks, et l'algorithme itère jusqu'à trouver

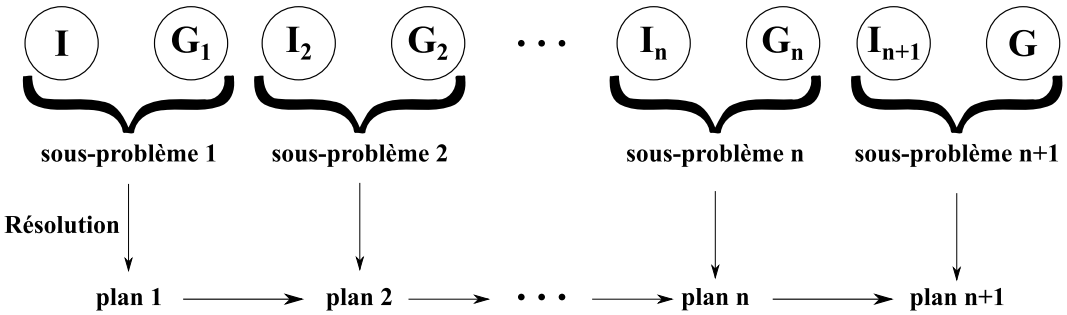


FIGURE 3.3 : Schéma de décomposition et résolution de STeLLa

le but du problème global. Cet algorithme est correct, mais non complet. En effet, si une des recherches intermédiaires mène à une impasse pour le problème original, alors la recherche globale échoue. Cet algorithme obtient de meilleurs résultats que le planificateur sous-jacent en terme de temps d'exécution, cependant on peut noter qu'il produit en moyenne des plans de qualité plus médiocre. En effet, l'utilisation d'une disjonction en tant que sous-but peut amener à une exploration un peu dispersée de l'espace de recherche, créant ainsi des plans solutions avec beaucoup d'actions inutiles [22].

3.6.2 SteLLa

L'algorithme STeLLa [Sebastia, Onaindia et Marzal, 2002] ; [Sebastia, Onaindia et Marzal, 2006] s'inspire de la méthode DSC. Une première différence à noter est le fait que les ordres utilisés entre les landmarks sont dans ce cas nécessairement corrects. Ensuite, à la place d'utiliser une disjonction des landmarks racines à chaque itération comme sous-buts, l'algorithme identifie des sous-ensembles de landmarks et créé une séquence de landmarks conjonctifs qui seront utilisés de manière séquentielle comme sous-buts. Une fois les sous-buts identifiés, cette méthode approxime des états initiaux intermédiaires pour finalement créer une séquence de sous-problèmes à résoudre indépendamment. Enfin, elle concatène les plans solutions de chacun de ces sous-problèmes (cf. figure 3.3). La résolution des sous-problèmes peut donc être faite de manière concurrente.

Les buts intermédiaires consistent donc en des conjonctions de landmarks. Pour séparer l'ensemble des landmarks en une séquence de conjonctions de landmarks, cet algorithme utilise un ensemble de règles :

- les landmarks ne doivent pas être mutuellement exclusifs ;
- un landmark peut être sélectionné pour un nouveau but intermédiaire si c'est un landmark racine (c'est-à-dire, tous ses prédécesseurs ont été sélectionnés pour des buts intermédiaires précédents) ;

22. Les plans solutions peuvent par exemple comporter des cycles.

– un landmark sélectionné pour un but intermédiaire est propagé aux sous-buts suivant jusqu'à ce qu'un de ses successeurs soit sélectionné.

Pour approximer les états initiaux intermédiaires, STeLLa utilise les *property space* [Fox et Long, 2011] qui permettent d'identifier des invariants dans le problème de planification[23]. Partant de l'état initial précédent $I_{i-1} = a, b_1$ et l'état but précédent $G_{i-1} = b_2$, si on a pu inférer que pour tous les états du problème on a $b_1 \vee b_2$, alors on peut approximer $I_i = a, b_2$.

Cependant, la résolution concurrente des sous-problèmes peut amener à une concaténation incorrecte des plans solutions. Il faut donc parfois réparer ces plans. Enfin, cet algorithme est correct, mais non complet.

3.6.3 SGPlan

SGPlan [Chen, Wah et Hsu, 2006] est un planificateur calculant une séquence de sous-buts à atteindre. Tout d'abord, il décompose en problème de planification en un ensemble de sous-problèmes, un pour chaque fluent du but G du problème. Ces sous-problèmes sont ensuite ordonnés grâce en particulier aux ordres raisonnables. Chaque sous-problème est décomposé à nouveau à l'aide d'une séquence de landmarks. Pour chaque sous-problème, les actions non pertinentes sont enlevées pour réduire l'espace de recherche. Enfin cette séquence de sous-problèmes est résolue de manière séquentielle.

23. Par exemple, dans le problème Zenotravel, l'avion se trouve toujours dans une des villes.

4 PLANIFICATEURS

Nous allons maintenant présenter quelques planificateurs efficaces et en particulier quelles techniques ils mettent en œuvre. Nous présenterons également les résultats de deux dernières compétitions internationales de planification (IPC).

Sommaire

4.1 Le framework Fast-Downward

Fast-Downward [Helmert, 2006] n'est pas qu'un planificateur, mais il est plutôt un cadre d'applications à partir duquel on peut construire un planificateur. Il utilise le formaliste SAS$^+$. Son ambition est de permettre d'une part une implémentation plus rapide de nouveaux planificateurs puisqu'on peut réutiliser des techniques déjà implémentées. Et d'autre part, à pouvoir comparer des planificateurs plus justement en homogénéisant certaines étapes comme le *parsing*, ou encore la gestion de la mémoire pour enregistrer les états.

4.2 Planification par recherche heuristique

4.2.1 Fast-Forward (FF)

FF [Hoffmann et Nebel, 2001] est un planificateur basé sur un algorithme d'*Enforced Hill-Climbing*, et utilisant l'heuristique h^{ff}. Il se sert aussi des *helpful-actions* : seuls les états successeurs atteignables par des actions du plan relaxé sont considérés. En cas d'échec de la recherche locale, le planificateur se replie sur un algorithme de type *Best-First Search*.

4.2.2 Yet Another Heuristic Search Planner (YAHSP)

YAHSP3 [Vidal, 2011] ; [Vidal, 2014] utilise un *weighted*-A* combiné avec l'heuristique h^{add}. De plus, à chaque développement de nœud, ce planificateur exécute un lookahead basé sur le plan relaxé.

YAHSP3 est utilisé par le planificateur DAE$_{YAHSP}$ [Bibaï et al., 2010] ; [Dréo et al., 2014] qui est algorithme évolutionnaire basés des séquences de sous-buts. YAHSP cherche un plan itérativement dans cette séquence. On peut remarquer que cette methode est proche de celle appliqué par l'algorithme STeLLa, sauf qu'il n'est pas appliqué avec des landmarks et que la séquence de sous-but évolue grâce à un processus basé sur un algorithme évolutionnaire.

4.2.3 LAMA

Le planificateur LAMA [Richter et Westphal, 2010] est basé sur un algorithme de type *Best-First Search* : un *weighted*-A* avec évaluation différée. Il est construit au-dessus du framework Fast-Downward, et donc utilise la représentation SAS$^+$. Il utilise le système des listes ouvertes multiples avec les deux heuristiques h^{LM} et h^{ff} et les *preferred operators*. Lorsque LAMA trouve une première solution, il fait un redémarrage de l'algorithme en réduisant le poids ω du *weighted*-A* de manière à améliorer la qualité de la solution.

4.2.4 Jasper

Jasper [Xie et al., 2014] ; [Xie, Müller et Holte, 2014a] ; [Xie, Müller et Holte, 2014b] est un planificateur qui s'inspire de LAMA-2011. Il utilise une recherche de type *Best-First* en raffinant le système de listes ouvertes multiples grâce notamment à un partitionnement de l'espace de recherche sur espace discret en deux dimensions grâce au coût depuis l'état initial jusqu'à l'état courant et l'évaluation heuristique h^{ff} (chaque couple est appelé type de nœud). La recherche alterne entre sa sélection du prochain nœud à développer entre la liste ouverte classique et les listes de types. Dans le cas des listes de types, une liste est sélectionnée aléatoirement, de même qu'un nœud à l'intérieur de celle-ci. Cette méthode permet de diversifier la recherche qui se focalise parfois dans une mauvaise direction à cause des heuristiques issues du problème relaxé.

4.2.5 Mercury

Mercury [Katz et Hoffmann, 2014] a été implémenté sur le framework Fast-Downward, et utilise un *weigted-A** de la même manière que LAMA. Cependant, la recherche est guidée grâce à l'heuristique basée sur la planification rouge-noir. Le *tie-breaking* se fait grâce à h^{LM}.

4.3 Marche aléatoire

Arvand [Nakhost et Mueller, 2013] utilise des marches aléatoires de Monte Carlo pour faire de la recherche locale dans l'espace d'états. Ce système adapte les paramètres de ses descentes le long de l'espace de recherche et des redémarrages de l'algorithme en fonction de l'amélioration des valeurs calculées par l'heuristique h^{ff}. Il biaise aussi la sélection aléatoire des actions via une préférence vers les *preferred operators*.

4.4 Compilation dans un autre modèle

Madagascar (M, Mp et MpC) [Rintanen, 2012] est un planificateur basé sur la compilation des modèles de planification en modèle de satisfaction de formules logiques booléennes (SAT). Le problème est encodé jusqu'à un horizon fini (nombre d'actions par plan fini), différentes techniques sont donc mises en œuvre pour sélectionner cet horizon et le faire grossir jusqu'à ce que le solveur SAT sous-jacent trouve une solution (résolution incrémentale, résolution en parallèle avec différents horizons).

4.5 Parallélisation

L'intérêt porté sur les algorithmes parallèles en planification de tâches prend de l'ampleur depuis les cinq dernières années. En effet, lors de la 7e compétition internationale de planification (IPC 2011 [Olaya, López et Jiménez, 2011]), une nouvelle épreuve a été créée pour les planificateurs basés sur des algorithmes parallèles. Cette épreuve a regroupé 8 planificateurs. Les architectures multi-cœurs, multi-processeurs ont le potentiel de fournir les capacités en mémoire et en calcul demandé pour résoudre des problèmes difficiles dans des temps raisonnables pour la planification optimale et sous-optimale.

Il existe plusieurs grandes méthodes pour paralléliser des algorithmes de recherches. On peut principalement les regrouper en deux catégories. Une première manière de créer un planificateur parallèle est de chercher à changer complètement le cœur des algorithmes, en essayant de découper soit l'espace de recherche en plusieurs ensembles complètement disjoints, soit en essayant de partitionner le problème en sous-ensembles de tâches. D'un autre côté, on peut lancer différents algorithmes (ou le même algorithme avec des paramètres différents) en parallèle sur différents processeurs en essayant de maximiser la diversité pour compenser les défauts d'un algorithme par les forces d'un autre. La difficulté ici est de choisir un ensemble cohérent de planificateurs pour exploiter au maximum les qualités de chacun de ceux-ci.

4.6 Résultats de la 7e compétition de planification (IPC 7)

Cette septième édition n'a pas vu d'introduction de nouvelle extension au langage PDDL ce qui était le cas lors des précédentes compétitions (cf. figure 4.1). Cependant une nouvelle catégorie a été introduite s'intéressant aux algorithmes parallèles. De plus, l'introduction d'un programme de gestion de lancement et de comparaison a été introduit, permettant une organisation plus simple pour les prochains organisateurs[24]. Nous utiliserons en partie cet outil lors de nos expérimentations.

Les résultats de la dernière compétition de planification [Olaya, López et Jiménez, 2011] dans la catégorie *satisficing* démontrent une fois de plus la suprématie des planificateurs basés sur la recherche en avant guidée par une heuristique avec LAMA. On peut aussi constater que deux planificateurs basés sur la marche aléatoire sont dans le top 10 (Roamer et Arvand). De plus, tous les planificateurs se basent, au moins en partie, sur les landmarks. Neuf des vingt-septs participants ont battu le gagnant de l'IPC 6 de cette catégorie, montrant une bonne progression des techniques de planification [Coles et al., 2012]

24. Programme de gestion de l'IPC 7 : `http://www.plg.inf.uc3m.es/sw-ipc2011`

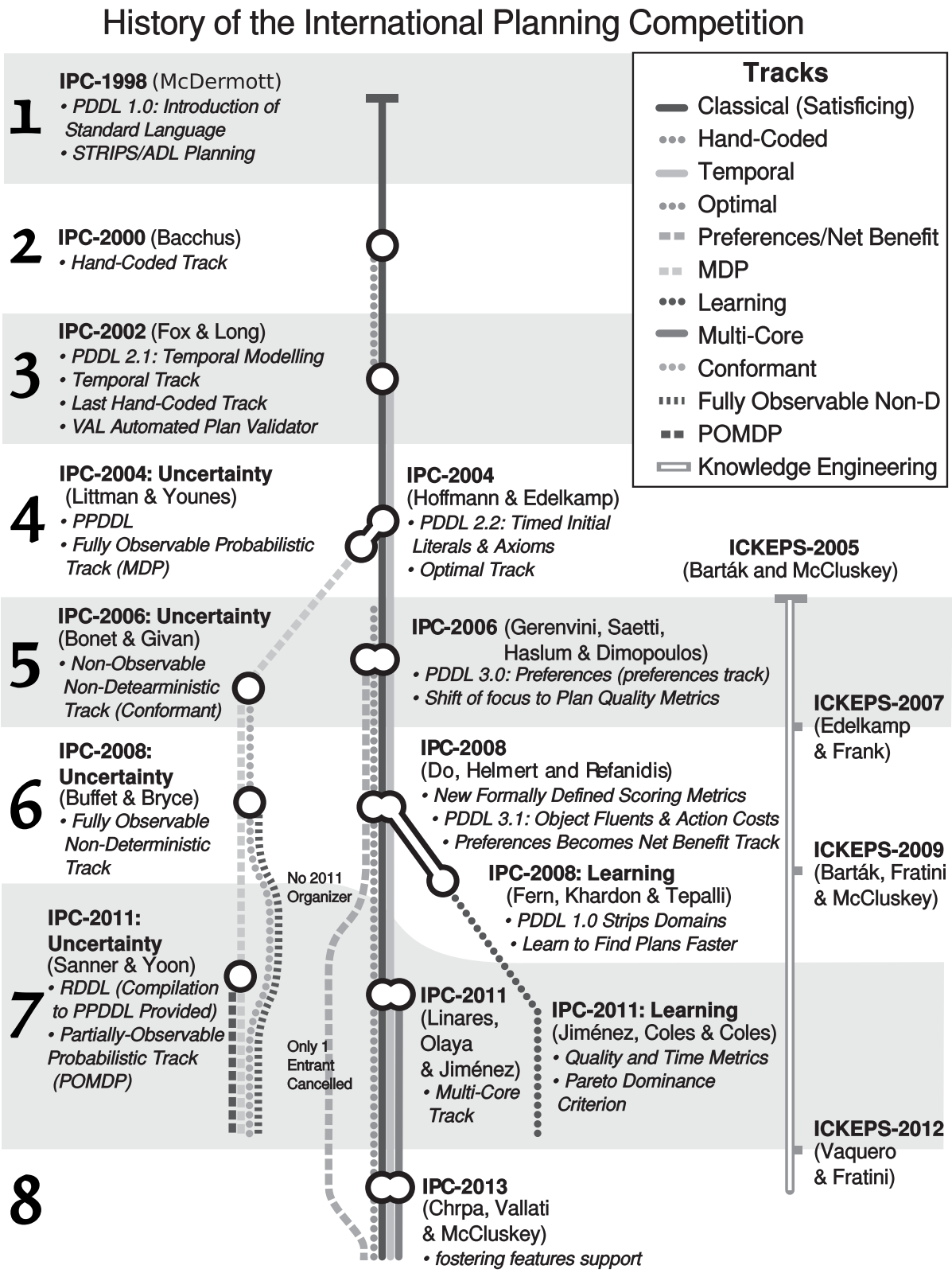


FIGURE 4.1 : Historique des compétitions internationales de planification (IPC). Figure extraite de [COLES et al., 2012] (seule la compétition déterministe a été mise à jour).

La catégorie multi-cœur a montré les excellentes performances des portfolios. En effet, les deux premiers planificateurs étaient des portfolio : ArvandHerd [Valenzano et al., 2012] et Ay Also Plan [Ernits, Gretton et Dearden, 2011]. Les planificateurs portfolios ont aussi montré de bonne performance, même en séquentiel. On peut cependant noter que même en utilisant les possibilités offertes par les processeurs multi-cœurs, ces planificateurs ont montré des performances en deçà de LAMA, vainqueur de la catégorie séquentielle.

4.7 Résultats de la 8e compétition de planification (IPC 8)

Dans la lignée de la septième édition, la dernière compétition de planification n'a pas non plus introduit de nouvelle extension au langage PDDL. Comme pour la septième compétition, la caractéristique *action-cost* a été conservée dans la plupart des domaines. Ils sont par ailleurs composés de préconditions négatives et d'effets conditionnels de manière à favoriser la généralisation de ces caractéristiques à tous les planificateurs.

En termes de vitesse de résolution des problèmes (catégorie *agile*), YAHSP3 est premier, suivi de Madagascar et de PROBE. Trois planificateurs basés sur trois méthodes différentes.

La catégorie *satisficing* montre aussi une belle réussite des planificateur portfolio, avec IBaCoP2 (vainqueur) et MIPlan (3e). Les premiers planificateurs non-portfolio sont Mercury (2e) et Jasper (4e). On peut constater que ce sont deux planificateurs qui essaient de pallier les problèmes des heuristiques basées sur la relaxation de manières différentes : l'un en incorporant des informations sur les retraits des actions et l'autre en diversifiant la recherche. L'ancien vainqueur aurait été classé 12e, attestant une bonne amélioration des méthodes de planification.

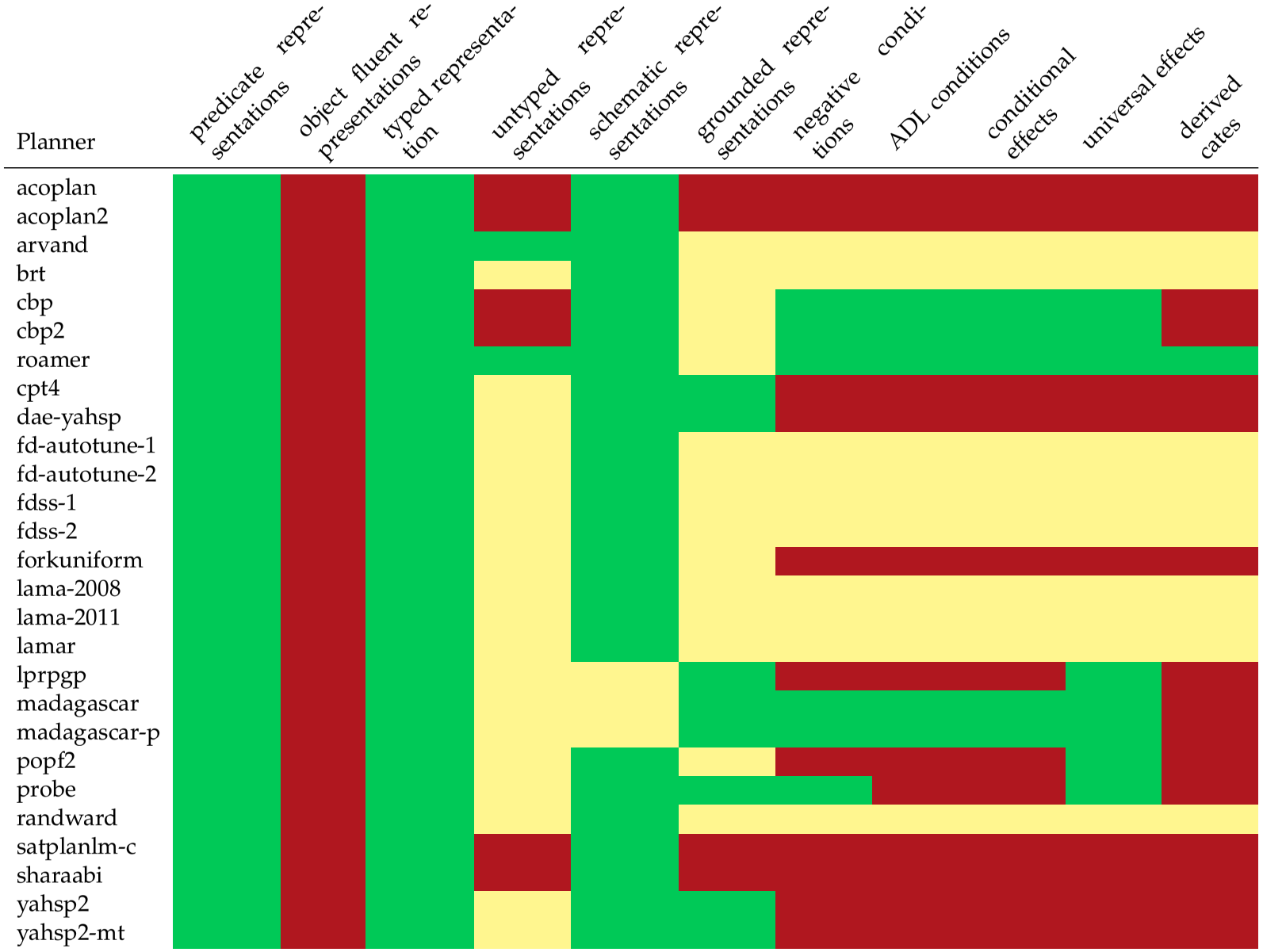
Dans la categorie multi-cœur, on voit à nouveau une domination des planificateurs portfolios, avec ArvandHerd encore vainqueur.

4.8 Support des caractéristiques de PDDL

Enfin, on peut regarder l'évolution du support des caractéristiques du langage PDDL entre les deux dernières IPC (résumé figures 4.1 et 4.2).

On peut remarquer que la caractéristique *object fluent* n'est supportée par aucun planificateur. Elle permet de créer des variables à valeur dans des domaines finis. Cette particularité est proche de la représentation SAS^+.

Les caractéristiques provenant d'ADL se démocratisent parmi les planificateurs, certainement grâce à l'augmentation de l'utilisation du *framework* Fast-Downward.



Planner	predicate representations	object fluent representations	typed representation	untyped representations	schematic representations	grounded representations	negative conditions	ADL conditions	conditional effects	universal effects	derived predicates
Total	27 100.00%	0 0.00%	27 100.00%	21 77.78%	27 100.00%	23 85.19%	16 59.26%	15 55.56%	15 55.56%	18 66.67%	11 40.74%

TABLEAU 4.1: Support des caractéristiques du langage PDDL parmis les planificateurs de l'IPC 7. Le vert correspond au support, le jaune au support de la caractéristique mais préfère ne pas l'utiliser et rouge ne supporte pas.

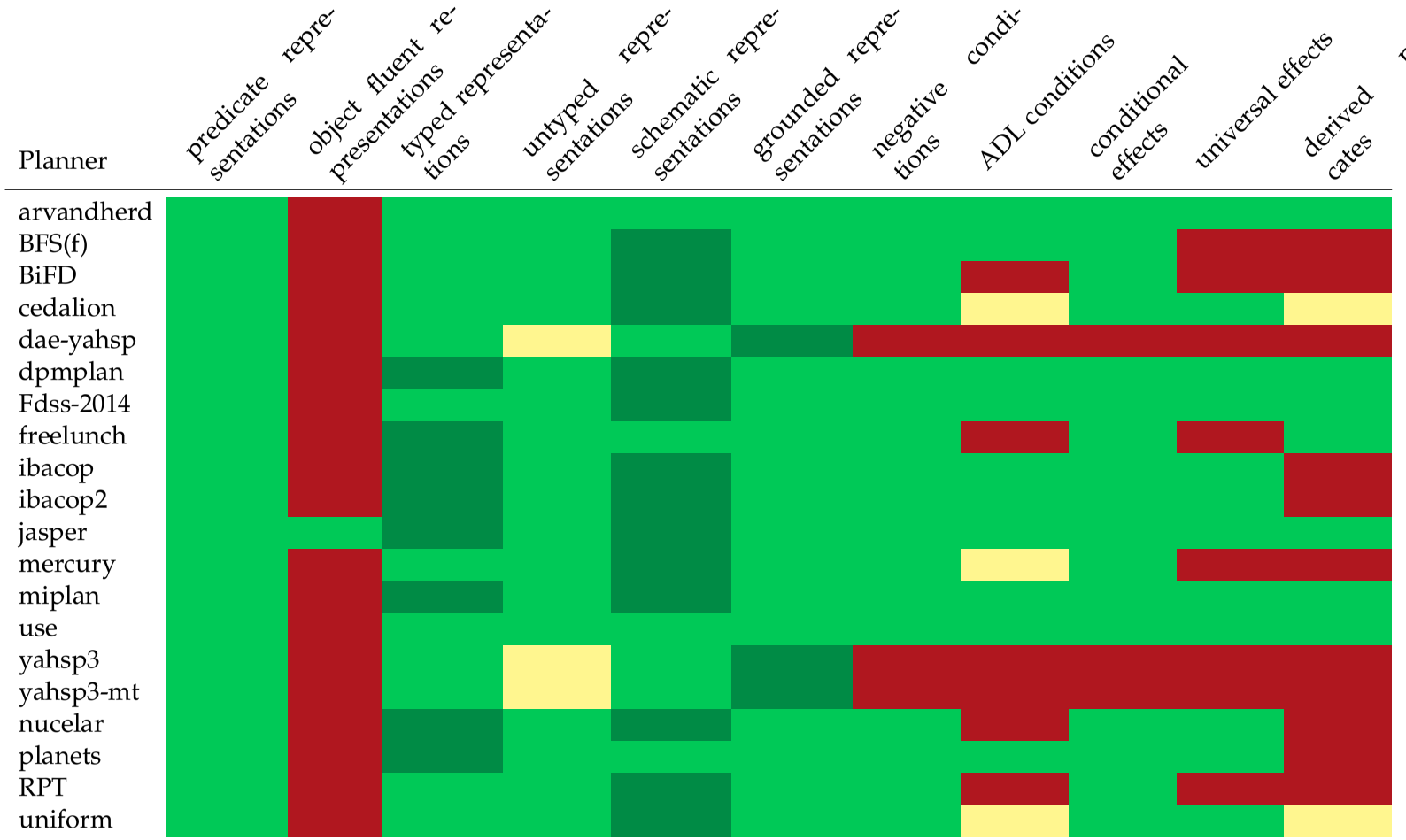


TABLEAU 4.2: Support des caractéristiques du langage PDDL parmis les planificateurs de l'IPC 8. Le vers foncé correspond au support et à la préférence, le vert correspond au support, le jaune au support de la caractéristique mais préfère ne pas l'utiliser et rouge ne supporte pas.

Deuxième partie

LMBFS

5

L'ALGORITHME LANDMARK-BASED META BEST-FIRST SEARCH

Les algorithmes DSC, SGPlan et STeLLa proposent une manière naturelle d'utiliser les landmarks en s'en servant de buts intermédiaires à atteindre avant le but global du problème. Cependant, peu d'extensions de ces algorithmes ont été proposées par la suite. Les landmarks sont maintenant principalement utilisés pour créer des heuristiques efficaces en planification classique optimale ou sous-optimale.

Même si ces dernières méthodes sont très efficaces, nous pensons que la segmentation d'un problème de planification en sous-problèmes via l'aide d'une analyse sur les landmarks est une piste encore mal explorée et potentiellement efficace. C'est pourquoi nous avons développé une méthode plus flexible que les approches précédentes ; rendant par ailleurs notre méthode de recherche complète.

Sommaire

5.1 Vue globale de l'algorithme

l'algorithme DSC impose des buts intermédiaires disjonctifs aux sous-problèmes qu'il génère, imposant ainsi au planificateur sous-jacent un choix dans la direction à suivre dans l'espace de recherche, menant ainsi à peu de focalisation dans l'espace de recherche et finalement créant des plans un peu plus longs. Pour remédier à ce problème, STeLLa détermine à l'avance une séquence de conjonctions de landmarks, mais reste lié à une unique séquence qui l'amène à être incomplet. C'est pourquoi, l'algorithme Landmark-based Meta Best-First Search (LMBFS) que nous proposons est un algorithme de recherche de type *Best-First* dans l'espace des ordres de landmarks. À partir d'un graphe de landmarks qu'il considère comme un ordre partiel, il va construire des ordres totaux à la manière des algorithmes de tri topologique.

Définition 30 : ***Ordre total de landmarks (séquence)***

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$ *un graphe de landmarks associé à* Π.
On dit que $\mathcal{T} = \langle l_1, \ldots, l_n \rangle$, *avec* $n \leqslant |\mathcal{L}|$ *est un ordre total compatible avec* Γ *si et seulement s'il respecte les deux conditions :*

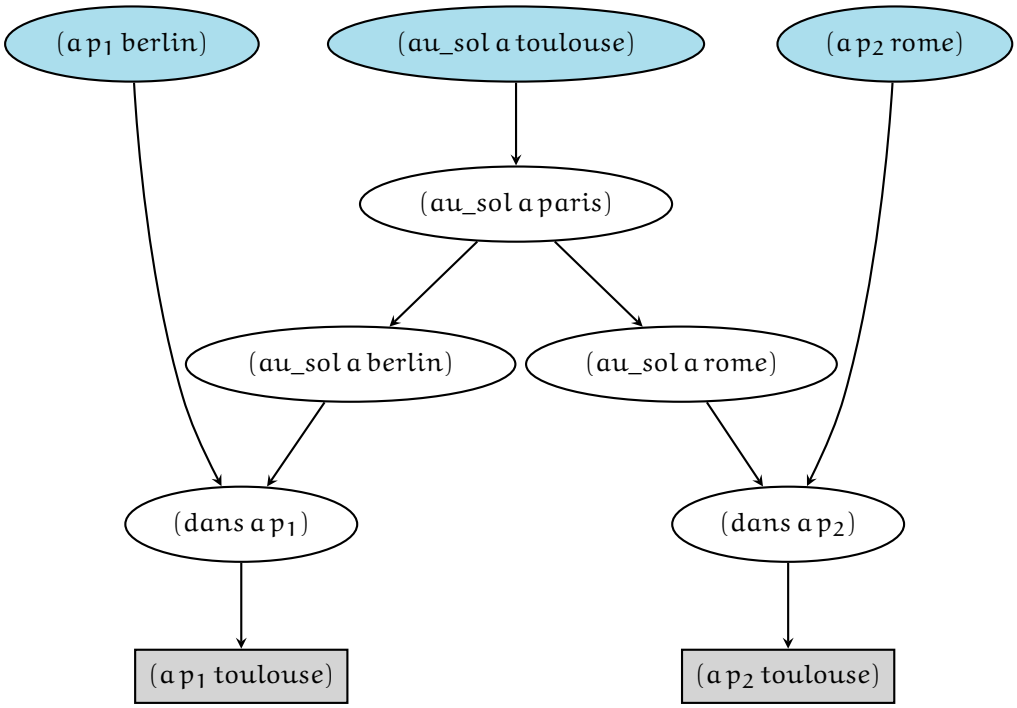
$$\textit{unicité :} \quad \big(\forall (i,j) \in [\![1,n]\!]^2\big) \quad i \neq j \implies l_i \neq l_j$$
$$\textit{compatibilité :} \quad \big(\forall (i,j) \in [\![1,n]\!]^2\big) \quad i < j \implies l_i \rightarrow l_j$$

On peut noter que tous les landmarks ne sont pas nécessairement présents dans les ordres totaux dérivés. En particulier ceux de l'état initial ne sont jamais considérés car déjà atteints. L'espace de recherche de l'algorithme consiste donc en l'arbre des ordres topologiques de l'ordre partiel, comme on peut voir en exemple figure 5.1(b), où chaque branche de l'arbre est un ordre total.

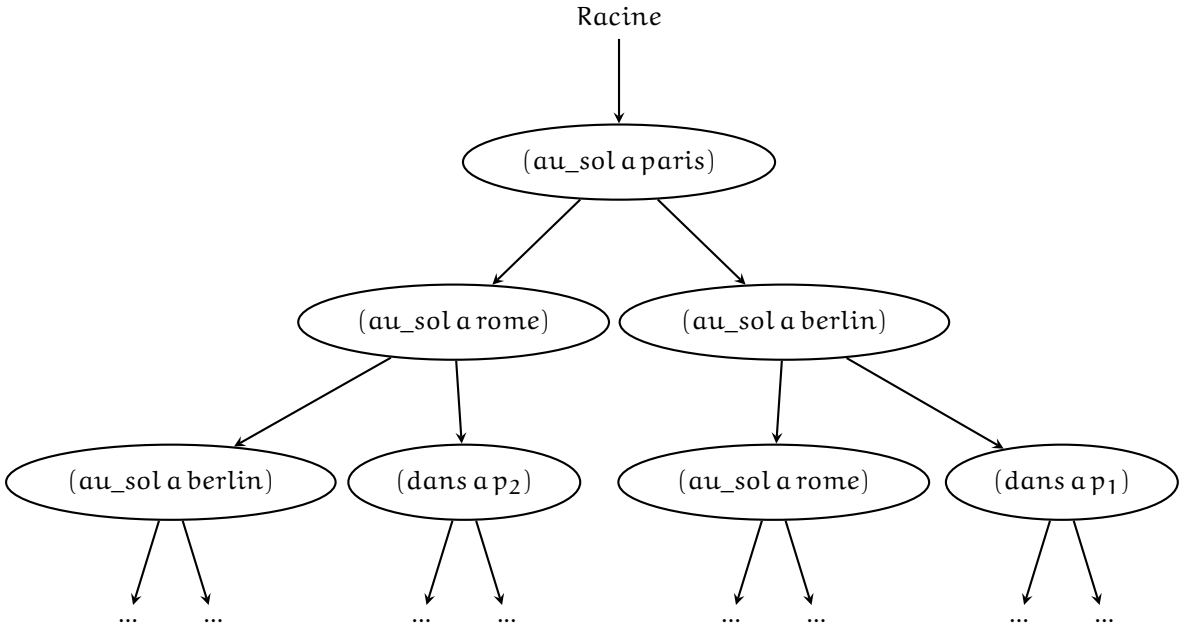
Chaque nœud de la recherche va demander la résolution d'un sous-problème de planification classique. Ces sous-problèmes peuvent être résolus par n'importe quel planificateur (ici, nous utilisons le modèle STRIPS, mais cet algorithme pourrait très bien s'appliquer directement au modèle SAS$^+$).

L'algorithme 3 présente un aperçu de notre méthode. On va tout d'abord extraire le graphe de landmarks. Ensuite, on va générer les nœuds initiaux de la recherche, c'est-à-dire l'ensemble des sous-problèmes qui partant de l'état initial auront un premier landmark comme but, pour les insérer dans la liste ouverte (détaillé section 5.2). Puis, on entre dans la boucle générale de l'algorithme où l'on commence tout d'abord par sélectionner le nœud le plus prometteur de la liste ouverte afin de pouvoir le développer, c'est-à-dire générer les nœuds fils dans l'arbre d'exploration (détaillé section 5.3). On doit alors résoudre chacun de ces sous-problèmes (détaillé section 5.4), les évaluer et les insérer dans la liste ouverte. Finalement, on itère jusqu'à avoir trouvé le but, ou qu'éventuellement le liste ouverte soit vide[25].

25. Si l'algorithme est complet, une liste ouverte vide correspond au fait que l'espace de recherche a été épuisé. Si aucun but n'a été trouvé, alors le problème ne possède aucune solution



(a) Graphe de landmarks de l'exemple Zenotravel (en bleu les fluents de l'état initial, et en gris ceux du but).



(b) Arbre d'exploration correspondant.

FIGURE 5.1 : Exemple d'exploration de l'algorithme LMBFS à partir d'un graphe de landmarks du problème Zenotravel. Chaque branche de l'arbre d'exploration est un tri topologique du graphe.

Algorithme 3: LMBFS — Vue globale

Entrées : Problème STRIPS $\Pi = \langle F, A, I, G \rangle$
Sorties : Plan solution (ou $\perp$ s'il n'y a pas de solutions)

Section 3.4 Analyse du problème : Création du graphe de landmarks.
Section 5.2 Insertion des nœuds initiaux dans la liste ouverte
répéter
  Sélection d'un problème π depuis la liste ouverte
Section 5.3 Génération des sous-problèmes fils de π
Section 5.4 Résolution des sous-problèmes et insertion dans la liste ouverte
jusqu'à $\mathcal{O}$ *vide ou but* G *atteint*
retourner *Plan solution*

5.2 Métanœud : sous-problème

Les nœuds de notre algorithme de recherche définissent en fait un problème de planification complet, nous les appelons métanœuds pour les différencier des nœuds de recherche de la planification sous-jacente. Ils sont définis grâce à un état du problème de planification, qui constituera l'état initial du sous-problème, ils se focalisent sur un landmark unique, et tiendront à jour la liste des landmarks déjà atteints.

Définition 31 : *Metanœud*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\mathcal{L}$ *un ensemble de landmarks du problème* Π.
On note alors m, *un* métanœud *comme un n-uplet* $m = \langle s, l, \mathcal{A} \rangle$ *où :*

- s *est un état du problème de planification* Π *;*
- l *est un landmark (*$l \in \mathcal{L}$*) ;*
- $\mathcal{A}$ *est un ensemble de landmarks (*$\mathcal{A} \subseteq \mathcal{L}$*).*

De la même manière que le propose SGPlan, seul un sous-ensemble des actions jugées intéressantes peuvent être disponibles pour le sous-problème, par exemple les actions permettant l'acquisition de landmarks autres que le landmark but peuvent être interdites. L'intention est de créer des problèmes de planification aussi simples que possible pour qu'ils soient résolus facilement. Une fonction de sélection des actions associées à un sous-problème est définie telle que :

Définition 32 : *Fonction de sélection des actions applicables*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A} \rangle$ *un métanœud et* Γ *un graphe de landmarks associé au problème* Π.
On définit l'ensemble de définition des fonctions de sélection des actions tel que :

$$ops_\Gamma : \left| \begin{array}{lcl} 2^F \times \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & 2^{\mathcal{A}} \\ m = \langle s, l, \mathcal{A} \rangle & \longmapsto & ops_\Gamma(m) \end{array} \right.$$

Enfin, à chaque métanœud, on peut associer un problème de planification :

Définition 33 : *Problème de planification associé au Métanœud*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A} \rangle$ *un métanœud et* Γ *un graphe de landmarks.*
On y associe le problème de planification noté Π_m *tel que*

$$\Pi_m = \langle F, ops_\Gamma(m), s, \{l\} \rangle$$

où $ops_\Gamma(l, \mathcal{A})$ *est une fonction de sélection des actions applicables.*

Dans un premier temps, on considérera simplement que toutes les actions sont sélectionnées, c'est-à-dire que pour tout métanœud m, $ops_\Gamma(m) = A$.

5.3 Génération des sous-problèmes

Afin d'explorer notre espace de recherche, il faut définir une manière de générer des métanœuds fils afin de pouvoir développer un métanœud donné. Il s'agit de générer un ensemble de sous-problèmes successeurs permettant de se rapprocher du but G du problème original en suivant l'ordre partiel défini par le graphe de landmarks Γ, tout en gardant à l'esprit que chacune des branches de l'arbre d'exploration doit être un ordre total unique, différent de toutes les autres branches. Nous allons donc définir différentes fonctions capables de générer des métanœuds fils et voir le rôle qu'elles jouent sur la complétude de l'algorithme.

5.3.1 Les racines du graphe de landmarks

On peut donc tout d'abord définir un premier générateur de métanœuds qui va permettre la création de tous les tris topologiques complets du graphe de landmarks [26]. Un peu à la manière de l'algorithme DSC qui prend comme but la disjonction de landmarks racines, ici nous générons une disjonction de métanœuds, un pour chaque landmark racine. Donc, à partir d'un métanœud, on va retirer l'ensemble des landmarks déjà atteints et aussi le landmark but du graphe de landmarks Γ et générer un métanœud fils pour chacun des landmarks racines.

26. c'est-à-dire de générer tous les ordres totaux possibles à partir de l'ordre partiel donné par le graphe de landmarks

Définition 34 : *Mise à jour des landmarks atteints*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A}, \rangle$ *un métanœud et* Γ *un graphe de landmark associé au problème* Π.
Si le problème Π_m *a une solution* ρ*, alors le nouvel ensemble des landmarks atteints est :*

$$A_{tt} : \left| \begin{array}{rcl} 2^{\mathcal{L}} \times 2^{A} & \longrightarrow & 2^{\mathcal{L}} \\ (\mathcal{A}, \rho) & \longmapsto & \mathcal{A} \cup \left\{ l \in \mathcal{L} \;\middle|\; \exists a \in \rho, l \in add(a) \right\} \end{array} \right.$$

Définition 35 : *Générateur de métanœuds : Next landmarks*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A}, \rangle$ *un métanœud et* Γ *un graphe de landmark associé au problème* Π.

$$\text{nextLM} : \left| \begin{array}{rcl} 2^{F} \times \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & 2^{2^{F} \times \mathcal{L} \times 2^{\mathcal{L}}} \\ m = \langle s, l, \mathcal{A} \rangle & \longmapsto & \begin{cases} \left\{ \left\langle s[\rho], l', A_{tt}(\mathcal{A}, \rho) \right\rangle \;\middle|\; l' \in \mathcal{R}\left(\Gamma \setminus A_{tt}(\mathcal{A}, \rho)\right) \right\} & \text{Si } \Pi_m \text{ a une solution } \rho \\ \emptyset & \text{Sinon} \end{cases} \end{array} \right.$$

Ainsi, lors du développement d'un métanœud m, on essaie d'atteindre son landmark but l. Dans le cas où un plan solution existe, on applique le plan à l'état initial s et on met à jour l'ensemble des landmarks déjà atteints $\mathcal{A}$ en y ajoutant le landmark l. Ensuite, on génère un nouveau métanœud par landmark racine de Γ privé des landmarks déjà atteints.

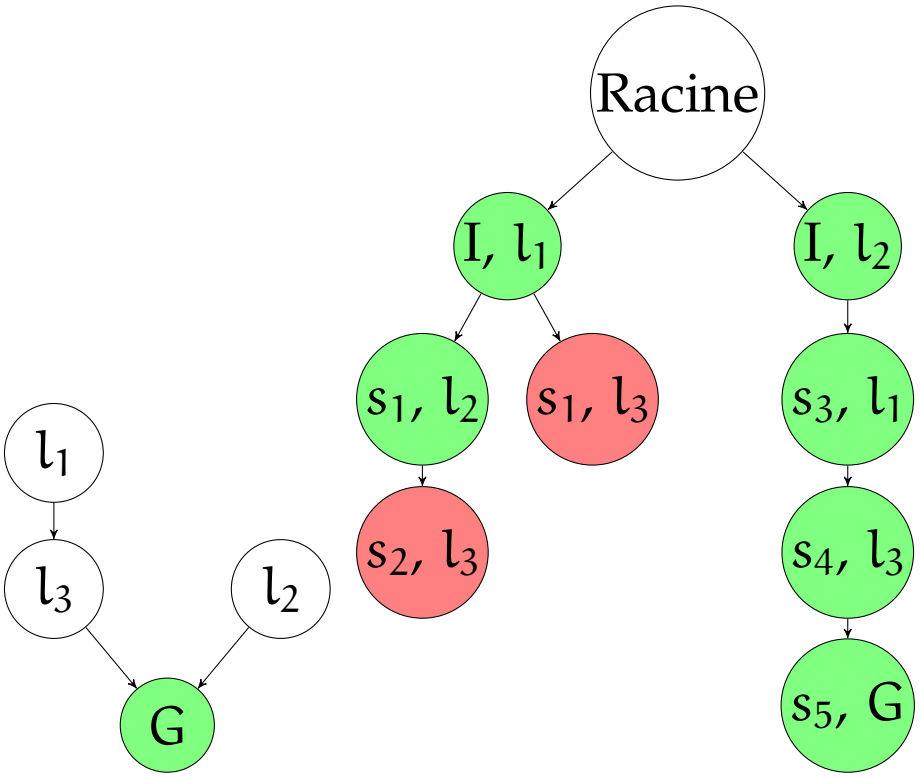
Avec ce premier opérateur nextLM, on peut explorer tous les ordres totaux complets dérivables à partir de l'ordre partiel offert par le graphe de landmarks. En effet, appliqué récursivement, cet opérateur nous donne bien l'arbre d'exploration décrit figure 5.1(b).

5.3.2 Exemple de déroulement de l'algorithme

Nous allons détailler le déroulement de l'algorithme au travers d'un exemple sur un problème simple ayant un état initial I, un but G et trois landmarks l_1, l_2 et l_3. Dans cet exemple nous ne détaillerons pas les plans solutions partiels trouvés ni les heuristiques [27]. Les métanœuds seront simplement notés (s, l) avec s l'état initial du métanœud et l le landmark but.

La figure 5.2(b) détaille l'arbre de recherche de l'algorithme LMBFS. Cet arbre se déroule du haut vers le bas, puis de la gauche vers la droite. À l'initialisation de l'algorithme, deux métanœuds sont ajoutés dans la liste ouverte (les deux fils de la

27. Nous considérons que l'algorithme se comporte comme un algorithme de recherche en profondeur d'abord lors de la sélection du prochain métanœud à développer.



(a) Graphe des landmarks

(b) Arbre de recherche. Les metanœuds où la planification du problème associé a réussi (respectivement a été un échec) ont un fond vert (respectivement rouge).

FIGURE 5.2 : Exemple de déroulement de l'algorithme sur un problème simple.

racine dans l'arbre) ayant tous deux l'état initial I du problème global comme état initial et respectivement les landmarks l_1 et l_2 comme buts.

À l'itération suivant, l'algorithme sélectionne le métanœud (I, l_1). Le planificateur sous-jacent trouve une solution amenant à l'état s_1. Donc, le landmark l_1 est retiré du graphe des landmarks, faisant apparaitre deux nouveaux landmarks racines : l_2 et l_3. Le générateur de métanœuds nextLM est ensuite appelé, et crée deux nouveaux métanœuds (s_1, l_2) et (s_1, l_3).

On itère en choisissant maintenant (s_1, l_2), et on trouve un plan amenant à l'état s_2. On enlève donc l_2 du graphe de landmarks, permettant à nextLM de générer le métanœud (s_2, l_3).

Cependant, lors la planification du sous-problème suivant, le planificateur sous-jacent ne trouve aucune solution pour le sous-problème $\Pi_{(s_2, l_3)}$. Donc la recherche continue avec un autre métanœud de la liste ouverte, jusqu'à éventuellement atteindre le but. Dans cet exemple, la branche $l_2 \rightarrow l_1 \rightarrow l_3 \rightarrow G$ mène bien au but global du problème et donc notre algorithme termine bien et trouve une solution.

Comme on peut le voir dans cet exemple, une fois l'arbre de landmarks vide, on planifie une dernière fois de l'état courant vers le but global du problème s'il n'a toujours pas été atteint. En effet, il arrive qu'un landmark but qui ait été atteint soit supprimé lors de la recherche d'un autre landmark. Pour implémenter ce cas de figure, on peut simplement ajouter un fluent factice f_g représentant l'achèvement du but global, et une action a_g ayant comme précondition tous les fluents du but G et ajoutant le fluent f_g.

5.3.3 Complétude

Cependant, même si le graphe des landmarks est consistant et complet, le générateur nextLM ne rend pas l'algorithme LMBFS complet. En effet, prenons en exemple le problème suivant :

Exemple 6 : *Générateur* nextLM *: problème des effets négatifs*

Considérons le problème $\Pi = \langle F, A, I, G \rangle$ *où :*

- $F = \{a, b, c, a', b', c', g\}$;
- $I = \{a, b, c\}$;
- $G = \{g\}$;
- $A = \{\alpha, \beta, \gamma, \delta, \epsilon\}$, *avec :*
  - $\alpha = \langle \{a\}, \{a'\}, \{\} \rangle$;
  - $\beta = \langle \{b\}, \{b'\}, \{a, c\} \rangle$;
  - $\gamma = \langle \{c\}, \{c'\}, \{\} \rangle$;
  - $\delta = \langle \{a', b'\}, \{g\}, \{\} \rangle$;
  - $\epsilon = \langle \{b', c'\}, \{g\}, \{\} \rangle$.

La figure 5.3 représente les relations entre les fluents et les actions de ce problème.

Dans cet exemple, les fluents b, b' et g sont des landmarks. Si on ne considère pas les fluents de l'état initial, qui par définition sont déjà atteints, la seule relation du graphe des landmarks est donc $b' \rightarrow g$. Donc, le premier métanœud créé utilise b' comme landmark but ($m_1 = \langle I, b', \{a, b, c\} \rangle$). Un plan solution possible, calculé par le planificateur sous-jacent peut être $\rho_1 = \langle \beta \rangle$ pour ce sous-problème (et c'est d'ailleurs le plan optimal). Ensuite, en appliquant l'opérateur nextLM sur le métanœud m_1 on obtient un unique nouveau métanœud puisque qu'il ne reste plus que le fluent g dans le graphe des landmarks :

$$\text{nextLM}(m_1) = \left\{ m_2 = \left\langle I[\beta], g, \{a, b, c, b'\} \right\rangle \right\}$$

On a donc $I[\beta] = \{b, b'\}$, or les deux actions produisant g ont d'une part b' mais aussi soit a', soit b' en précondition. De plus, pour produire ces fluents, il faut appliquer soit l'action α, soit l'action γ. Ce qui implique l'obtention du fluent a ou du fluent c, qui malencontreusement ne sont plus productibles car aucune action

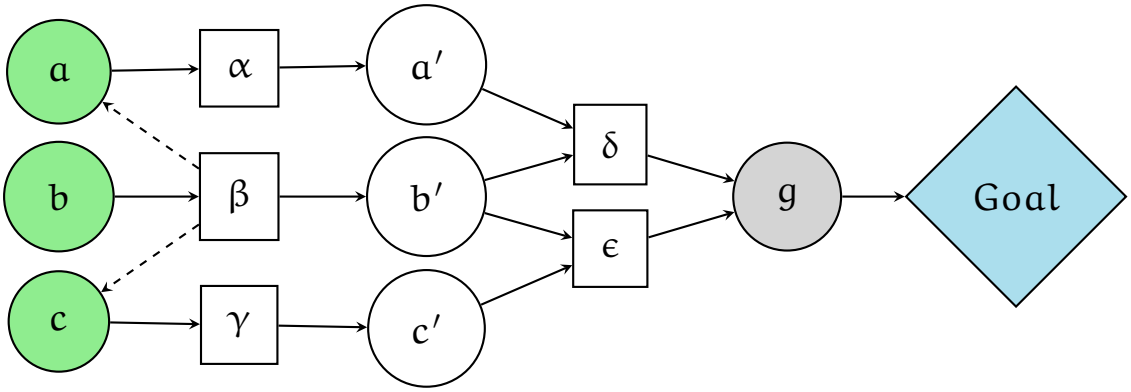


FIGURE 5.3 : Graphe de relations entre les fluents du problème des effets négatifs (Exemple 6). Les cercles sont les fluents, les carrés sont les actions, les flèches représentent les ajouts (pleines) et les suppressions (en pointillées) des atomes par les actions. Les cercles verts représente les fluents de l'état initial. On peut voir directement que les solutions associées à ce problème appliquent soit l'action α, soit l'action γ en premier.

ne permet de les créer. Et ils ne sont plus présents car ils ont été retirés par l'action β. Il est donc impossible d'atteindre le but du métanœud m_2. Donc le problème Π_{m_2} n'a pas de solution, et il n'y a pas d'autres métanœuds générés par l'opérateur nextLM.

À ce stade, l'algorithme LMBFS répond donc que le problème original est insoluble. Or, il existe pourtant des plans solutions tel que $\rho^\star = \langle \alpha, \beta, \delta \rangle$.

Cet exemple précis fait apparaitre un défaut de l'algorithme lorsqu'il fait face à un landmark disjonctif, ici $a' \vee c'$. Plutôt que d'essayer de détecter l'ensemble des disjonctions, nous allons utiliser, en complément de nextLM, d'autres générateurs de métanœuds. Nous essaierons aussi de modifier le comportement du planificateur sous-jacent.

5.3.4 Des « sauts » dans le graphe des landmarks

Afin de pallier les limites de l'opérateur nextLM, nous avons défini un ensemble d'opérateurs. Ceux-ci, utilisés en conjonction avec l'opérateur nextLM, rendent l'algorithme complet.

5.3.4.1 Générateurs de métanœuds *Cut-parents*

Ces opérateurs suppriment les ancêtres des landmarks non-racines. Un premier opérateur va simplement générer un métanœud par chacun des landmarks fils du métanœud développé, indépendamment du fait que leurs landmarks parents aient été réalisés ou non. Pour cela, certains landmarks non atteints sont directement ajoutés à l'ensemble des landmarks déjà atteints. Certaines branches du graphe

de landmarks pourront être coupées, menant ainsi à la création d'ordres totaux focalisés sur différentes branches du graphe.

Définition 36 : *Générateur de métanœuds : Cut-parents*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A} \rangle$ *un métanœud et* Γ *un graphe de landmark associé au problème* Π*.*

$$\text{cutParents}(m) = \begin{cases} \left\{ \left\langle s[\rho], l', A_{tt}(\mathcal{A}, \rho) \cup \mathcal{P}_\Gamma(l') \right\rangle \;\middle|\; l' \in \text{Ch}_\Gamma(l) \right\} & \textit{Si } \Pi_m \textit{ a une solution } \rho \\ \emptyset & \textit{Sinon} \end{cases}$$

Un autre générateur, pouvant être associé à un restart de l'algorithme, génère un métanœud repartant de l'état initial I du problème original, focalisant directement sur un landmark en coupant toute sa branche d'ancêtres.

Définition 37 : *Générateur de métanœuds : Restart cut-parents*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A} \rangle$ *un métanœud et* Γ *un graphe des landmarks associé au problème* Π*.*

$$\text{restartCutParents}(m) = \left\{ \left\langle I, l', (I \cap \mathcal{L}) \cup \mathcal{P}_\Gamma(l') \right\rangle \;\middle|\; l' \in \text{Ch}_\Gamma(l) \right\}$$

Indépendamment du fait que Π_m *possède une solution ou non.*

Un ordre total construit à partir de l'ordre partiel défini par le graphe des landmarks est potentiellement trop restrictif (comme vu dans le contre-exemple 6). En utilisant ces deux opérateurs en complément de nextLM, certains landmarks sont ignorés en essayant d'atteindre directement des landmarks se trouvant plus loin dans le graphe des landmarks. La complétude apportée par ces deux nouveaux générateurs est détaillé section 5.5.1.

Comme nous le discuteront en détail plus tard, il faudra utiliser ces opérateurs judicieusement, voire en dernier recours. En effet, ils agissent comme des *restarts* de l'algorithme, et seront utilisés quand l'algorithme n'a plus d'autres choix dans sa recherche classique (avec l'opérateur nextLM).

5.3.4.2 Générateurs de métanœuds *Delete Landmark*

De manière complètement indépendante des générateurs de type *Cut-parents*, on peut définir un opérateur générique de suppression d'un landmark, les métanœuds étant générés comme si un landmark n'existait pas.

Définition 38 : *Générateur de métanœuds : Delete landmark*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A} \rangle$ *un métanœud et* Γ *un graphe des landmark associé au problème* Π*.*

$$\text{deleteLM}(m) = \left\{ \langle s, l', \mathcal{A} \cup \{l\} \rangle \;\middle|\; l' \in \mathcal{R}\left(\Gamma \setminus (\mathcal{A} \cup \{l\})\right) \right\}$$

Indépendamment du fait que Π_m *possède une solution ou non.*

Cet opérateur « saute » un landmark, et génère un métanœud pour chaque landmark racine, à la manière de l'opérateur nextLM, mais indépendant du fait que Π_m possède une solution ou non. Appliquer récursivement cet opérateur depuis un métanœud initial (ayant l'état initial I comme état de départ) videra le graphe de landmarks pour finalement créer un métanœud équivalent au problème original Π.

Remarque : Les opérateurs *Cut-parents* peuvent être vus comme plusieurs applications de l'opérateur deleteLM.

5.4 Planification sous-jacente

La planification des sous-problèmes peut être théoriquement effectuée par n'importe quel planificateur, utilisé comme une boite noire. Cependant, il y a un échange incessant entre l'algorithme LMBFS et le planificateur sous-jacent. De plus, les problèmes générés sont a priori simples, et devraient être résolus rapidement. Il faut donc un planificateur capable de résoudre les problèmes simples très rapidement, avec peu de pré-traitement. Enfin, il y a, a priori, peu de landmarks dans les sous-problèmes générés, donc les planificateurs basés sur les landmarks ne sont peut-être pas adaptés.

5.5 Vue détaillée de l'algorithme

L'algorithme *Landmark-based Meta-Best First Search* (algorithme 4) est donc un algorithme de type *Best-First Search*, basé sur les métanœuds définis précédemment (cf. définition 31).

Cet algorithme est exécuté sur le problème $\Pi_g = \langle F \cup \{g\}, A \cup \{a_g\}, I, \{g\} \rangle$ où g est un fluent factice, représentant la réalisation du but G. L'action a_g est une action factice permettant la production dudit fluent factice tel que : $a_g = \langle G, \{g\}, \emptyset \rangle$. Donc la production de g implique la résolution du problème Π.

Tout d'abord, les métanœuds associés à chaque landmark racine du graphe Γ sont ajoutés à la liste ouverte $\mathcal{O}$. Ensuite, à chaque itération, le meilleur métanœud m est extrait de la liste ouverte. Un appel est fait au planificateur sous-jacent, demandant la résolution du problème associé Π_m. Si le métanœud n'a pas déjà

Algorithme 4: Landmark-based Meta Best-First Search.

Paramètre(s) : Un problème de planification STRIPS $\Pi = \langle F, A, I, G \rangle$, un graphe de landmarks Γ, une fonction générant les successeurs d'un métanœud succ, h une heuristique et subplanner un planificateur.

Valeur(s) de sortie : Un plan solution (ou $\perp$ si le problème n'admet pas de solution)

1 $\mathcal{O} \leftarrow \emptyset$ `// Liste ouverte`
2 $\mathcal{C} \leftarrow \emptyset$ `// Liste fermée`
3 $\Gamma \leftarrow \Gamma \setminus \{I \cap \mathcal{L}\}$ `// Retrait des fluents de l'état initial`
4 $(\forall l \in \mathcal{R}(\Gamma))\ \mathcal{O} \leftarrow \langle I, l, I \cap \mathcal{L} \rangle$ `// Génération des métanœuds initiaux`
5 **while** $\mathcal{O} \neq \emptyset$ **do**
6 $\quad m \leftarrow \underset{\langle s,l,\mathcal{A}\rangle \in \mathcal{O}}{\text{argmin}}\ h(\text{subplanner}(\Pi_m))$
7 $\quad \mathcal{O} \leftarrow \mathcal{O} \setminus \{m\}$
8 $\quad$ **if** $m \notin \mathcal{C}$ **then**
9 $\quad\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{m\}$
10 $\quad\quad$ **if** $G \subseteq s$ **then** `// but global G trouvé ?`
11 $\quad\quad\quad$ **return** $\text{Plan}(m)$ `// Retrouve plan de I à m`
12 $\quad\quad \mathcal{O} \leftarrow \mathcal{O} \cup \text{succ}(m)$
13 **return** $\perp$

été développé (c'est-à-dire, s'il n'est pas déjà dans la liste fermée), alors on vérifie si le but a déjà été atteint ; et si ce n'est pas le cas, on génère les successeurs du métanœud et on les insère dans la liste ouverte. L'algorithme itère ainsi jusqu'à ce que la liste ouverte soit vide, ou que g ait été atteint.

La fonction « succ » appliquée au métanœud courant (Algorithm 4 ligne 12) calcule l'union d'un ensemble de générateurs de métanœuds. Nous avons en particulier étudié deux ensembles de successeurs :

Définition 39 : *Union de générateurs de métanœuds* succDel

On note succDel *la fonction réalisant l'union des générateurs* nextLM *et* deleteLM, *définie telle que :*

$$\text{succDel} : \begin{array}{ccc} 2^F \times \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & 2^F \times \mathcal{L} \times 2^{\mathcal{L}} \\ m & \longmapsto & \text{nextLM}(m) \cup \text{deleteLM}(m) \end{array}$$

Définition 40 : *Union de générateurs de métanœuds* succCut

On note succCut *la fonction réalisant l'union des générateurs*nextLM, cutParents *et* restartCutParents, *définie telle que :*

$$\text{succCut} : \left| \begin{array}{ccc} 2^F \times \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & 2^F \times \mathcal{L} \times 2^{\mathcal{L}} \\ m & \longmapsto & \text{nextLM}(m) \cup \text{cutParents}(m) \cup \text{restartCutParents}(m) \end{array} \right.$$

L'opérateur nextLM est le cœur de l'algorihme : il permet de générer des séquences complètes de landmarks. Pour assurer la complétude de l'algorithme, les autres opérateurs viennent supporter nextLM. Si on applique ces opérateurs à l'expansion de chaque métanœud, on obtiendra finalement : $m = \langle I, g, \mathcal{L}\{g\}\rangle$, et son problème associé Π_m est en fait le problème orignal Π_g.

On peut aussi noter qu'ici, l'algorithme lance le planificateur sous-jacent pour chaque métanœud de la liste ouverte, il faut évidemment garder en mémoire les résultats de chaque exécution pour éviter de lancer plusieurs fois la planification d'un métanœud.

5.5.1 Correction et complétude de l'algorithme

Théorème 7 : *Correction & Complétude de l'algorithme LMBFS*

L'algorithme Landmark-based Meta Best-First Search, *associé aux opérateurs* succCut *ou* succDel, *est* correct et complet *si le planificateur sous-jacent est* correct et complet.

Démonstration de la *correction* de l'algorithme LMBFS :

Pour prouver la correction cet algorithme, il faut d'abord prouver qu'il existe des plans partiels corrects menant aux états initiaux de tous les métanœuds qui sont générés. C'est-à-dire qu'ils respectent la relation $s = I[\rho]$, où s est l'état initial d'un métanœud, et ρ est un plan.

Supposons que le planificateur sous-jacent soit correct et complet.

(1) Soit $m = \langle s, l, \mathcal{A}\rangle$, un métanœud. Si le problème Π_m admet un plan solution ρ alors :

$$(\forall m' = \langle s', l', \mathcal{A}'\rangle \in \text{nextLM}(m))\ s' = s[\rho]$$

où s' est l'état but atteint lors de la résolution de Π_m. Or, par récurrence immédiate à partir de n'importe quel métanœud inital $m_1 = \langle I, l_1, \mathcal{A}_1\rangle$, et pour n'importe quelle chaine de métanœuds $(m_1, \ldots, m_n)$ générés par l'opérateur nextLM, on a $s_n = I[\rho_1 \circ \ldots \circ \rho_{n-1}]$.

(2) Idem pour le générateur cutParents.

(3) Pour tout métanœud m, l'opérateur restartCutParents crée des métanœuds m_i ayant I comme état initial.

(4) Pour tout métanœud $m = \langle s, l, \mathcal{A}\rangle$, l'opérateur restartCutParents crée des métanœuds m_i ayant s comme état initial. Donc, d'après **(1) (2)** et **(3)**, il existe un plan ρ tel que $s = I[\rho]$.

Donc d'après **(1) (2) (3)** et **(4)**, tous les générateurs de métanœuds créent des métanœuds respectant la relation $I = s[\rho]$ où ρ est la concaténation de plans partiels valides, solutions des métanœuds ayant conduit à la génération de $m = \langle s, l, \mathcal{A} \rangle$.

Finalement, si un métanœud $m = \langle s, g, \mathcal{A} \rangle$ ayant le but global du problème comme but intermédiaire, et que le problème Π_m a une solution ρ, alors, on sait qu'il existe un plan ρ' tel que $I[\rho'] = s$ et que $G \subseteq s[\rho]$. Donc $G \subseteq I[\rho' \circ \rho]$. Donc $\rho' \circ \rho$ est un plan solution de Π. ■

Démonstration de la *complétude* de l'algorithme LMBFS :

La complétude provient du fait que les opérateurs restartCutParents et deleteLM sont appelés systématiquement à chaque développement de métanœud. On a ainsi nécessairement une branche dans l'arbre de recherche consistant uniquement à l'application d'un de ces deux opérateurs. Or ils gardent l'état initial du précédent métanœud et ajoutent les landmarks directement dans l'ensemble des landmarks atteints $\mathcal{A}$. Donc, au bout de ces branches de l'arbre de recherche, on obtient un métanœud final, où tous les landmarks $\mathcal{L} \setminus \{g\}$ ont été mis dans $\mathcal{A}$, avec l'état initial du problème original : $m = \langle I, g, \mathcal{L} \setminus \{g\} \rangle$ correspondant au problème original $\Pi_m = \Pi_g$. Donc la complétude provient de la complétude du planificateur sous-jacent.

Après la génération des métanœuds initiaux, la liste ouverte contient :

$$\left\{ \langle I, l, I \cap \mathcal{L} \rangle \;\middle|\; \forall l \in \mathcal{R}(\Gamma) \right\}$$

Si Γ ne contient que g, alors le seul métanœud créé est $\langle I, g, \emptyset \rangle$ donc le problème associé correspond au problème global Π_g.

Sinon, si on prend n'importe quel métanœud initial, noté $m = \langle I, l, \mathcal{A} \rangle$, et qu'on applique l'opérateur restartCutParents à m, on obtient l'ensemble :

$$\left\{ \langle I, l', \mathcal{A} \cup \{l'\} \rangle \;\middle|\; \forall l' \in \mathcal{R}((\mathcal{L} \setminus \mathcal{A}, \rightarrow)) \right\}$$

Et si on applique récursivement cette méthode jusqu'à vider le graphe des landmarks, c'est-à-dire $\mathcal{R}((\mathcal{L} \setminus \mathcal{A}', \rightarrow)) = g$. Ce qui arrive nécessairement puisque g est l'unique fils de tous les nœuds du graphe (par transitivité) donc si on retire tour à tour les nœuds n'ayant plus de nœuds parents (les racines), alors nécessairement, g sera le dernier nœud. Donc le dernier métanœud généré par application récursive de restartCutParents sera $m_g = \langle i, g, \mathcal{L} \setminus \{g\} \rangle$ dont le problème associé Π_{m_g} est exactement le problème global Π_g.

Idem pour l'opérateur deleteLM.

Donc, lors du développement exhaustifs de tous les métanœuds de l'espace de recherche, le métanœud m_g sera ajouté dans la liste ouverte. Donc, si le problème n'est pas résolu avant par une autre chaine de métanœud, alors ce métanœud, représentant le problème initial, sera envoyé par le planificateur sous-jacent. ■

5.5.2 Duplication des metanœuds

Les métanœuds, en particulier ceux issus des générateurs de type *Cut-parents*, peuvent être créés plusieurs fois lors de l'exécution de l'algorithme LMBFS. Dans le cas où un métanœud existe déjà dans la liste ouverte, ou dans la liste fermée, il

est inutile de le dupliquer. Deux métanœuds sont équivalents si leurs états initiaux sont les mêmes, si leurs landmarks but sont équivalents et enfin si leurs ensembles de landmarks déjà atteints sont identiques.

On peut cependant noter, qu'il faut aussi vérifier la longueur du plan ayant amené à la création de deux métanœuds équivalents. Si le nouveau métanœud a un plan moins coûteux, il faut mettre à jour celui qui se trouve dans la liste ouverte ou fermée. Dans le cas où une solution est atteignable depuis ce nouveau métanœud, ceci permet d'avoir une solution de meilleure qualité.

5.6 Conclusion

Nous venons de voir le cadre théorique de l'algorithme Landmark-based Meta Best-First Search (LMBFS). Cet algorithme explore les différents ordres totaux générables à partir d'un graphe de landmarks. Nous avons pu voir que cette exploration peut être vue sous la forme d'un arbre où chaque nœud forme un sous-problème de planification focalisé sur un landmark. Maintenant que l'idée générale de l'algorithme a été décrite, nous allons pouvoir nous concentrer sur la façon dont l'algorithme a été implémenté, et surtout quelles techniques permettent de le rendre efficace.

6

CONSTRUCTION D'UN PLANIFICATEUR EFFICACE

Nous allons commencer par présenter une implémentation de l'algorithme LMBFS en l'agrémentant d'une heuristique de sélection du prochain métanœud à développer. Ensuite nous présenterons différentes techniques, comme l'évaluation différée, appliquées à l'algorithme LMBFS. Chacune de ces techniques a été rigoureusement testée pour décider de leur conservation pour la version définitive de l'algorithme.

Sommaire

6.1 Dispositif expérimental

6.1.1 Ensemble des problèmes

Avant de détailler les différentes versions que nous avons développées et testées sur l'algorithme LMBFS, présentons rapidement le dispositif expérimental que nous avons utilisé. Les différentes méthodes de planification sont régulièrement mises en comparaison via les différentes compétitions internationales de planification (IPC) [28]. Ces compétitions donnent aussi l'occasion de définir des ensembles de benchmarks. Ils sont censés être représentatifs d'un large échantillon de problèmes de planification et mettre en avant les nouvelles possibilités du langage de représentation PDDL [McDermott et al., 1998] ; [Fox et Long, 2003] ; [Edelkamp et Hoffmann, 2004] ; [Gerevini et Long, 2006] qui évolue au fil de ces compétitions. Nous nous baserons sur cet ensemble de benchmarks pour évaluer notre algorithme.

En particulier nous utiliserons les domaines des deux dernières compétitions [29, 30]. Nous avons sélectionné seulement les domaines que notre planificateur est capables de lire. En particulier, l'analyseur grammatical n'est pas capable de lire les domaines utilisant les préconditions négatives ou les quantificateurs.

Les 13 domaines considérés pour l'IPC 7 représentent un total 260 problèmes (20 problèmes par domaines). Ces domaines sont :

- barman
- elevators
- floortile
- nomystery
- openstacks
- parcprinter
- parking
- pegsol
- scanalyzer
- sokoban
- transport
- visitall
- woodworking

Les 9 domaines considérés pour l'IPC 7 représentent un total 180 problèmes (20 problèmes par domaines). Ces domaines sont :

28. IPC — `http://ipc.icaps-conference.org`
29. IPC 7 (2011) — `http://www.plg.inf.uc3m.es/ipc2011-deterministic`
30. IPC 8 (2014) — `http://helios.hud.ac.uk/scommv/IPC-14`

- barman
- childsnack
- floortile
- ged
- hiking
- parking
- thoughtful
- transport
- visitall

6.1.2 Métriques

Ayant comme ambition de pouvoir présenter des résultats comparables à ceux effectués lors de ces compétitions, nous avons mis en place un dispositif proche de celles-ci, et nous utilisons les mêmes métriques. De la même manière que pour la partie *agile* de la compétition, nous nous intéresserons au temps de résolution du problème, en particulier grâce à la métrique :

$$m_{\text{time}} : \left| \begin{array}{ccc} \mathbb{R}^+ \times \mathbb{R}^+ & \longrightarrow & \mathbb{R} \\ (T, T^\star) & \longmapsto & \dfrac{1}{1 + \log_{10} \dfrac{T}{T^\star}} \end{array} \right.$$

où T est le temps utilisé par un planificateur pour trouver une première solution au problème, et $T^\star$ est le meilleur de ces temps parmi tous les planificateurs considérés. Les temps de calculs sont arrondis à la seconde, et lorsque qu'ils sont inférieurs à 1, ils sont ramenés à 1s.

Pour juger de la qualité des solutions trouvées, les planificateurs seront lancés en mode *anytime* et seule la meilleure solution trouvée pendant le temps alloué sera considérée grâce à la métrique :

$$m_{\text{quality}} : \left| \begin{array}{ccc} \mathbb{R}^+ \times \mathbb{R}^+ & \longrightarrow & \mathbb{R}^+ \\ (Q, Q^\star) & \longmapsto & \dfrac{Q^\star}{Q} \end{array} \right.$$

où Q est le coût de la meilleure solution trouvée par un planificateur et

$Q^\star$ le coût de la meilleure des solutions trouvées par tous les planificateurs. Dans les deux cas, en cas d'échec de résolution du problème, un score de 0 est donné. De même, ces métriques donnent toujours un score compris entre 0 et 1.

Enfin, un dernier moyen de caractérisation sera utilisé pour pallier le fait que ces deux métriques peuvent perdre l'information sur le nombre de problèmes résolus. En effet si un planificateur résout beaucoup de problèmes simples très rapidement, mais n'évolue plus sur la durée, il aura un score très haut comparé

aux autres planificateurs [31]. La métrique m_{time} échoue à montrer l'évolution sur la durée. C'est pourquoi nous nous intéressons aussi au nombre de problèmes résolus en fonction du temps.

6.1.3 Matériel

Toutes les expérimentations ont été effectués en utilisant un cœur du processeur Intel X5670 (2.96Ghz) avec une limite de 24Go de mémoire. Le temps limite pour la résolution des problèmes a été fixé à 10 minutes ou 30 minutes CPU.

6.2 Graphe de landmarks

Nous nous sommes principalement intéressés à la méthode présentée par [Zhu et Givan, 2003] (cf. section 3.4.1) permettant l'extraction rapide d'un graphe de landmarks. En particulier parce que l'utilisation de landmarks en tant que sous-buts implique l'utilisation de landmarks causaux. En effet, il n'est pas utile de chercher directement des landmarks qui peuvent être vus comme des effets de bords de la recherche du but global.

Comme amélioration, on peut simplement noter qu'un landmark nécessaire à l'établissement d'un autre landmark implique un ordre naturel entre eux [Keyder, Richter et Helmert, 2010]. Lors de la génération du graphe des landmarks, il suffit de regarder si un landmark l se trouve dans les labels d'un autre. Dans ce cas-là, on peut conclure qu'il existe une relation naturelle entre eux : $l \rightarrow_n l'$.

D'autres part, il est important de préciser que nous calculons le graphe de landmarks à l'initialisation de l'algorithme, à partir de l'état initial du problème global Π. Ainsi, nous nous intéressons seulement à la première apparition de chaque landmarks.

6.3 Planification sous-jacente

Dans notre implémentation, nous avons choisi d'utiliser YAHSP [Vidal, 2011] parce que nous pensons que les problèmes générés seront souvent résolus grâce à quelques lookaheads. De plus, il est directement disponible en tant que bibliothèque, évitant ainsi les étapes de *parsing* et de pré-traitement à chaque appel de résolution de sous-problèmes. Enfin, il a déjà été utilisé en tant que planificateur sous-jacent [Bibaï et al., 2010].

De plus, la dernière compétition internationale de planification (cf. section 4.7) nous permet de confirmer ce choix puisque le planificateur YAHSP a fini premier de la catégorie *agile*.

31. Le problème vient en partie aussi du choix des instances.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
$h^{\mathcal{L}_{left}}$	0	9	3	5	0	15	0
h^{ff}	0	0	1	5	0	0	0
h^{add}	0	0	2	3	0	0	0

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
$h^{\mathcal{L}_{left}}$	20	17	10	14	4	6	**103**
h^{ff}	18	9	8	0	0	1	**42**
h^{add}	17	10	6	0	0	1	**39**

TABLEAU 6.1: Nombre d'instances résolues par domaine en utilisant les heuristiques $h^{\mathcal{L}_{left}}$, h^{ff}, et h^{add} (opérateur succDel dans tous les cas)

6.4 Heuristiques de selection des sous-problèmes

Pour améliorer les performances de l'algorithme, on doit ouvrir à chaque itération le plus prometteur des métanœuds. Dans les algorithmes de type *Best First Search*, on utilise donc des heuristiques. Nous nous sommes intéressés à des heuristiques classiques tel que h^{add} ou h^{ff} (cf. section 2.2).

Nous utilisons aussi une heuristique permettant à l'algrithme LMBFS de se comporter comme un algorithme de recherche en profondeur d'abord. Pour cela, on compte simplement le nombre de landmarks restant à atteindre :

Définition 41 : *Heuristique* $h^{\mathcal{L}_{left}}$

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\Gamma = \langle \mathcal{L}, \rightarrow \rangle$ *un graphe de landmarks :*

$$h^{\mathcal{L}_{left}} : \left| \begin{array}{rcl} 2^F \times \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & \mathbb{N} \\ m = \langle s, l, \mathcal{A} \rangle & \longmapsto & \left| \mathcal{L} \setminus \mathcal{A} \right| \end{array} \right.$$

Cette heuristique est très proche de l'heuristique h^{LM} mais ne prend pas en compte les landmarks à atteindre à nouveau. En effet, comme la recherche de landmarks que nous utilisons actuellement ne dérive aucun ordre glouton impératif, on peut simplifier l'heuristique. Cependant, à part l'acyclicité du graphe de landmarks, il n'y a aucune limite théorique empêchant l'utilisation de l'heuristique h^{LM} avec une autre méthode de génération de landmarks.

Le tableau 6.1 présente le nombre de problèmes résolus par domaine en utilisant les heuristiques $h^{\mathcal{L}_{left}}$, h^{ff}, et h^{add}. Le premier constat que l'on peut faire est que l'heuristique $h^{\mathcal{L}_{left}}$ est bien meilleure que les deux autres, sur l'ensemble des domaines, mais aussi domaine par domaine. Ensuite, on peut s'inquiéter du faible nombre d'instances résolues dans l'absolu, qui est de moins de 50%.

Pour cette première expérimentation, nous nous sommes seulement intéressés à l'opérateur succDel. En effet, dans cette configuration, l'utilisation du générateur restartCutParents conduit notre algorithme à faire des coupes dans le graphe des

landmarks trop rapidement, en particulier en utilisant les heuristiques basées sur les états h^{add} et h^{ff}, et reviendrait à finalement exécuter YAHSP directement sur le problème complet. Prenons en exemple la figure 6.1 qui considère, pour l'exemple, que la valeur heuristique associée à l'état s_2 est meilleure que celle de s_1. Le premier métanœud étendu serait donc celui amenant à s_2, permettant donc la création du métanœud (I, e) qui correspond au problème initial (puisque ce problème ne possède qu'un seul fluent dans le but global). Ainsi, on peut voir que le générateur restartCutParents peut directement amener à la résolution du problème complet par YAHSP s'il est utilisé sans discernement.

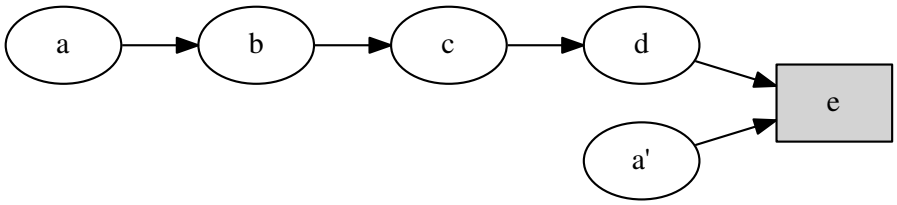
6.5 Génération paresseuse des métanœuds

Pour forcer l'exécution d'ordre total complet, on peut choisir de préférer les métanœuds générés par l'opérateur nextLM par rapport aux métanœuds générés par d'autres opérateurs permettant ainsi de focaliser la recherche sur les séquences complètes de landmarks. De cette manière, le traitement des autres métanœuds (générant des séquences dégradées) est différé jusqu'à ce qu'il n'y ait plus d'autre choix. Cette méthode est inspirée des méthodes basées sur les *preferred operators* (cf. section 2.4.2).

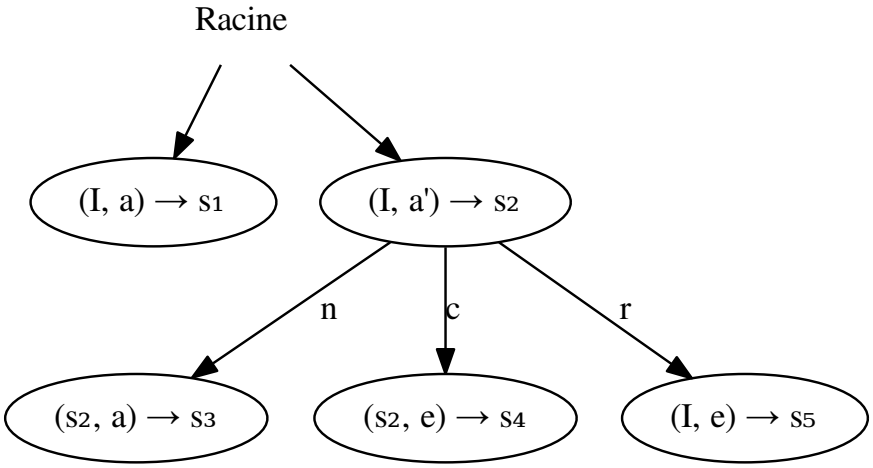
De plus, les opérateurs de génération de landmarks permettant de rendre l'algorithme complet cutParents, restartCutParents et deleteLM (section 5.3) génèrent beaucoup de métanœuds supplémentaires, induisant ainsi des ralentissements de l'exécution de l'algorithme. En particulier, l'opérateur deleteLM double le facteur de branchement. De plus, certains de ces opérateurs agissent comme des redémarrages de l'algorithme, et doivent donc être utilisés avec parcimonie, notamment quand l'algorithme ne trouve plus de possibilité d'atteindre le but dans sa recherche classique.

Pour pallier ce problème, on peut choisir de ne générer les métanœuds de type deleteLM que lorsque la liste ouverte est vide. Pour implémenter cela, nous avons choisi d'ajouter une liste ouverte secondaire : quand un métanœud est inséré dans la liste fermée, il est aussi inséré dans cette seconde liste ouverte. Puis quand la liste ouverte principale est vide, il suffit de retirer un métanœud de cette liste ouverte secondaire, et de générer ses fils via deleteLM. L'heuristique ordonnant les métanœuds de la liste ouverte secondaire est la même que celle utilisée pour la liste principale, assurant ainsi le même comportement que sans génération paresseuse. Et on peut évidemment faire de la même manière avec les opérateurs cutParents et restartCutParents. On peut faire le parallèle avec les techniques de liste ouverte multiple utilisant différentes heuristiques.

Le tableau 6.2 présente le nombre de problèmes résolus par domaine en utilisant les heuristiques $h^{\mathcal{L}_{left}}$, h^{ff}, et h^{add} et la génération paresseuse des métanœuds. On peut tout d'abord constater que la génération paresseuse augmente bel et bien le nombre d'instances résolues lorsque l'algorithme utilise l'heuristique $h^{\mathcal{L}_{left}}$. Cependant, ce nombre diminue en utilisant les deux autres heuristiques.



(a) Graphe de landmarks.



(b) Une exploration possible de l'algorithme LMBFS utilisant les générateurs succCut. Chaque nœud est composé d'un couple (s, l) où s est l'état initial du métanœud et l le but, et de l'état atteint lors de la planification du sous-problème associé. L'étiquette des nœuds représente le générateur de métanœud : n pour nextLM, c pour cutParents et r pour restartCutParents.

FIGURE 6.1 : Exemple d'arbre d'exploration de l'algorithme LMBFS sans discernement entre les différents générateurs de métanœuds.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
succCut, $h^{\mathcal{L}_{\mathtt{left}}}$	6	20	6	5	7	5	0
succDel, $h^{\mathcal{L}_{\mathtt{left}}}$	6	20	6	5	7	5	0
succCut, h^{ff}	0	0	3	3	0	0	0
succCut, h^{add}	0	0	3	3	0	0	0
succDel, h^{ff}	0	0	3	3	0	0	0
succDel, h^{add}	0	0	3	3	0	0	0
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
succCut, $h^{\mathcal{L}_{\mathtt{left}}}$	19	17	9	20	20	2	**136**
succDel, $h^{\mathcal{L}_{\mathtt{left}}}$	19	17	4	20	20	2	**131**
succCut, h^{ff}	16	4	8	0	0	1	**35**
succCut, h^{add}	16	4	8	0	0	1	**35**
succDel, h^{ff}	16	4	3	0	0	1	**32**
succDel, h^{add}	16	4	5	0	0	1	**30**

TABLEAU 6.2: Nombre d'instances de problèmes résolues par domaine en utilisant les heuristiques $h^{\mathcal{L}_{\mathtt{left}}}$, h^{ff}, et h^{add} en combinaison avec les générateurs succCut et succDel. La génération paresseuse est activée dans tous les cas.

D'autre part, on peut voir que l'influence du choix entre les opérateurs succCut ou succDel n'affecte que très peu le fait de résoudre un problème ou non. Le domaine *sokoban* est le seul dont le nombre d'instances résolues augmente en utilisant succCut, ce qui peut s'expliquer par le fort nombre d'impasses présentes dans ce domaine.

6.6 Sélection d'actions

Lors de la génération des sous-problèmes liés à un métanœud, nous avons défini une fonction ops_γ sélectionnant l'ensemble des actions d'un sous problème. Pour l'instant nous avons défini cette fonction telle que renvoyant simplement l'ensemble des actions du problème global A. Pour rendre les sous-problèmes encore plus simples à résoudre, nous pouvons définir une fonction enlevant de A l'ensemble des actions ajoutant un landmark racine autre que le landmark objectif du sous-problème. Cette fonction permet de focaliser la recherche sur le landmark l, en interdisant la production d'autres landmarks racines.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
LMBFS,ops^r_Γ	14.00	20.00	20.00	4.00	20.00	2.00	0.00
LMBFS	14.00	20.00	20.00	5.00	16.00	2.00	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
LMBFS,ops^r_Γ	20.00	17.00	5.00	20.00	20.00	1.00	**163.00**
LMBFS	19.00	17.00	5.00	20.00	20.00	1.00	**159.00**

TABLEAU 6.3: Nombre d'instances de problèmes résolues par domaine en utilisant la sélection d'action avec ops^r_Γ ou non (avec l'heuristique $h^{\mathcal{L}_{left}}$ et l'opérateur succDel)

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
LMBFS,ops^r_Γ	13.03	19.80	20.00	4.00	20.00	1.82	0.00
LMBFS	13.34	20.00	20.00	4.45	11.94	1.97	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
LMBFS,ops^r_Γ	19.10	17.00	4.80	19.51	16.62	1.00	**156.67**
LMBFS	16.03	17.00	4.75	19.77	20.00	1.00	**150.26**

TABLEAU 6.4: Impact de la fonction de sélection d'action ops^r_Γ sur le temps passé à chercher la première solution (somme de la métrique m_{time}). Pour cette expérience, LMBFS utilise l'heuristique $h^{\mathcal{L}_{left}}$ et le générateur succDel.

Définition 42 : ***Retrait des actions ajoutant d'autre landmarks racines***

Soient $\Pi = \langle F, A, I, G\rangle$ *un problème de planification,* $m = \langle s, l, \mathcal{A}, h, \rho\rangle$ *un métanœud et* $\Gamma = \langle \mathcal{L}, \rightarrow\rangle$ *un graphe de landmarks associé au problème* Π.
On note $ops^r_\Gamma(l, \mathcal{A})$*, la fonction de sélection des actions permettant l'ajout des landmarks déjà atteints* $\mathcal{A}$ *et du landmark but* l.

$$ops^r_\Gamma : \left| \begin{array}{rcl} \mathcal{L} \times 2^{\mathcal{L}} & \longrightarrow & 2^A \\ (l, \mathcal{A}) & \longmapsto & \left\{ a \in A \mid \big(l \in add(a)\big) \vee \big(add(a) \cap \mathcal{R}(\Gamma \setminus \mathcal{A}) = \emptyset\big) \right\} \end{array} \right.$$

Cette fonction permet d'augmenter légèrement le nombre d'instances résolues (Tableau 6.3), mais ceci quasi-exclusivement pour le domaine *openstack*. De même, on peut constater une amélioration du temps passé pour trouver une solution pour ce domaine.

Cependant on peut noter une baisse des deux métriques m_{time} et $m_{quality}$ dans le domaine *visitall* alors que les deux méthodes ont résolu tous les problèmes. *Visitall* est un domaine composé d'une grille $n \times n$ où un agent doit visiter toutes

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
LMBFS	13.67	20.00	20.00	5.00	16.00	2.00	0.00
LMBFS,$\text{ops}^{\text{r}}_{\Gamma}$	13.97	19.47	19.95	4.00	20.00	2.00	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
LMBFS	18.75	16.71	4.88	20.00	20.00	1.00	**158.00**
LMBFS,$\text{ops}^{\text{r}}_{\Gamma}$	19.26	16.50	5.00	20.00	13.10	1.00	**154.25**

TABLEAU 6.5: Impact de la fonction de sélection d'action $\text{ops}^{\text{r}}_{\Gamma}$ sur la qualité de la meilleure solution trouvée (somme de la métrique $\text{m}_{\text{quality}}$). Pour cette expérience, LMBFS utilise l'heuristique $\text{h}^{\mathcal{L}_{\text{left}}}$ et le générateur succDel.

les cases. Il n'y a pas de landmark dans ce problème, à part l'ensemble des cases qui composent le but[32]. De plus, il n'existe aucun ordre entre ces landmarks. Dans le cas où les sous-problèmes disposent de toutes les actions, le planificateur sous-jacent n'a aucun problème pour trouver un chemin amenant à la case but. Par contre, dans le cas où $\text{ops}^{\text{r}}_{\Gamma}$ est utilisé, si le but d'un sous-problème n'est pas directement adjacent à la position du robot[33] alors le sous-problème est impossible. Ainsi, LMBFS est obligé de tester beaucoup de sous-problèmes avant de trouver un but atteignable, ralentissant ainsi la recherche.

On peut cependant se demander pourquoi la qualité est impactée. Bien qu'explorant de proche en proche les cases, la version avec restriction d'actions peut quand même effectuer des trajets inutiles (Figure 6.2(b)), voire des allers-retours : elle autorise l'exploration à seulement une case *non-explorée* par rapport à la case courante. De plus aucune des versions n'est guidée pour trouver une solution proche de l'optimal (Figure 6.2(a)). Enfin, LMBFS sans restriction d'action peut explorer différents ordres totaux entièrement, améliorant petit à petit la qualité de la solution, tandis que la version interdisant certaines actions sera ralentie dans le nombre de ses solutions trouvées (puisqu'elle ne pourra que rarement dérouler entièrement des ordres totaux).

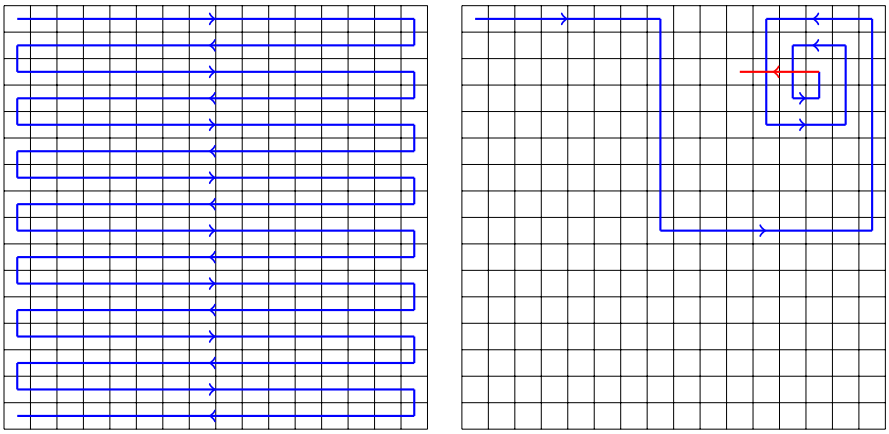
Ainsi, bien qu'améliorant le nombre de solutions trouvées, cette méthode ne sera pas utilisée à cause du problème soulevé dans le domaine *visitall* et probablement présent dans d'autres domaines.

6.7 Évaluation différée des sous-problèmes

Contrairement aux algorithmes de recherche classiques dans les espaces d'états, l'ouverture d'un nœud est très lente. En effet, pour chacun des fils générés, il faut résoudre un problème de planification, puis évaluer heuristiquement l'état

32. En fait, il existe un landmark disjonctif de cardinal deux à quatre, composé de la position de robot à une case de la grille dans toutes les directions.

33. Celui de l'état initial du métanœud courant.



(a) Exploration complète optimale (b) Début d'exploration sous-optimale

FIGURE 6.2 : Différentes solutions à un problème du domaine *visitall*.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
différé	14	20	20	5	16	2	0
immédiat	6	20	6	5	7	5	0
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
différé	19	17	5	20	20	1	**159**
immédiat	19	17	4	20	20	2	**131**

TABLEAU 6.6: Nombre d'instances résolues par domaine en utilisant l'évaluation différée ou non (avec l'heuristique $h^{\mathcal{L}_{\text{left}}}$ et l'opérateur succDel)

atteint. Évaluer complètement tous les fils est effectivement une bonne idée en recherche optimale puisque ceci permet, grâce à une bonne heuristique, de couper toute une partie de l'espace de recherche. Cependant, lors d'une recherche plus gloutonne, et surtout si l'évaluation d'un nœud est coûteuse, on peut privilégier l'exploration en profondeur. Pour cela, on peut appliquer l'évaluation heuristique différée [HELMERT, 2006] ; [RICHTER et HELMERT, 2009] où, lors du développement d'un nœud, on choisit d'utiliser l'évaluation heuristique du parent. Dans notre cas, on utilisera donc la valeur heuristique calculée sur l'état but du métanœud parent. Ainsi, la résolution des sous-problèmes ne se fait plus lors de la génération d'un métanœud, mais lorsque qu'il sort de la liste ouverte.

Ainsi, le couple $h^{\mathcal{L}_{\text{left}}}$ avec l'évaluation différée permet une exploration rapide d'ordres totaux puisque l'algorithme n'aura à résoudre qu'un seul sous-problème à chaque développement d'un nœud.

Comme le montre le tableau 6.6, l'évaluation différée permet d'augmenter sensiblement le nombre d'instances résolues. De plus, la figure 6.3 nous montre

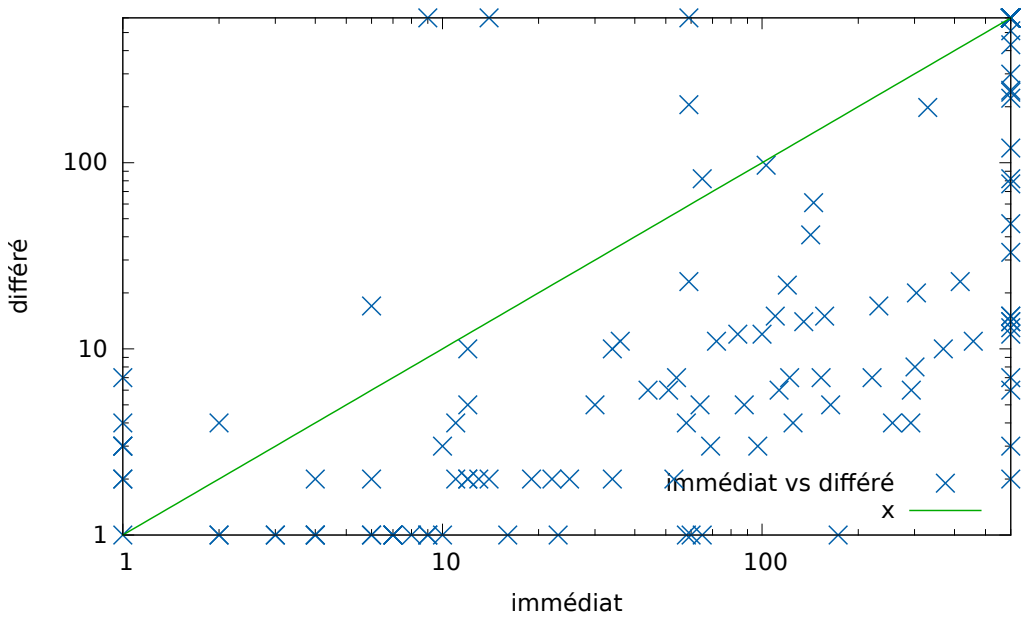


FIGURE 6.3 : Efficacité de l'évaluation différée des métanœuds : temps en secondes pour trouver la première solution. Chaque point représente une instance, l'axe x représente le temps de résolution en utilisant l'évaluation immédiate et l'axe y représente celui de l'évaluation différée. La droite verte représente la fonction linéaire $y = x$. Les instances sur les axes $x = 600$ et $y = 600$ n'ont été résolues que par une seule méthode. Pour cette expérience, LMBFS utilise l'heuristique $h^{\mathcal{L}_{\text{left}}}$ et le générateur succDel.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
différé	13.46	20.00	20.00	5.00	16.00	2.00	0.00
immédiat	4.77	10.67	2.71	3.84	2.66	4.65	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
différé	16.94	16.45	4.91	20.00	20.00	1.00	**155.75**
immédiat	14.28	16.08	3.72	11.77	11.38	2.00	**88.52**

TABLEAU 6.7: Impact de l'évaluation différée des sous-problèmes sur le temps passé à chercher la première solution (somme de la métrique m_{time}). Pour cette expérience, LMBFS utilise l'heuristique $h^{\mathcal{L}_{\text{left}}}$ et le générateur succDel.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
YAHSP3	8.00	20.00	6.00	7.00	20.00	20.00	10.00
LMBFS	15.00	20.00	20.00	6.00	19.00	2.00	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
YAHSP3	20.00	18.00	19.00	20.00	20.00	16.00	**204.00**
LMBFS	19.00	17.00	5.00	20.00	20.00	1.00	**164.00**

TABLEAU 6.8: Nombre d'instances de problèmes résolues par domaine.

clairement une réduction du temps pour trouver la première solution dans la plupart des cas, ce qui est confirmé par le tableau 6.7.

6.8 Première comparaison avec le planificateur sous-jacent

À partir d'ici, nous noterons LMBFS l'algorithme utilisant le générateur succDel et l'heuristique $h^{\mathcal{L}_{left}}$.

Comme nous pouvons l'observer avec le tableau 6.8, LMBFS n'est pas aussi performant que YAHSP3 sur l'ensemble des domaines de l'IPC 7. Cependant on observe une amélioration sur les domaines *barman* et *floortile*. Les faiblesses de LMBFS peuvent en partie s'expliquer par la sur-focalisation de LMBFS sur les landmarks et non sur le but global du problème. En effet, la recherche d'un landmark peut amener dans une mauvaise direction qui, dans le meilleur des cas, peut être inversée mais peut aussi mener à une impasse. Par exemple, dans le domaine *sokoban* [34], se focaliser uniquement sur le placement d'une pierre en négligeant complètement les autres, peut amener à les bloquer et ainsi s'interdire la possibilité de résoudre le problème. LMBFS règle ce problème en essayant tous les ordres possibles, et éventuellement en supprimant certains, voir tous les landmarks. On observe des domaines où ceci marche bien, typiquement le domaine *barman*.

Cependant, cette recherche exhaustive de tous les ordres est parfois trop longue, en particulier parce que le nombre d'ordres totaux peut être grand. Par exemple, si on prend un problème avec n buts, aucun autre landmark, et aucun ordre entre ces buts, alors le nombre d'ordres totaux est de $n!$: il s'agit du nombre d'arrangements sans répétition de n éléments choisis parmi n éléments, c'est-à-dire : A_n^n.

34. Il s'agit d'un domaine où un joueur se déplace dans un labyrinthe en deux dimensions où il doit déplacer des pierres vers des cases buts en les poussant. Si une pierre se retrouve dans un coin, elle ne peut plus être déplacé.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
YAHSP3	8.00	13.57	5.61	7.00	20.00	20.00	10.00
LMBFS	12.24	19.94	19.98	5.96	11.92	2.00	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
YAHSP3	18.75	11.39	19.00	17.60	19.71	16.00	**186.62**
LMBFS	18.26	16.46	3.63	18.37	18.26	0.69	**147.71**

TABLEAU 6.9: Comparaisons de la somme de la métrique qualité du meilleur plan trouvé par domaine.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
$LMBFS_g$	20.00	20.00	20.00	4.00	19.00	15.00	13.00
LMBFS	15.00	20.00	20.00	6.00	19.00	2.00	0.00
	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
$LMBFS_g$	19.00	17.00	8.00	20.00	20.00	15.00	**210.00**
LMBFS	19.00	17.00	5.00	20.00	20.00	1.00	**164.00**

TABLEAU 6.10: Nombre d'instances de problèmes résolues par domaine.

6.9 Double focalisation du planificateur sous-jacent

Afin de pallier ce problème de focalisation, nous avons choisi de modifier le planificateur sous-jacent en lui faisant tester l'accessibilité du but à chaque itération de son algorithme principal. Pour cela, à chaque fois que YAHSP développe un nœud, en plus d'évaluer heuristiquement ce nœud vers le landmark but courant, on ajoute une évaluation de h^1 vers le but global du problème. Si le but global n'est plus atteignable, alors il est abandonné. Cette version de l'algorithme est notée $LMBFS_g$.

Cette modification implique que le planificateur n'est plus uniquement une boîte noire. Cependant la modification est suffisamment simple pour être facilement adaptable à d'autres planificateurs, et en particulier directement applicable à tous ceux basés sur la recherche dans l'espace d'états. Par ailleurs, on peut rappeler que ce test n'est pas nécessairement totalement fiable. Dans le cas de l'heuristique h^1, elle ne détecte pas dans tous les cas que le but n'est pas atteignable ; par contre si $h^1(s) = \infty$ alors le but n'est pas atteignable. Cette capacité à reconnaitre l'accessibilité au but est lié à la capacité de h^+ à reconnaitre les impasses [HOFFMANN, 2005] résumé figure 2.2 : les domaines présents dans la colonne *unrecognized* sont les domaines où h^+ n'est pas capable de déterminer les impasses dans tous les cas.

Le tableau 6.10 montre une couverture plus grande des domaines pour $LMBFS_g$, améliorant ainsi le nombre de problèmes résolus dans *barman*, *parcprinter*, *parking*

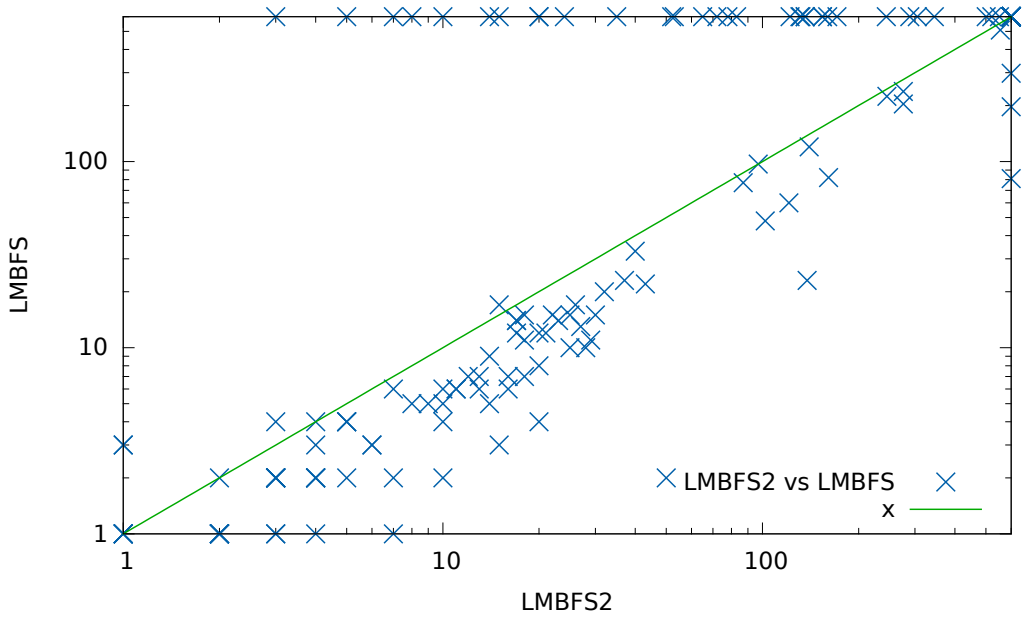


FIGURE 6.4 : Efficacité de la modification du planificateur sous-jacent : temps en secondes pour trouver la première solution. Chaque point représente une instance, l'axe x représente le temps de résolution en utilisant LMBFS$_g$ et l'axe y représente celui de LMBFS. La droite verte représente la fonction linéaire $y = x$. Les points sur les axes $x = 600$ et $y = 600$ n'ont été résolues que par une seule méthode. Pour cette expérience, LMBFS et LMBFS$_g$ utilisent l'heuristique $h^{\mathcal{L}_{\text{left}}}$ et le générateur succDel.

et *woodworking*. Cette amélioration se fait aux dépens de la rapidité pour trouver une première solution, comme nous le montre la figure 6.4.

6.10 Comparaison à d'autres planificateurs sur les domaines de l'IPC 7

Afin d'analyser les performances de l'algorithme LMBFS, nous avons sélectionné les meilleurs planificateurs non-portfolio de la dernière IPC. Depuis la catégorie *agile*, nous avons donc sélectionné *Madagascar-pC* et *PROBE*, et depuis la catégorie sous-optimale : *Jasper* et *Mercury*. Dans cette section nous évaluerons l'algorithme sur l'ensemble des domaines de l'IPC 7, excepté *tidybot* parce que le planificateur sous-jacent que notre implémentation utilise ne le décode pas correctement.

La figure 6.5 et le tableau 6.13 montrent que *Jasper* et *Mercury* résolvent clairement plus de problèmes que les autres planificateurs dès les premières secondes.

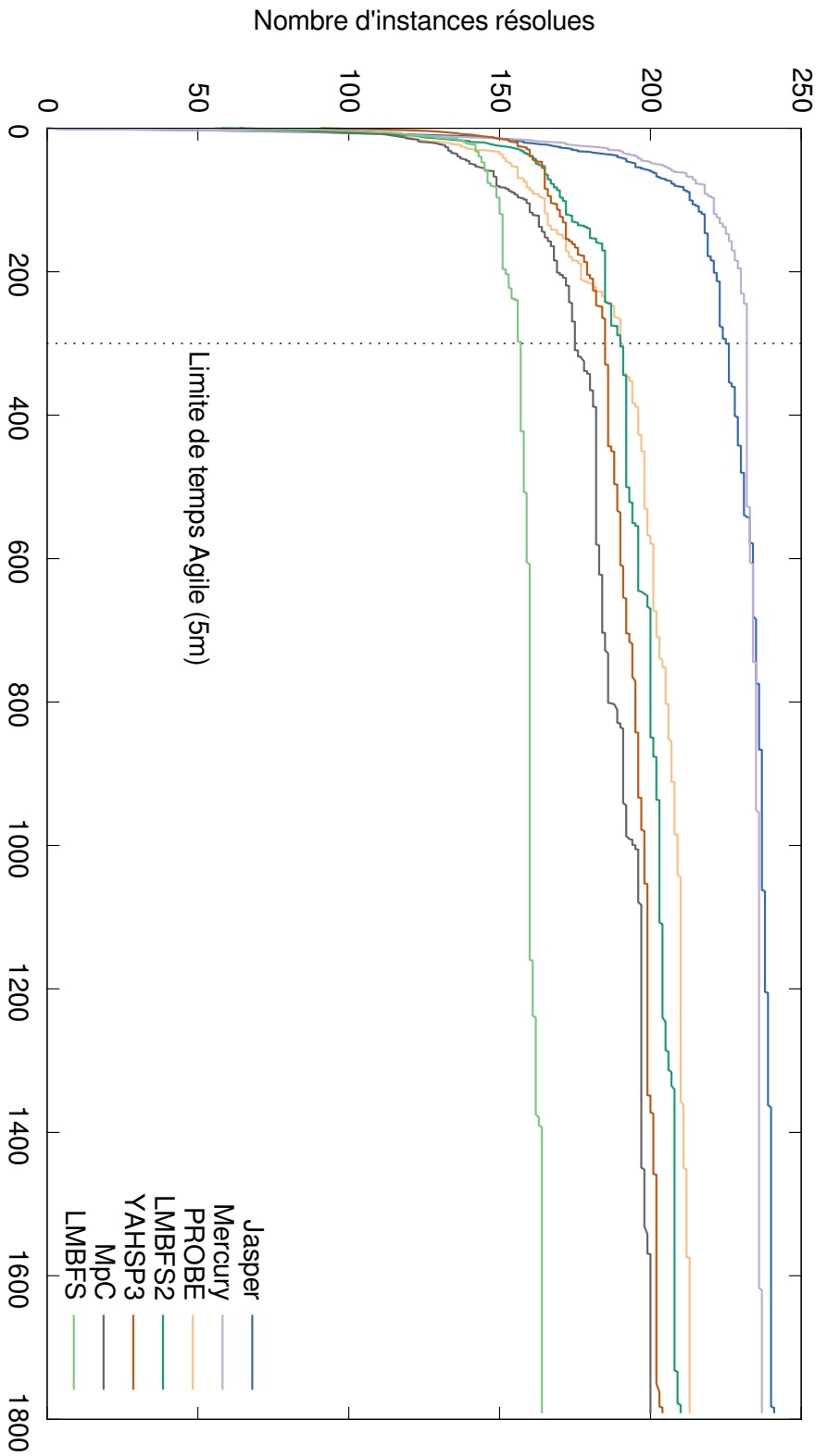


FIGURE 6.5 : Nombre d'instances résolues en fonction du temps sur l'ensemble des domaines de l'IPC 7.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
YAHSP3	2.34	20.00	2.24	5.56	20.00	19.77	4.04
Jasper	12.03	11.02	1.89	15.47	10.09	14.53	18.32
Mercury	12.74	13.08	2.57	12.35	10.25	15.23	19.67
PROBE	18.96	9.79	3.00	4.30	9.06	12.42	11.82
$LMBFS_g$	9.34	15.21	20.00	1.95	9.81	11.69	5.49
MpC	8.10	12.12	20.00	7.56	6.06	20.00	14.28
LMBFS	8.01	17.04	20.00	3.77	10.24	0.66	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
YAHSP3	19.33	14.84	16.17	20.00	20.00	9.10	**173.38**
Jasper	15.89	11.82	12.59	7.43	10.43	17.44	**158.94**
Mercury	15.25	12.54	12.14	9.47	10.49	12.37	**158.16**
PROBE	18.90	18.01	14.92	8.14	9.77	15.66	**154.76**
$LMBFS_g$	13.96	16.47	3.99	14.96	15.55	8.41	**146.84**
MpC	16.38	14.60	1.35	1.40	3.61	19.44	**144.88**
LMBFS	13.86	16.68	2.49	17.16	17.55	1.00	**128.46**

TABLEAU 6.11: Comparaison sur l'IPC 7 en utilisant la métrique m_{time}.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
Jasper	20.00	12.64	2.86	18.86	19.30	19.87	19.01
Mercury	17.29	14.15	4.13	13.89	19.73	19.14	16.89
PROBE	17.97	6.48	2.07	4.86	12.85	12.23	6.16
MpC	11.45	20.00	8.01	14.18	5.90	18.12	9.03
YAHSP3	5.58	2.95	5.61	6.70	12.74	17.81	5.99
$LMBFS_g$	6.65	4.37	19.54	3.90	7.51	13.88	4.75
LMBFS	5.18	4.40	19.98	5.79	7.51	1.61	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
Jasper	20.00	20.00	17.28	12.19	16.08	9.01	**207.10**
Mercury	20.47	17.51	16.28	16.68	20.00	8.01	**203.19**
PROBE	18.60	16.16	13.75	6.41	17.25	9.66	**144.44**
MpC	11.72	13.08	1.93	4.00	4.97	20.00	**142.39**
YAHSP3	13.46	7.91	16.57	6.69	17.19	6.00	**125.19**
$LMBFS_g$	14.36	12.57	5.43	6.99	15.96	5.68	**121.59**
LMBFS	13.07	12.78	3.21	7.01	15.97	0.06	**96.58**

TABLEAU 6.12: Comparaison sur l'IPC 7 en utilisant la métrique $m_{quality}$.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
Jasper	20.00	20.00	5.00	19.00	20.00	20.00	20.00
Mercury	20.00	20.00	6.00	14.00	20.00	20.00	20.00
PROBE	20.00	19.00	5.00	5.00	14.00	14.00	19.00
$LMBFS_g$	20.00	20.00	20.00	4.00	19.00	15.00	13.00
YAHSP3	8.00	20.00	6.00	7.00	20.00	20.00	10.00
MpC	19.00	20.00	20.00	15.00	11.00	20.00	20.00
LMBFS	15.00	20.00	20.00	6.00	19.00	2.00	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
Jasper	20.00	20.00	18.00	19.00	20.00	20.00	**241.00**
Mercury	20.00	20.00	17.00	20.00	20.00	20.00	**237.00**
PROBE	20.00	20.00	17.00	20.00	20.00	20.00	**213.00**
$LMBFS_g$	19.00	17.00	8.00	20.00	20.00	15.00	**210.00**
YAHSP3	20.00	18.00	19.00	20.00	20.00	16.00	**204.00**
MpC	19.00	19.00	4.00	4.00	9.00	20.00	**200.00**
LMBFS	19.00	17.00	5.00	20.00	20.00	1.00	**164.00**

TABLEAU 6.13: Nombre d'instances de problèmes résolues par domaine de l'IPC 7.

$LMBFS_g$ est dans le deuxième groupe où tous les planificateurs restent dans un intervalle restreint de 20 domaines de différences. LMBFS est en retrait.

Au niveau du temps passé pour trouver la première solution, on observe le même comportement, excepté pour YAHSP3 qui arrive en tête (tableau 6.11).

On peut voir que les deux versions de LMBFS sont en tête sur le domaine *floortile*[35], ce qui est permis grâce à une bonne séquentialisation du but. Mais ce résultat doit être pris avec précaution, puisqu'il provient en partie de l'ordre dans lequel est écrit le but dans le fichier PDDL de descriptions. En effet, dans les benchmarks de l'IPC 7, ce but est écrit dans l'ordre de la grille (de gauche à droite, puis de haut en bas). Ce qui amène le premier ordre total testé par LMBFS à suivre ce même arragement, qui est finalement un ordre logique pour résoudre ce problème (un peu de la même manière que pour le domaine *visitall*, figure 6.2(a)). En mélangeant les fluents composant le but, LMBFS parvient moins facilement à résourdre ce problème. Cette observation amène une première discussion sur l'heuristique $h^{\mathcal{L}_{\text{left}}}$ qui ne choisit finalement pas le prochain landmark à atteindre et qui est très certainement une des pistes d'amélioration de l'algorithme.

En termes de qualité (figure 6.12), on peut voir LMBFS est aussi en retrait. Cependant, on peut voir que sur les domaines où YAHSP et LMBFS résolvent autant de problèmes la métrique de qualité est globalement plus grande pour

35. Il s'agit d'un domaine où un ensemble de robots doivent peindre une forme sur un grille. Les robots ne peuvent peindre qu'en face, ou derrière eux et ne peuvent pas passer sur une case qui a déjà été peinte.

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
LMBFS	5.46	10.03	20.00	11.53	20.00	0.00	7.97	13.15	18.08	**106.21**
$LMBFS_g$	7.09	0.00	20.00	9.49	18.18	0.40	7.74	11.11	13.26	**87.27**
YAHSP3	1.44	0.00	0.92	18.99	11.90	0.00	8.83	20.00	20.00	**82.09**
PROBE	18.50	0.00	1.33	11.97	10.08	6.36	18.00	5.76	3.07	**75.07**
Mercury	12.63	0.00	1.50	13.93	6.97	19.91	0.00	8.95	9.08	**72.97**
Jasper	11.90	0.00	1.20	14.65	7.98	17.10	0.00	5.06	9.52	**67.40**
MpC	3.03	7.00	20.00	6.45	5.13	9.01	4.54	0.00	0.00	**55.16**

TABLEAU 6.14: Comparaison sur l'IPC 8 en utilisant la métrique m_{time}.

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
Mercury	14.06	0.00	1.39	19.59	16.16	18.91	0.00	20.00	20.00	**110.12**
Jasper	19.97	0.00	1.39	18.05	17.43	18.03	0.00	12.24	15.37	**102.49**
LMBFS	3.75	10.07	20.00	0.51	14.41	0.00	7.34	8.00	15.65	**79.72**
PROBE	15.55	0.00	0.81	6.82	16.14	5.24	18.00	6.43	7.21	**76.21**
$LMBFS_g$	5.34	0.00	19.77	0.46	14.30	0.41	7.19	7.92	15.59	**70.98**
YAHSP3	2.20	0.00	1.80	1.25	3.53	0.00	7.59	7.91	17.19	**41.48**
MpC	4.93	7.00	8.14	2.67	4.33	7.19	3.17	0.00	0.00	**37.44**

TABLEAU 6.15: Comparaison sur l'IPC 8 en utilisant la métrique $m_{quality}$.

les deux versions de LMBFS (par exemple *elevators*, *pegsol*, *scanalyser*, *transport* et *visitall*). Donc LMBFS n'altère pas la qualité des solutions avec YAHSP en tant que planificateur sous-jacent, et peut même l'améliorer.

6.11 Comparaison à d'autres planificateurs sur les domaines de l'IPC 8

Dans cette section nous évaluerons l'algorithme sur l'ensemble des domaines de l'IPC 8. On peut tout d'abord observer que la progression des deux versions de LMBFS sur le nombre d'instances résolues (figure 6.6 et tableau 6.16). De même, la métrique m_{time} (tableau 6.14) confirme ces résultats.

On peut observer aussi que l'algorithme LMBFS passe devant $LMBFS_g$ dans toutes les métriques. En nombre d'instances résolues, on peut voir que le domaine *childsnack*[36] est résolu seulement par $LMBFS_g$. Ce domaine ne comporte aucun landmark causal autre que les fluents du but. Pour LMBFS, la problématique revient à trouver une bonne manière de séquencer le but pour résoudre le problème. En

36. Dans ce domaine, il faut faire des sandwichs des enfants en tenant compte de l'allergie de certains enfants au gluten.

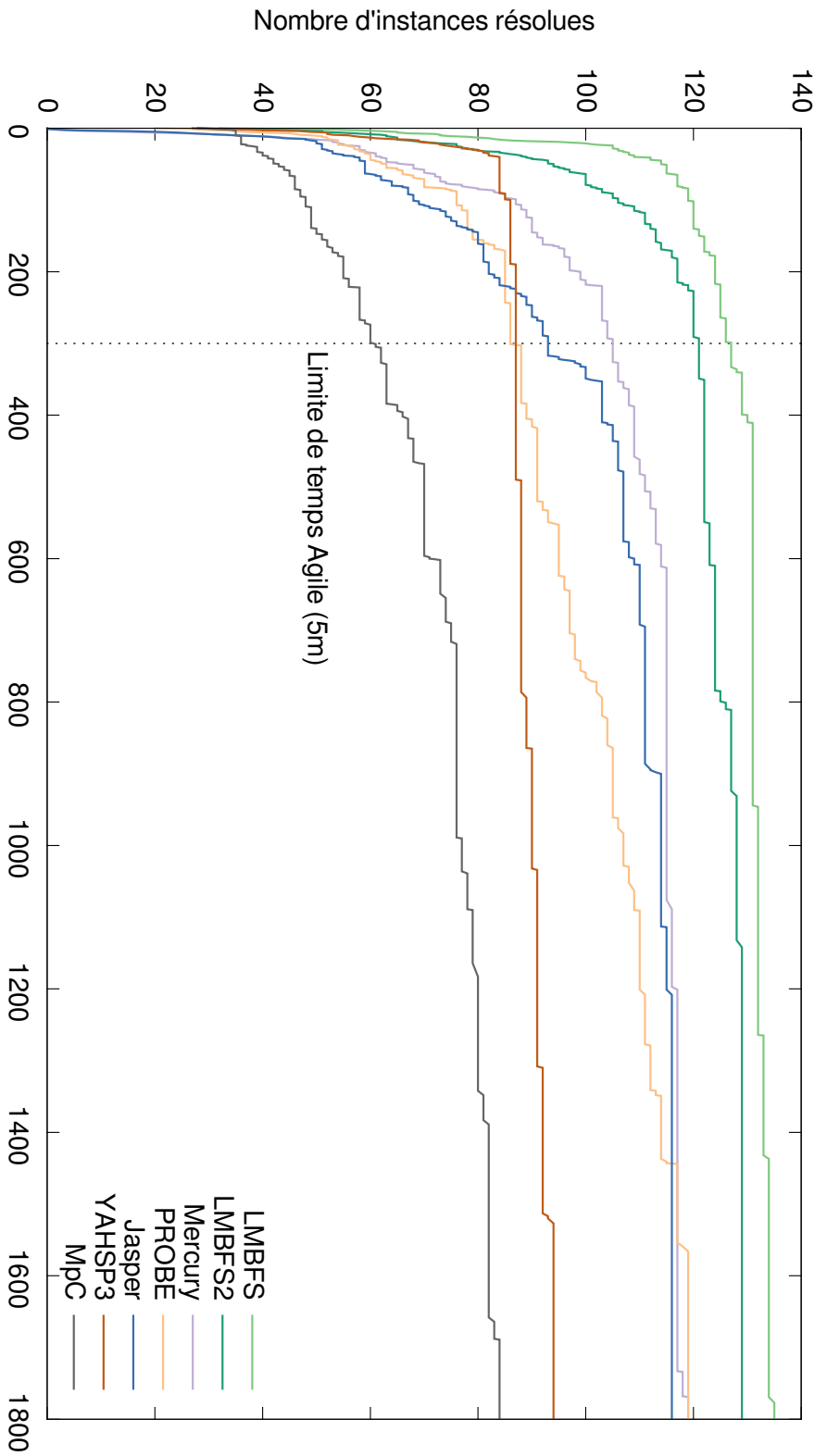


FIGURE 6.6 : Nombre d'instances résolues en fonction du temps sur l'ensemble des domaines de l'IPC 8.

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
LMBFS	13.00	12.00	20.00	20.00	20.00	0.00	10.00	20.00	20.00	**135.00**
$LMBFS_g$	19.00	0.00	20.00	19.00	20.00	1.00	10.00	20.00	20.00	**129.00**
Mercury	20.00	0.00	2.00	20.00	17.00	20.00	0.00	20.00	20.00	**119.00**
PROBE	20.00	0.00	2.00	20.00	20.00	12.00	18.00	18.00	9.00	**119.00**
Jasper	20.00	0.00	2.00	20.00	19.00	20.00	0.00	15.00	20.00	**116.00**
YAHSP3	5.00	0.00	2.00	20.00	17.00	0.00	10.00	20.00	20.00	**94.00**
MpC	11.00	7.00	20.00	18.00	9.00	14.00	5.00	0.00	0.00	**84.00**

TABLEAU 6.16: Nombre d'instances de problèmes résolues par domaine de l'IPC 8.

effet, dans ce domaine, le problème de ressources est critique. En effet, dans les instances le nombre de pains avec et sans gluten correspond au nombre d'enfants allergiques ou non. Ainsi, le problème relaxé ne relève l'impasse que lorsqu'aucun pain d'un type n'est disponible, par contre s'il reste au moins un pain disponible alors les heuristiques basées sur la relaxation du problème considérerons que le problème est toujours soluble même si plusieurs enfants ont besoin de ce type de pain. Ainsi, un ordre où l'ont s'occupe de servir d'abord les enfants qui ne peuvent manger que du pain sans gluten permet de résoudre le problème (à condition que le planificateur sous-jacent ne propose pas de solution gaspillant du pain).

Par contre, la métrique qualité montre que les deux versions de LMBFS ne produisent pas des plans de bonne qualité (figure 6.15). Ceux deux versions produisent des plans de qualité comparable à YAHSP3.

6.12 Discussion

Nous avons pu construire deux versions efficaces de l'algorithme LMBFS, utilisant l'évaluation différée des métaœuds, la génération paresseuse et une heuristique permettant à l'algorithme de fonctionner à la manière d'une recherche en profondeur. Une première piste d'amélioration de l'algorithme est dans la conception d'une meilleure heuristique que la simple $h^{\mathcal{L}_{left}}$. En effet, cette heuristique permet d'explorer des ordres totaux complètement, et idéalement atteindre une solution rapidement. Cependant, elle construit les ordres totaux de manière aléatoire[37]. Ainsi, on peut certainement ajouter à ce niveau une sélection du prochain landmark racine qui doit être utilisé comme sous-but. Nous avons par exemple essayé de choisir le landmark le plus proche selon h^1, mais sans résultat probant.

La deuxième version, $LMBFS_g$, modifie le planificateur sous-jacent de manière à conserver à tout moment l'accessibilité au but global. Cette deuxième version

37. En fait, les ordres sont construit de manière déterministe en fonction de la manière dont est écrite le problème PDDL.

permet une bien meilleure couverture de l'ensemble des problèmes de l'IPC 7, mais est un peu en dessous de la version nominale sur les problèmes de l'IPC 8. Le coût induit par le test d'accessibilité au but ralenti globalement la recherche, mais permet de résoudre les problèmes où les impasses ne sont pas réversibles. Pour aller plus loin, on peut aussi envisager de modifier l'heuristique utilisée par le planificateur sous-jacent. Par exemple envisager une heuristique qui fusionnerait une heuristique visant le landmark but h_l et une autre visant le but global h_g. On chercherait donc à minimiser h_l, mais pas au prix d'une augmentation de h_g.

Les opérateurs de suppression de landmarks sont appliquées sans discernement, on pourrait envisager l'apprentissage des erreurs lors de l'échec de résolution d'un ordre total. Et ainsi essayer de repérer le landmark qui fait échouer la séquence, et choisir de ne supprimer que celui-ci. D'autre part on pourrait ajouter certains ordres non nécessairement corrects dans le graphe de landmarks. Un peu à la manière des *no-good* en CSP, où certaines méthodes apprennent les incompatibilités en ajoutant des contraintes au problème.

Enfin, on pourrait s'inspirer des sondes (*probe*, cf. section 2.4.4), pour la construction d'ordres totaux, voire pour estimer l'intérêt de certains ordres totaux par rapport à d'autres.

7

PARALLÉLISATION DE L'ALGORITHME

Suite à une expérience réussie de la parallélisation d'un algorithme de recherche de solutions sous-optimales basée sur un algorithme de recherche de type *Best-First* : YAHSP-MT [VIDAL, VERNHES et INFANTES, 2011] ; nous avons essayé de paralléliser l'algorithme LMBFS en suivant la même méthode de parallélisation : HDA* [KISHIMOTO, FUKUNAGA et BOTEA, 2009].

Au-delà des défis algorithmiques, on s'est aussi intéressé aux questions techniques rendant l'implémentation souvent difficile, et les comportements potentiellement inattendus des algorithmes (la parallélisation rend potentiellement non déterministes les algorithmes déterministes). De plus les performances des algorithmes deviennent fortement dépendantes de leur implémentation, de l'architecture matérielle sous-jacente, et des échanges de données entre les tâches (latences, débit des échanges).

Cette partie est un travail encore au stade embryonnaire, la version parallélisée ici ne correspond pas tout à fait à l'algorithme LMBFS présenté dans les parties précédentes (la sélection d'action est utilisée dans ce chapitre) et la version LMBFS_g n'a pas été testée en version parallèle. Cependant, ce chapitre présente les premiers écueils rencontrés lors de la parallélisation de l'algorithme LMBFS.

Sommaire

7.1 Parallélisation des algorithmes de type *Best-First*

Les premières parallélisations des algorithmes de recherche de type *Best-First* appliquées à la planification se sont tout d'abord intéressées à la résolution sur de petites architectures, multi-cœur sur une seule machine ; tout d'abord pour la résolution optimale des problèmes [Zhou et Hansen, 2007], puis pour la résolution sous-optimale et anytime [Burns et al., 2009] ; [Burns et al., 2010].

HDA*[Kishimoto, Fukunaga et Botea, 2009] est une extension de l'algorithme A* conçue pour la résolution optimale des problèmes de planification sur des grappes de serveurs à mémoire distribuée. Il s'agit d'un algorithme décentralisé, c'est-à-dire qu'il n'y a pas d'unité de traitement maîtresse rassemblant les données et contrôlant le déroulement de l'algorithme : les échanges de messages se font entre les différentes unités de traitement sans chef d'orchestre. Cet algorithme distribue chaque nœud du graphe de recherche entre les différentes unités de traitement grâce à une clé de hachage calculée à partir des états du problème de planification. Pour cela, la liste des nœuds qui doivent être développés (liste ouverte) est disjointe, chaque unité de traitement possède sa propre liste ouverte, ainsi chaque calcul qui doit être fait pour un état s (sélection des actions applicables, développement du nœud, calcul des heuristiques...) est fait une seule fois par l'unité de traitement auquel cet état s est assigné. La communication entre les unités de traitement est faite de manière asynchrone : une unité de traitement développe un nœud de sa liste ouverte, puis envoie les fils de ce nœud aux unités de traitement auxquels ils appartiennent. Il doit aussi vérifier périodiquement si de nouveaux messages contenant de nouveaux nœuds lui appartenant lui ont été envoyés pour les incorporer dans sa propre liste ouverte. Ce principe a donc été appliqué avec succès en planification optimale [Kishimoto, Fukunaga et Botea, 2009] et aussi appliqué avec succès en planification sous-optimale [Vidal, Vernhes et Infantes, 2011].

7.2 HDA*(LMBFS)

La première étape a consisté à définir une clé de hachage représentant un métanœud. Une clé unique représentant un métanœud $m = \langle s, l, \mathcal{A} \rangle$, qui est toujours composé de :

- s, l'état de départ du sous-problème ;
- $\mathcal{A}$, l'ensemble des landmarks déjà atteints ;
- l, le landmark but du sous-problème.

Cette clé est ensuite hachée en un entier naturel modulo le nombre d'unités de traitement assignant ainsi une unité de traitement unique à chaque métanœud, en se basant de la méthode de hashage *DJB2* [38] qui est détaillée dans l'algorithme 5.

38. Cette fonction est aussi appelée *DJBX33A* pour *Daniel J. Bernstein, Times 33 with Addition*.

Algorithme 5: Metanœud — Hash (basé sur DJB2)

Entrées : Un métanœud $m = \langle s, l, \mathcal{A} \rangle$
le nombre d'unités de traitement n
Sorties : Un entier naturel positif

```
/* On considère que les états du problème de planification et
   les ensembles de landmarks sont représentés par des tableaux
   de bits, où s(i) représente le ième octet du bitarray s. On
   considère aussi que le but l est représenté par un unique
   identifiant entier                                          */
```

$key \leftarrow 5381$
for $i \leftarrow 0$ **to** $nombre\_octets(s) - 1$ **do**
  $key \leftarrow key * 33 + s(i)$
for $i \leftarrow 0$ **to** $nombre\_octets(A) - 1$ **do**
  $key \leftarrow key * 33 + \mathcal{A}(i)$
$key \leftarrow key * 33 + l$
retourner $key \ \% \ n$

L'algorithme 6 détaille l'algorithme LMBFS parallélisé en s'inspirant fortement de l'algorithme HDA*. Les modifications principales de LMBFS par rapport à sa version séquentielle sont :

- Ligne 4 : les métanœuds initiaux sont seulement créés par l'unité de traitement maître [39] puis envoyés à leurs unités de traitement respectives ;
- Ligne 8 : avant de sélectionner le prochain métanœud à ouvrir, l'algorithme vérifie s'il y a de nouveaux messages entrants ;
- Ligne 18 : pour chaque métanœud généré, on calcule sa clé de hachage et on l'envoie à son unité de traitement ;
- Ligne 20 : la condition d'arrêt de la boucle principale est modifiée pour continuer tant qu'aucune unité de calcul n'a trouvé de solution au problème global (plutôt que de continuer tant que la liste ouverte n'est pas vide).

7.3 Détails d'implémentation

L'implémentation de l'algorithme utilise MPI [*MPI : A Message-Passing Interface Standard*], une norme définissant une bibliothèque permettant d'écrire des programmes parallèles via des échanges de données entre processus grâce au *passage de messages*. Plusieurs difficultés ont dû être surmontées afin de réussir une implémentation correcte de l'algorithme.

39. Une unité de traitement est sélectionnée (par exemple l'unité 0) pour les étapes d'initialisation et de finalisation de l'algorithme.

Algorithme 6: Landmark-based Meta Best-First Search : HDA*

Paramètre(s) : Un problème de planification STRIPS $\Pi = \langle F, A, I, G \rangle$, un graphe de landmarks Γ, une fonction générant les successeurs d'un métanœud succ, h une heuristique et subplanner un planificateur.

Valeur(s) de sortie : Un plan solution (ou $\perp$ si le problème n'admet pas de solution)

```
1  O ← ∅                                                    // Liste ouverte
2  C ← ∅                                                    // Liste fermée
3  if est_unité_maitre() then
4      for l ∈ R(Γ) do                  // Génération des métanœuds initiaux
5          m ← ⟨I, l, I ∩ L⟩
6          envoyer(m, hash(m))
7  repeat
8      O ← O ∪ recupere_métanoeud_entrant()
9      m ← argmin_{⟨s,l,A⟩∈O} h(m)
10     O ← O \ {m}
11     if m ∉ C then
12         C ← C ∪ {m}
13         ρ' ← subplanner(Π_m)
14         if ρ' ≠⊥ then
15             if G ⊆ s[ρ'] then                  // but global G trouvé ?
16                 diffuser_message(SOLUTION_TROUVE)
17                 return Plan(m)
18             for m ∈ succ(m) do       // Envoi des métanœuds successeurs
19                 envoyer(m, hash(m))
20 until solution_found()
21 return ⊥
```

En particulier, pour limiter l'utilisation de la mémoire, le plan partiel ρ (amenant de l'état initial I jusqu'à l'état initial du métanœud) n'est pas stocké en entier dans chaque métanœud. Ces derniers enregistrent simplement le plan partiel ρ' solution du sous-problème associé et un pointeur vers le métanœud parent. Donc, pour retrouver un plan complet lorsque l'algorithme a trouvé une solution (Algorithme 6 ligne 15) l'algorithme doit remonter toute la chaine jusqu'au métanœud initial en demandant itérativement aux différentes unités les plans partiels, puis concaténer ceux-ci.

Un autre problème qui est apparu lors des premiers tests de l'algorithme est que contrairement à HDA*, le développement des nœuds peut être potentiellement long. En effet, le développement d'un nœud implique la résolution d'un problème de planification, et dans certain cas cette étape de résolution peut-être longue si le problème est difficile ou impossible (en particulier, pour les métanœuds de type restartCutParents et deleteLM). Si un tel cas apparait dans une unité de calcul, l'algorithme ne peut pas terminer même s'il a trouvé une solution dans une autre unité de calcul (il doit en effet synchroniser toutes les unités de calculs pour pouvoir retrouver la solution). Pour pallier ce problème, le planificateur sous-jacent (YAHSP) a dû être modifié pour vérifier si le message SOLUTION_TROUVE a été envoyé (cette vérification est faite juste avant que YAHSP ouvre un nouveau nœud).

Une amélioration possible de l'algorithme pointé dans une autre méthode de parallélisation d'un algorithme de type A* [Romein et al., 1999] permettant de maîtriser le problème des temps perdu en communication est d'envoyer les nœuds ayant le même destinataire par paquets. Cette stratégie a été implémentée et testée dans le travail sur HDA* [Kishimoto, Fukunaga et Botea, 2009], dans la version parallèle de YAHSP [Vidal, Vernhes et Infantes, 2011] et l'est aussi dans notre algorithme.

La génération paresseuse des nœuds ne peut être implémentée telle quelle dans la version parallèle de l'algorithme puisque les métanœuds générés par une unité de calcul ne sont pas necéssairement développés par celle-ci. Donc le fait que la liste ouverte d'une unité de calcul soit vide n'implique pas nécessairement que tous les métanœuds générés par l'opérateur nextLM aient été ouverts. Cette condition n'est donc pas suffisante pour différer la génération des métanœuds de type deleteLM.

7.4 Dispositif expérimental

Toutes les expérimentations ont été effectuées en utilisant un cluster de 4 serveurs. Chaque machine possède 2 processeurs Intel X5670 (2.96Ghz) comportant 6 cœurs physiques et 24Go de mémoire. Les serveurs peuvent communiquer entre eux via un réseau gigabit Ethernet. Le temps limite pour la résolution des problèmes a été fixé à 10 minutes (temps réel).

La version parallèle de l'algorithme LMBFS basée sur HDA*est dénotée MPI $a \times b$ où a est le nombre de serveurs utilisés, et b le nombre de processus lancés

par serveur. Six différentes versions sont testées pour évaluer le comportement de l'algorithme (passage à l'échelle, communications intra- et inter-serveurs) :

- MPI 1×2,
- MPI 1×4,
- MPI 4×1,
- MPI 4×3,
- MPI 4×6,
- et MPI 4×12.

Dans toutes ces versions, l'heuristique utilisée est $h^{\mathcal{L}_{\mathtt{left}}}$ et l'opérateur succDel pour la génération des métanœuds successeurs. De plus l'algorithme fonctionne en évaluation différée et utilise la sélection d'action via l'opérateur $\mathrm{ops}^{r}_{\Gamma}$ (cf. section 6.6).

D'autre part, une version parallèle non coopérative de l'algorithme LMBFS a été développée. Cette version consiste simplement en un ensemble de versions séquentielles lancées en parallèle où les nœuds ayant la même valeur heuristique sont randomisés (dans la liste ouverte de LMBFS et de YAHSP). Les processus de cette version sont complètement indépendants et ne communiquent pas entre eux, ils s'arrêtent simplement lorsque l'un d'eux a trouvé une solution. Cette version est simplement notée Random, et consiste en 48 processus lancés en parallèle. Dans cette version, les processus sont sujets aux problèmes de compétitivité entre eux, comme en particulier les défauts de cache. Elle permet donc une comparaison plus réaliste qu'avec une version séquentielle simple.

Finalement, on se basera essentiellement sur le speed-up pour comparer les différentes versions parallèles :

Définition 43 : *Speed-up (accélération)*

Le speed-up est l'accélération d'un algorithme parallèle P *par rapport à l'algorithme séquentiel correspondant* S*. C'est-à-dire à quel point* P *est plus rapide que* S *:*

$$S_p : \left| \begin{array}{ccc} \mathbb{R}_*^{+^2} & \longrightarrow & \mathbb{R}_*^+ \\ (T_s, T_p) & \longmapsto & \dfrac{T_s}{T_p} \end{array} \right.$$

où T_s *est le temps de résolution de l'algorithme séquentiel et* T_p*, le temps de résolution de l'algorithme parallèle. Le temps de résolution peut être remplacé par d'autres métriques.*

7.5 Résultats

7.5.1 Nombre de métanœuds ouverts

On peut tout d'abord observer sur la figure 7.1 l'augmentation du nombre de métanœuds ouverts par instance. Le nombre de métanœuds ouverts semble

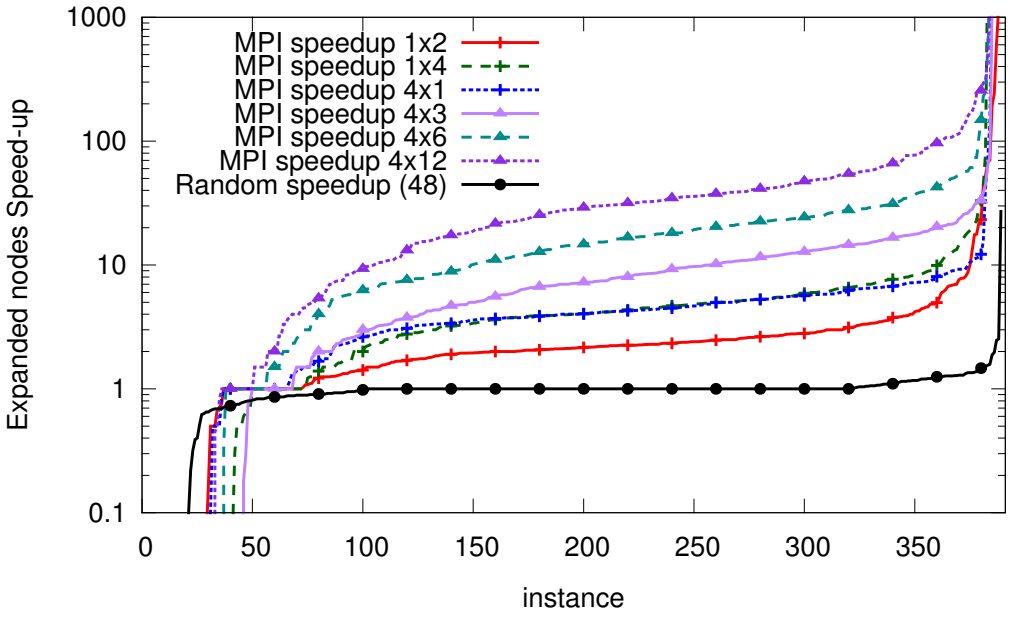


FIGURE 7.1 : Augmentation du nombre de métanœuds ouverts : version parallèle vs. version séquentielle (Les instances sont triées par speed-up croissant).

augmenter quasi linéairement avec l'augmentation du nombre d'unités de calculs. De ce point de vue, la communication inter-serveurs ne semble pas avoir plus d'impact que la communication intra-serveur (lorsqu'on passe de la version MPI 1×4 à MPI 4×1). Pour certaines instances (moins de 50), on observe un speed-up très faible, il s'agit en fait des instances où la version séquentielle ouvre très peu de noœuds, mais où la version parallèle en ouvre par conséquent plusieurs en parallèle avant de s'arrêter.

7.5.2 Temps de résolution

La figure 7.2(a) montre par contre que, même si le nombre de métanœuds ouverts augmente, la version MPI de l'algorithme ne passe pas à l'échelle, et il fait même moins bien que la version séquentielle dans certaines instances. La version MPI a besoin d'un temps de pré-calcul et de synchronisation pour démarrer (environ une seconde). Mais, même si nous sélectionnons les instances résolues en plus d'une seconde (Figure 7.2(b)), nous pouvons voir que dans la plupart des instances, la version parallèle prend autant de temps que la version séquentielle (sinon plus, lorsque le nombre d'unités de calcul utilisées augmente). Pour un petit sous-ensemble d'instances, on observe quand même un speed-up linéaire voir super-linéaire.

Le pourcentage d'instances résolues n'est pas non plus amélioré par rapport

Pourcentage d'instances résolues en moins de	1s	10s	10m
Séquentielle	72.05 %	84.87 %	92.82 %
Aléatoire (48 processus)	82.56 %	93.08 %	97.69 %
MPI 1×2	65.64 %	79.49 %	90.26 %
MPI 1×4	60.00 %	75.64 %	87.69 %
MPI 4×1	63.59 %	78.97 %	88.72 %
MPI 4×3	57.18 %	75.90 %	87.18 %
MPI 4×6	32.31 %	77.44 %	90.00 %
MPI 4×12	0.00 %	73.08 %	92.05 %

TABLEAU 7.1: Pourcentage d'instances résolues par LMBFS.

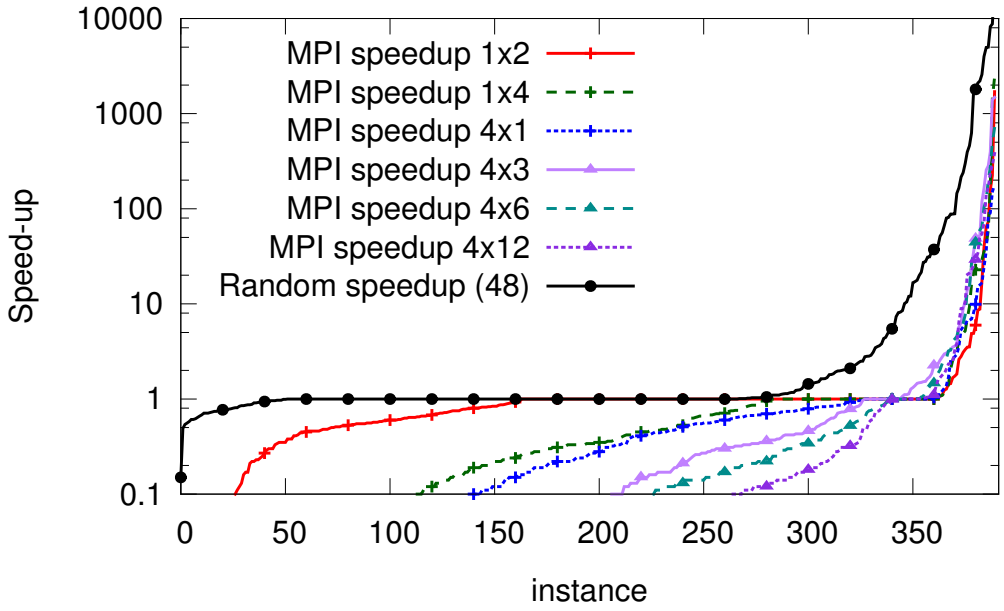
à la version séquentielle. Cependant, il augmente avec le nombre de processus (tableau 7.1).

7.5.3 Liste ouverte avec *tie-breaking* aléatoire

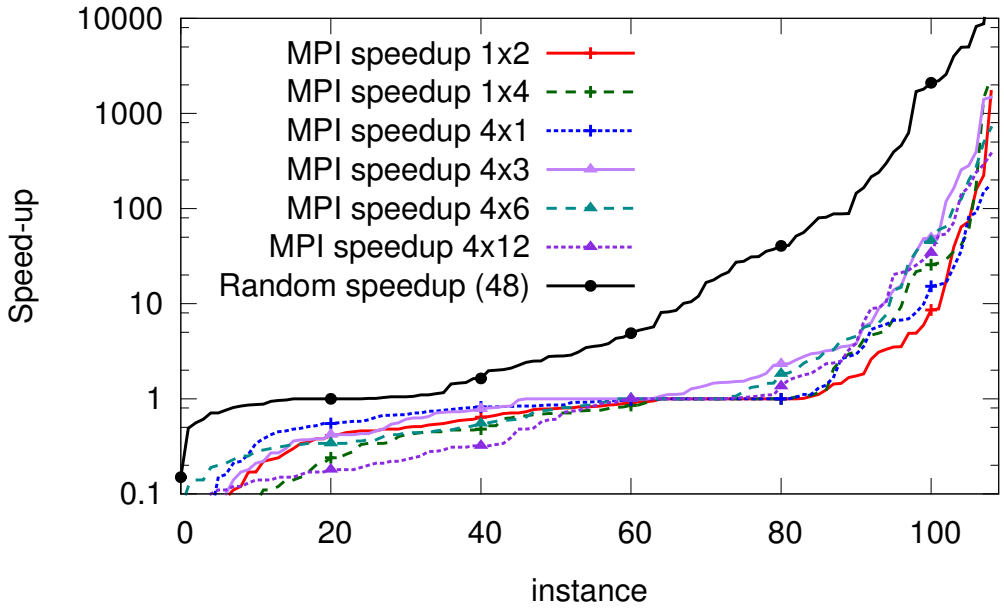
Sur les figures 7.2(a) et 7.2(b), on peut voir que la version aléatoire de l'algorithme (avec 48 processus en parallèle) est toujours meilleure que la version parallèle. Les figures 7.3(a) et 7.3(b) montrent aussi cette supériorité face à la version séquentielle de LMBFS. On peut donc voir qu'une piste possible pour améliorer l'algorithme LMBFS est de trouver de meilleures heuristiques, capables de courber les larges plateaux induits par l'heuristique $h^{\mathcal{L}_{\text{left}}}$.

7.6 Discussion

Cette version parallèle de l'algorithme (MPI) explore beaucoup de métanœuds peu intéressants, en grande partie générés par l'opérateur deleteLM. Les problèmes associés sont en général plus difficiles, moins guidés par le graphe de landmarks, puisque les métanœuds de type deleteLM apparaissent plus tôt. En effet, à cause du temps de transmission et de la parallélisation de l'algorithme, les métanœuds de type deleteLM, même si délaissés au profit des métanœud nextLM, ont tendance à apparaître plus tôt. Ce comportement amène le planificateur sous-jacent à être bloqué dans la résolution de problèmes difficiles alors qu'une unité de calcul devrait explorer les métanœuds plus intéressants reçus pendant cette longue tentative de résolution. En d'autres termes, ces développements de métanœuds prématurés amène l'algorithme vers des branches inintéressantes de l'arbre de recherche. Une première étape serait de tester une parallélisation de l'algorithme LMBFS_g. La recherche en largeur induite par l'algorithme HDA* couplée à la résolution moins gloutonne des sous-problèmes pourrait mener à de meilleurs résultats qu'avec l'algorithme LMBFS. Il y a cependant de fortes chances de retrouver les mêmes

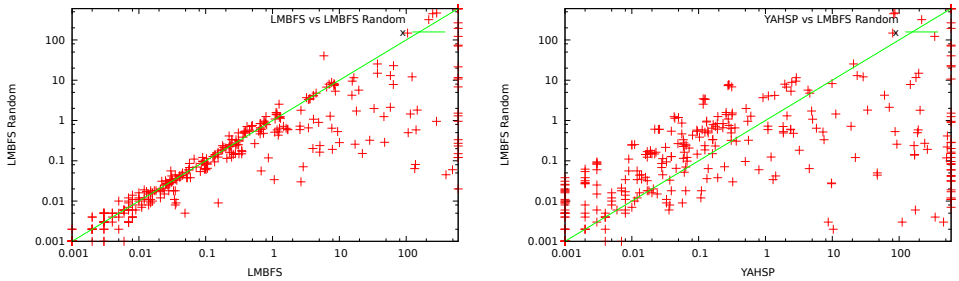


(a) versions parallèles vs version séquentielle



(b) versions parallèles vs version séquentielle (Sélection d'instances où la version séquentielle résout les problèmes en plus d'une seconde)

FIGURE 7.2 : Speed-up des temps de résolutions (temps réel, triés par speed-up croissant).



(a) Temps de résolution pour LMBFS : version séquentielle vs. version aléatoire (48)

(b) Temps de résolution pour YAHSP vs. version aléatoire (48)

FIGURE 7.3 : Résultats pour la version aléatoire (en secondes)

problèmes qu'avec l'algorithme LMBFS. Cependant, on a vu que HDA* ne semble pas adapté pour paralléliser un algorithme de ce type.

De plus, l'algorithme LMBFS est basé sur l'exploration des ordres de landmarks. La version séquentielle essaie, à la manière d'une recherche en profondeur, tous les ordres totaux possibles jusqu'à pouvoir vider complètement le graphe des landmarks (imposé par l'heuristique $h^{\mathcal{L}_{\mathtt{left}}}$ et la génération paresseuse, cf. section 6.5). Cependant, dans la version parallèle, ce comportement n'est pas reproduit à cause de l'exploration prématurée des métanœuds de type deleteLM.

Si on arrive à sélectionner a priori des ordres totaux complets, on pourrait en résoudre différents en parallèle. De plus, à la manière des portfolios, on pourrait même lancer la résolution de différents ordres totaux en utilisant des heuristiques, des paramètres et des techniques différentes, sélectionnés grâce à un analyse du problème initial et de l'ordre lui-même.

Enfin, à la manière de l'algorithme SteLLA, on pourrait s'intéresser à l'approximation des états intermédiaires, et ainsi calculer en parallèle l'ensemble des sous-problèmes d'un ordre total préalablement choisi.

8

CONJONCTION DE LANDMARKS

Afin d'ajouter plus d'informations au graphe de landmarks, et ainsi permettre à l'algorithme LMBFS de faire des recherches intermédiaires plus informées, nous nous sommes intéressés à l'extraction de landmarks conjonctifs depuis un graphe de landmarks.

Cette partie de l'étude est au stade embryonnaire. La théorie développée ici n'est certainement pas complètement aboutie. Elle permet cependant d'éclairer en partie l'intérêt des landmarks conjonctifs pour l'algorithme LMBFS, mais aussi pour la recherche en avant dans l'espace d'états.

Sommaire

8.1 Landmarks Conjonctifs

Partant de la définition des landmarks causaux (définition 21), on peut définir l'ensemble de landmarks causaux conjonctifs en généralisant la relation d'appartenance à une relation d'inclusion :

Définition 44 : *Landmarks causaux conjonctifs*

Soit $\Pi = \langle F, A, I, G \rangle$ *un problème de planification. Un fluent* $l \in F$ *est un* landmark causal conjonctif *pour* Π *s'il respecte une de ces deux conditions :*

$$\begin{cases} l \subseteq G \\ (\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l \subseteq pre(a) \end{cases}$$

On notera $\mathcal{L}_{cc}(\Pi)$ *(ou* $\mathcal{L}$ *quand non ambigu), l'ensemble des landmarks causaux conjonctifs du problème* Π*.*

Une première manière d'extraire des landmarks causaux conjonctifs est de calculer l'intersection des préconditions des actions ayant un landmark en précondition. L'intuition étant que si toutes les actions ayant un landmark en précondition ont aussi d'autres landmarks en précondition alors ces landmarks doivent être présents ensemble dans un état à un moment dans tous les plans solutions :

Théorème 8 : *Extraction d'un sous ensemble de* $\mathcal{L}_{cc}$ *à partir des préconditions*

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification, et* $\mathcal{L}_c$ *un ensemble de landmarks causaux.*
Notons $\mathcal{C}_{pre}$*, la fonction donnant l'ensemble des actions consommant un fluent, définie telle que :*

$$\mathcal{C}_{pre} : \left| \begin{array}{ccc} F & \longrightarrow & 2^A \\ f & \longmapsto & \{a \in A \mid f \in pre(a)\} \end{array} \right.$$

Alors, $(\forall l \in \mathcal{L})$*, l'ensemble*

$$l_c = \left(\bigcap_{a \in \mathcal{C}_{pre}(l)} pre(a) \right) \cap \mathcal{L}$$

est un landmark causal conjonctif.
Notons l_{c_0}*, le landmark* $l \in l_c$ *ayant généré le landmark conjonctif* l_c*.*
Notons $\mathcal{L}_c^{pre}$ *l'ensemble des landmarks causaux conjonctifs issus de cette méthode d'extraction.*

Démonstration :

Grâce à la définition des landmarks causaux, on sait que pour tout $l \in l_c$, on a une de ces deux conditions :

$$\begin{cases} l \in G \\ (\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l \in pre(a) \text{ c'est-à-dire } a \in \mathcal{C}_{pre}(l) \end{cases}$$

car l est un landmark causal. Donc, si $l \notin G$, alors

$$(\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ a \in \mathcal{C}_{\text{pre}}(l) \tag{1}$$

Or :

$$l_c = \left(\bigcap_{a \in \mathcal{C}_{\text{pre}}(l)} pre(a) \right) \cap \mathcal{L}$$

c'est-à-dire :

$$(\forall a \in \mathcal{C}_{\text{pre}}(l))\ l_c \subseteq pre(a)$$

Donc, d'après (1) :

$$(\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l_c \subseteq pre(a)$$

D'autre part, si $l \in G$ alors :

- Si $(\forall l' \in l_c)\ l' \in G$, alors $l_c \subseteq G$;
- sinon, $(\exists l' \in l_c)\ l' \notin G$ alors l_c est un landmark causal conjonctif, via la preuve ci-dessus.

■

D'autre part, on peut aussi calculer l'intersection des préconditions des actions ajoutant un landmark. L'intuition étant que si toutes les actions ajoutant un landmark ont plusieurs landmarks en précondition alors ces landmarks doivent être présents ensembles dans un état à un moment dans tous les plans solutions :

Théorème 9 : ***Extraction d'un sous ensemble de $\mathcal{L}_{cc}$ à partir des ajouts***

Soient $\Pi = \langle F, A, I, G \rangle$ un problème de planification, et $\mathcal{L}_c$ un ensemble de landmarks causaux.
Notons $\mathcal{C}_{\text{add}}$, la fonction donnant l'ensemble des actions ajoutant un fluent, définie telle que :

$$\mathcal{C}_{\text{add}} : \left| \begin{array}{ccc} F & \longrightarrow & 2^A \\ f & \longmapsto & \{a \in A \mid f \in add(a)\} \end{array} \right.$$

Alors, $(\forall l \in \mathcal{L})$, l'ensemble

$$l_c = \left(\bigcap_{a \in \mathcal{C}_{\text{add}}(l)} pre(a) \right) \cap \mathcal{L}$$

est un landmark causal conjonctif.
Notons $\mathcal{L}_c^{add}$ l'ensemble des landmarks causaux conjonctifs issus de cette méthode d'extraction.

Nous nous sommes limités à ces deux types d'extraction, mais on pourrait aussi s'interesser aux intersections d'ajouts d'actions.

8.2 Ordres

Il faut ensuite pouvoir insérer ces landmarks dans le graphe des landmarks. Pour cela nous pouvons essayer de voir quelles relations entre landmarks s'étendent aux landmarks conjonctifs.

Tout d'abord, tout landmark devant être obtenu avant un landmark de la conjonction doit aussi être obtenu avant la conjonction elle-même :

Théorème 10 : ***Ordre naturel → incident sur l'ensemble*** $\mathcal{L}_c^{pre}$

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $\mathcal{L}$ *un ensemble de landmarks et* $l_c \in \mathcal{L}_c^{pre}$ *un landmark causal conjonctif issu du théorème 8.*

$$(\forall l \in l_c)\left(\forall l' \in \mathcal{L} \setminus \{l\}\right)\ l' \to l \implies l' \to l_c$$

Démonstration :

Prenons, $l \in l_c$ et $l' \in \mathcal{L} \setminus \{l\}$ tel que $l' \to l$.
On a, par définition,

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l \in I[\rho_{1\to j}] \implies (\exists i < j)\ l' \in I[\rho_{1\to i}] \tag{1}$$

Or, puisque $(\forall s \in 2^F)\ l_c \subseteq s \implies l \in s$, on a :

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l_c \subseteq I[\rho_{1\to j}] \implies l \in I[\rho_{1\to j}]$$

De plus, on sait par définition que l_c remplit une de ces deux conditions :

$$\begin{cases} l_c \subseteq G \\ (\forall \rho \in \mathcal{P}^\star)(\exists a \in \rho)\ l_c \subseteq pre(a) \end{cases}$$

Dans les deux cas, on sait que $(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l_c \subseteq I[\rho_{1\to j}]$, donc soit :

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l_c \subseteq I[\rho_{1\to j}] \implies l \in I[\rho_{1\to j}]$$

Donc, par transitivé avec (1),

$$(\forall \rho \in \mathcal{P}^\star)(\exists j)\ l_c \subseteq I[\rho_{1\to j}] \implies l \in I[\rho_{1\to j}] \implies (\exists i < j)\ l' \in I[\rho_{1\to i}]$$

Donc

$$l' \to l_c$$

■

D'autre part, tout landmark étant obtenu après le landmark générateur l_{c_0} doit être obtenu après le landmark conjonctif.

Théorème 11 : ***Ordre naturel → sortant sur l'ensemble*** $\mathcal{L}_c^{pre}$

Soient $\Pi = \langle F, A, I, G \rangle$ *un problème de planification,* $\mathcal{L}$ *un ensemble de landmarks et* l_c *un landmark causal conjonctif issu du théorème 8.*

$$\left(\forall l' \in \mathcal{L}\right)\ l_{c_0} \to l' \implies l_c \to l'$$

Et on peut les définir de la même manière pour le théorème 9.

Théorème 12 : *Ordre naturel → incident sur l'ensemble $\mathcal{L}_c^{add}$*

Soient $\Pi = \langle F, A, I, G \rangle$ un problème de planification, $\mathcal{L}$, un ensemble de landmarks et l_c un landmark causal conjonctif issu du théorème 9.

$$(\forall l \in l_c,)(\exists l' \in l_c \setminus \{l\})\ l \rightarrow l' \implies l \rightarrow l_c$$

Théorème 13 : *Ordre glouton impératif → incident sur l'ensemble $\mathcal{L}_c^{add}$*

Soient $\Pi = \langle F, A, I, G \rangle$ un problème de planification, $\mathcal{L}$, un ensemble de landmarks et l_c un landmark causal conjonctif issu du théorème 9.

$$l_c \rightarrow_{gn} l_{c_0}$$

Par contre, le fait d'être un landmark d'une conjonction $l \in l_c$ n'implique pas necessairement d'avoir $l \rightarrow l_c$. En effet, il est possible que l'on puisse obtenir ces landmarks uniquement en même temps.

8.3 Suppression de landmarks simples

En utilisant ces méthodes de génération de landmarks conjonctifs, certains landmarks deviennent redondants. Ainsi, tout landmark dont l'ensemble des arcs entrants et sortants est un sous-ensemble des arcs d'un landmark conjonctif est supprimé du graphe.

Théorème 14 : *Landmark non-informatif*

Soit $\mathcal{L}$, un ensemble de landmarks comprendant des landmarks simples et des landmarks issus de $\mathcal{L}_c^{pre}$ et de $\mathcal{L}_c^{pre}$. Soit $(l, l_c) \in \mathcal{L}^2$ tel que $l \in l_c$.
Si

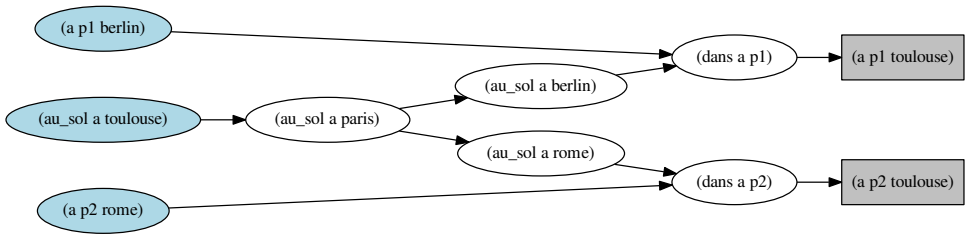
$$\begin{cases} (\forall l' \in \mathcal{L})\ l \rightarrow l' \text{ et } l_c \rightarrow l' \\ (\forall l' \in \mathcal{L})\ l' \rightarrow l \text{ et } l' \rightarrow l_c \end{cases}$$

Alors, le landmark l est redondant par rapport à l_c.

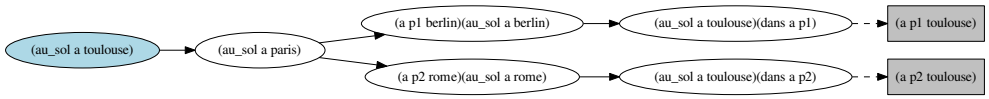
On peut voir un exemple de génération de graphe de landmark conjonctif sur la figure 8.1 pour le problème fil-rouge Zenotravel. On peut remarquer qu'il capture bien le fait que l'avion et le passager doivent se trouver au même endroit pour avancer dans la résolution du problème.

8.4 Expérimentation

Nous avons comparé cette méthode de génération de landmarks conjonctifs avec l'algorithme LMBFS, que nous dénommerons LMBFS_c avec une limite de dix minutes. Ce nouvel algorithme n'améliore pas les performances de l'algorithme,



(a) Graphe de landmarks simple



(b) Graphe de landmarks avec conjonction

FIGURE 8.1 : Graphe de landmarks pour le problème fil-rouge Zenotravel avec et sans conjonctions.

ni sur les domaines de l'IPC 7, ni sur ceux de l'IPC 8 (figures 8.2 et 8.3). Non seulement l'algorithme est plus lent pour trouver une solution, mais il résout moins de problèmes.

Les temps de génération des landmarks conjonctifs sont globalement assez faibles et négligeables, sauf pour certains domaines tel que *openstacks* ou *visitall* (tableau 8.1). Pour ces deux derniers, le problème vient de la méthode de réduction transitive que nous utilisons [40] lors de la génération du graphe de landmarks qui a une complexité de $\mathcal{O}(|\mathcal{L}|^3)$ [41].

8.5 Implémentation dans Fast-Downward

Cette méthode de génération a aussi été implémentée dans Fast-Downward, de manière à tester si elle pouvait améliorer les résultats obtenus par l'heuristique h^{LM}. Pour cela nous avons ajouté notre méthode d'extraction de landmarks conjonctifs après la méthode d'extraction de landmarks depuis le formalisme SAS$^+$ utilisée par LAMA [Richter, Helmert et Westphal, 2008] (noté LAMA$_c$). De même, notre méthode a été utilisée après unification des landmarks utilisés par LAMA et ceux

40. Cette méthode de réduction transitive est basée sur l'algorithme de Floyd-Warshall.

41. Il n'y a pas ce problème dans l'implémentation que nous avons faite avec le planificateur Fast-Downward.

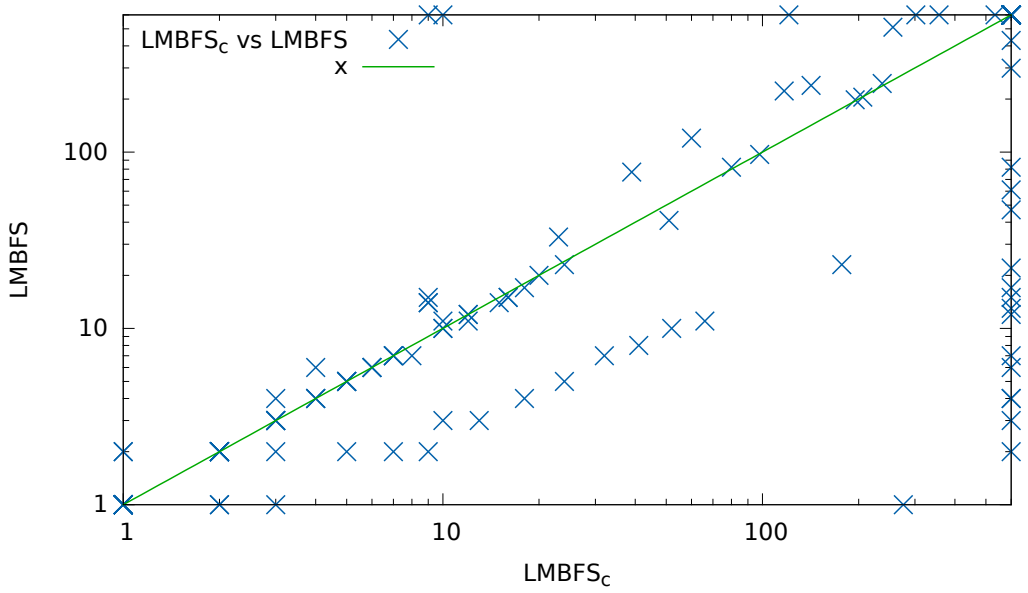


FIGURE 8.2 : LMBFS vs LMBFS$_c$. Chaque point représente les temps de résolution pour chaque instance de l'IPC 7.

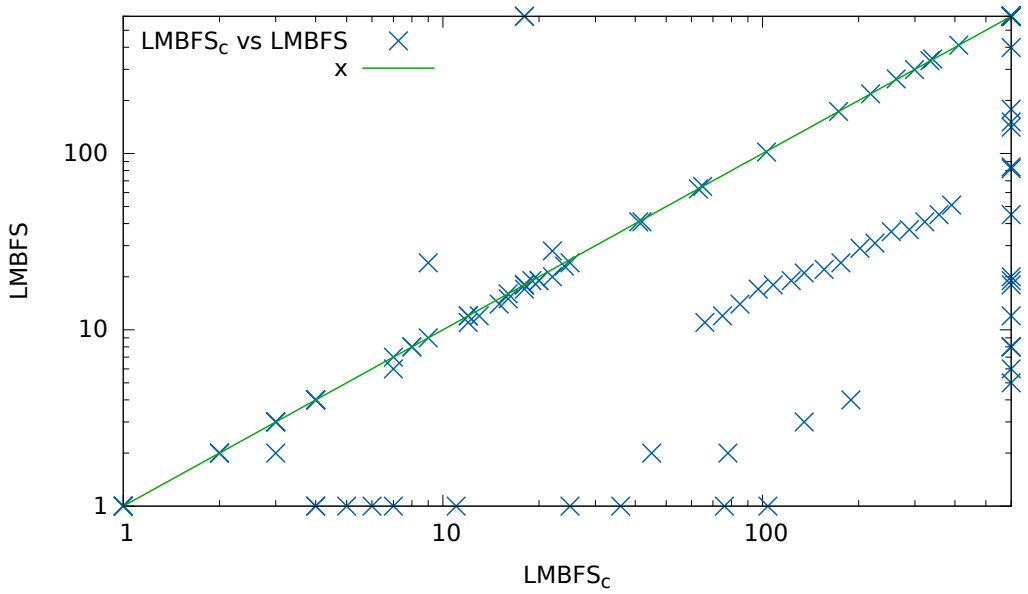


FIGURE 8.3 : LMBFS vs LMBFS$_c$. Chaque point représente les temps de résolution pour chaque instance de l'IPC 8.

Domaine	augmentation moyenne du nombre de landmarks	Temps moyen (s)
ipc7-barman	1.68	0.01
ipc7-elevators	1.71	0.49
ipc7-floortile	1.00	0.00
ipc7-nomystery	2.01	0.01
ipc7-openstacks	1.40	9.25
ipc7-parcprinter	1.48	0.02
ipc7-parking	2.11	0.58
ipc7-pegsol	1.11	0.00
ipc7-scanalyzer	1.00	0.14
ipc7-sokoban	1.03	0.00
ipc7-transport	1.00	0.25
ipc7-visitall	1.00	17.82
ipc7-woodworking	1.65	0.22
ipc8-barman	1.69	0.01
ipc8-childsnack	1.00	0.01
ipc8-floortile	1.00	0.00
ipc8-ged	1.00	0.02
ipc8-hiking	1.31	0.34
ipc8-parking	2.11	1.74
ipc8-thoughtful	1.09	0.10
ipc8-transport	1.00	0.75
ipc8-visitall	1.00	187.25

TABLEAU 8.1: Augmentation moyenne du nombre de landmarks et temps de génération moyen par domaine.

de la méthode de Zhu et Givan [2003] (noté Merged). Le planificateur LAMA servira de planificateur support.

Le nombre d'instances résolues est cependant inférieur dans tous les domaines de l'IPC 7 (figure 8.2), impactant ainsi le score global des deux autres métriques (figures 8.3 et 8.4). On peut cependant noter une amélioration de qualité sur certains domaines (comme *barman, elevators, floortile*), mais de façon non consistante sur l'ensemble des domaines. On peut aussi observer une diminution du temps passé pour trouver la première solution sur le domaine *barman*, mais une augmentation de ce temps sur pratiquement toutes les autres instances.

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking	
LAMA	20.00	20.00	6.00	12.00	20.00	20.00	20.00	
Merged	20.00	20.00	6.00	7.00	20.00	16.00	20.00	
$LAMA_c$	20.00	20.00	6.00	7.00	20.00	16.00	20.00	
	pegsol	scanalyzer	sokoban	tidybot	transport	visitall	woodworking	**total**
LAMA	20.00	20.00	19.00	17.00	19.00	20.00	20.00	**253.00**
Merged	20.00	20.00	19.00	15.00	19.00	20.00	20.00	**242.00**
$LAMA_c$	20.00	20.00	19.00	15.00	18.00	20.00	20.00	**241.00**

TABLEAU 8.2: Nombre d'instances résolues (IPC 7).

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking	
LAMA	18.38	17.93	5.43	11.93	19.84	19.98	19.22	
Merged	19.83	18.61	6.00	6.93	18.94	15.98	16.30	
$LAMA_c$	19.40	17.41	5.43	6.93	19.04	16.00	16.30	
	pegsol	scanalyzer	sokoban	tidybot	transport	visitall	woodworking	**total**
LAMA	20.00	19.14	18.57	16.66	18.24	20.00	19.88	**245.20**
Merged	19.90	19.00	18.83	14.27	18.53	20.00	19.72	**232.84**
$LAMA_c$	20.00	19.07	18.60	14.27	17.24	20.00	19.77	**229.47**

TABLEAU 8.3: Métrique $m_{quality}$ (IPC 7).

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking	
LAMA	15.18	19.86	6.00	11.24	20.00	20.00	19.71	
$LAMA_c$	17.72	18.67	5.99	7.00	12.50	16.00	16.40	
Merged	19.77	17.71	5.99	7.00	11.20	16.00	15.32	
	pegsol	scanalyzer	sokoban	tidybot	transport	visitall	woodworking	**total**
LAMA	20.00	20.00	18.07	17.00	19.00	20.00	16.41	**242.47**
$LAMA_c$	20.00	19.50	18.60	12.94	17.78	19.61	18.42	**221.13**
Merged	19.99	17.16	17.16	11.40	17.08	18.84	14.80	**209.43**

TABLEAU 8.4: Métrique m_{time} (IPC 7).

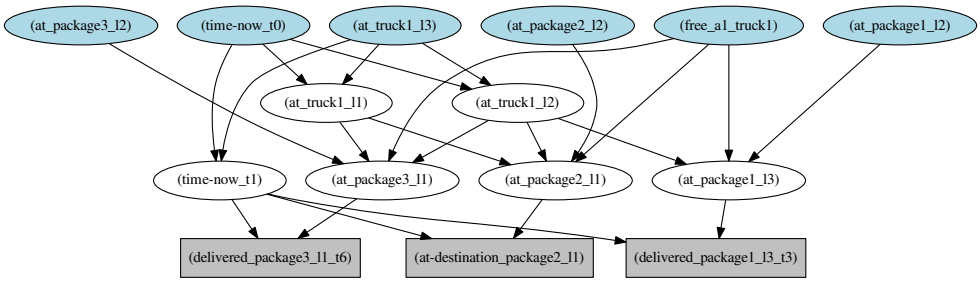


FIGURE 8.4 : Graphe de landmarks d'un problème du domaine *trucks*.

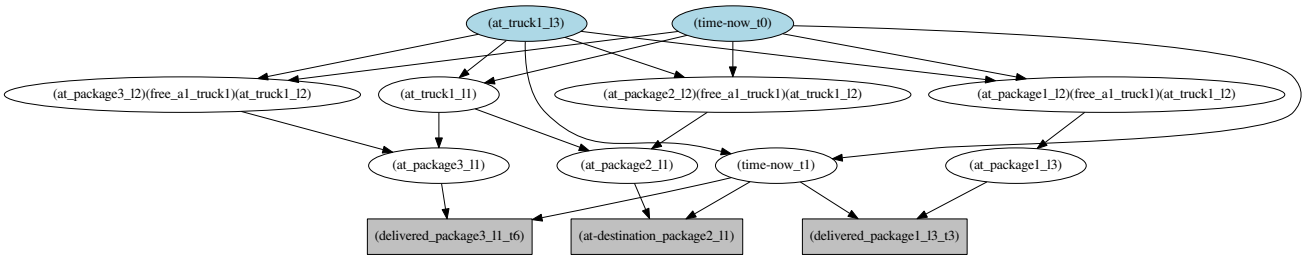


FIGURE 8.5 : Graphe de landmarks d'un problème du domaine *trucks* avec landmarks conjonctifs.

8.6 Discussion

Initialement, cette méthode a été conçue pour faire remonter plus tôt dans le graphe des landmarks objectifs. Si on considère le domaine *trucks*, où des colis doivent être déplacés entre certaines zones grâce à une flotte de camion, certains landmarks sont préjudiciables lorsque qu'on les utilise comme sous-buts. La figure 8.4 nous montre un exemple de graphe de landmarks pour un problème de ce domaine. On peut voir que les landmarks $(\texttt{at truck}_1\ \texttt{l}_1)$ et $(\texttt{at truck}_1\ \texttt{l}_2)$ ne portent pas suffisamment d'information s'ils sont utilisés seuls. En effet, il faut avoir chargé le bon colis dans le camion, avant de le déplacer à la zone d'arrivée.

Dans la plupart des domaines de logistique où ce problème apparait, l'algorithme LMBFS s'en sort en augmentant la longueur du plan (en fait, en faisant effectuer un aller-retour au camion). Mais dans ce domaine, les colis doivent être livrés à un temps donné, et les allers-retours de camions ne permettent pas de trouver de solution. Ainsi, le planificateur LMBFS dépend complètement de l'opérateur deleteLM pour résoudre ce type de problèmes.

On peut constater que notre méthode n'a pas forcément tendance à faire remon-

ter la bonne information (figure 8.5). Le problème pour ce domaine particulier[42] est le fait que l'information « le colis est dans le camion » est une disjonction de fluents. Ainsi, il faut plutôt s'intéresser à la modélisation des ressources dans le graphe de landmarks.

42. Ce problème est commun à tous les domaines de type logistique et lorsqu'une ressource est en jeu.

Troisième partie

Conclusion et Perspectives

9 Conclusion

L'objectif de cette thèse consiste en la conception d'un planificateur basé sur la segmentation d'un problème classique en un ensemble de sous-problèmes en utilisant des landmarks. Cette disjonction de l'espace permettra peut-être ainsi une parallélisation aisée de l'algorithme.

Dans un premier temps, nous avons donc pu développer l'algorithme *Landmark-based Meta Best First Search* : un algorithme essayant de reconsidérer les techniques de partition de l'espace de recherche grâce à des landmarks et leur utilisation comme sous-buts. Nous avons pu définir des métanœuds, nœuds d'un algorithme de recherche globale de type *Best-First search*, mais en même temps associables à un sous-problème de planification. Une difficulté a pu être identifiée et surmontée : le fait que le générateur de métanœuds nextLM n'est pas suffisant pour assurer la complétude de l'algorithme, amenant ainsi à la conception de nouveaux générateurs, deleteLM et cutParents.

Nous avons pu construire deux versions efficaces de l'algorithme LMBFS, utilisant l'évaluation différée des métaœuds, la génération paresseuse et une heuristique permettant à l'algorithme de fonctionner à la manière d'une recherche en profondeur. La deuxième version, LMBFS_g , modifie le planificateur sous-jacent de manière à conserver à tout moment l'accessibilité au but global. Cette deuxième version permet une bien meilleure couverture de l'ensemble des problèmes de l'IPC 7, mais est un peu en dessous de la version nominale sur les problèmes de l'IPC 8. Le coût induit par le test d'accessibilité au but ralentit globalement la recherche, mais permet de résoudre les problèmes où les impasses ne sont pas réversibles.

Dans un deuxième temps, nous avons tenté une première parallélisation de notre algorithme. Cette version parallèle de l'algorithme (MPI) explore beaucoup de métanœuds peu intéressants, en grande partie générés par l'opérateur deleteLM. Les problèmes associés sont en général plus difficiles, moins guidés par le graphe de landmarks, puisque les métanœuds de type deleteLM apparaissent plus tôt. En effet, à cause du temps de transmission et de la parallélisation de l'algorithme, les

métanœuds de type deleteLM, même si délaissés au profit des métanœud nextLM, ont tendance à apparaître plus tôt. Ce comportement amène le planificateur sous-jacent à être bloqué dans la résolution de problèmes difficiles alors qu'une unité de calcul devrait explorer les métanœuds plus intéressants reçus pendant cette longue tentative de résolution. En d'autres termes, ces développements de métanœuds prématurés amène l'algorithme vers des branches inintéressantes de l'arbre de recherche. Même si non convaincante, cette parallélisation de l'algorithme a pu nous permettre de comprendre un peu plus certains aspects de l'algorithme LMBFS, mais aussi à identifier un cas où la méthode HDA* ne semble pas trop adapté.

D'autre part, afin d'ajouter plus d'informations au graphe de landmarks, et ainsi permettre à l'algorithme LMBFS de faire des recherches intermédiaires plus informées, nous nous sommes intéressés à l'extraction de landmarks conjonctifs depuis un graphe de landmarks. Nous avons effectivement pu concevoir deux méthodes d'extraction de landmarks conjonctifs à partir d'un ensemble de landmarks existant et d'une analyse des actions du problème. Cependant, nous avons pu voir que ces méthodes ne permettent pas nécessairement de faire remonter les bonnes informations dans le graphe de landmarks, et n'améliorent que rarement les temps de résolutions de l'algorithme LMBFS.

Nous avons pu montrer que la décomposition du problème de planification de tâches basée sur les landmarks est une méthode qui est toujours compétitive. Les résultats entre les deux dernières compétitions internationales de planification montrent cependant des différences entre les classements des mêmes planificateurs. Ces différences peuvent provenir de l'ensemble des benchmarks choisis et des métriques utilisées. Les résultats obtenus par l'algorithme LMBFS avec les benchmarks de l'IPC 7 pourraient amener à dire qu'il n'est pas pertinent par rapport aux autres techniques alors qu'il montre de très bonne performance sur ceux de l'IPC 8. Et même lors du développement d'un planificateur, on pourrait évincer certaines parties de l'algorithme jugées non pertinentes à la vue d'une sélection de benchmarks, alors qu'elles peuvent s'avérer utiles pour d'autres. Il faudrait ainsi juger un planificateur sur chaque domaine en analysant intrinsèquement les propriétés des techniques utilisées. Un travail qui a d'ailleurs commencé à être étudié grâce à l'analyse de la relaxation sur différents domaines [Hoffmann, 2011].

Publications

Premières réflexions sur un algorithme de segmentation d'un problème de planification en un ensemble de sous-problèmes :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [août 2012].
« The Landmark-based Meta Best-First Search Algorithm for Classical Planning ».
Dans : *Proceedings of the 5th European Starting AI Researcher Symposium (STAIRS-2012)*. T. 241. Frontiers in Artificial Intelligence and Applications. IOS Press, p. 336–347

Présentation de l'algorithme *Landmark-based Meta Best-First Search* avec expérimentations :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [août 2013b].
« Problem Splitting using Heuristic Search in Landmark Orderings ». Dans : *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-2013)*. AAAI Press

Première parallélisation de l'algorithme :
Simon Vernhes, Guillaume Infantes et Vincent Vidal [juin 2013a].
« Landmark-based Meta Best-First Search Algorithm : First Parallelization Attempt and Evaluation ». Dans : *Proceedings of the 5th ICAPS Workshop on Heuristics and Search for Domain-independent Planning (HSDIP-2013)*, p. 44–52

10

PERSPECTIVES

Bien que l'heuristique $h^{\mathcal{L}_{left}}$ affiche de bonnes performances, elle peut être largement améliorée. En effet, lorsqu'on regarde les résultats obtenus en lançant 48 processus en parallèle utilisant une liste ouverte avec *tie-breaking* aléatoire, on se rend compte qu'il y a de la marge pour améliorer l'algorithme. Pour cela il faut être capable de choisir sciemment le prochain landmark qui servira de but. Nous avons essayé d'estimer la distance de tous les landmarks racines depuis l'état courant grâce à h^1 et sélectionner le plus proche, cependant des résultats préliminaires ne montrent pas de gain de performances.

La deuxième version, LMBFS_g , modifie le planificateur sous-jacent de manière à conserver à tout moment l'accessibilité au but global. On pourrait aller plus loin, et envisager de modifier l'heuristique utilisé par le planificateur sous-jacent. Par exemple envisager une heuristique qui fusionnerait une heuristique visant le landmark but h_l et une autre visant le but global h_g. On chercherait donc à minimiser h_l, mais pas au prix d'une augmentation de h_g.

On peut aussi se questionner sur la pertinence des opérateurs cutParents et deleteLM lorsque l'opérateur nextLM s'avère insuffisant, et en particulier comment appliquer ces « sauts » dans le graphe de landmarks de manière plus avisée. On peut ainsi essayer de repérer le landmark qui fait échouer la planification de la séquence de landmarks, et choisir de ne supprimer que celui-ci. D'autres part on pourrait ajouter certains ordres non nécessairement corrects dans le graphe de landmarks. Un peu à la manière des *no-good* en CSP, où certaines méthodes apprennent les incompatibilités en ajoutant des contraintes au problème.

On cherchera aussi à calculer à l'avance des ordres totaux qui sembleraient intéressants, contenant éventuellement des « sauts » du graphe de landmarks et appliquer une recherche complète basée sur cet ordre-là ; c'est-à-dire, une descente complète le long de la branche décrite par cet ordre total. On pourrait aussi changer complètement de type d'algorithme. Ainsi, au lieu d'effectuer un algorithme de type *best-first*, on pourrait calculer à l'avance un ordre total à la manière de l'algorithme

STeLLa. Puis, on utiliserait un planificateur pour résoudre séquentiellement cet ordre total, et en cas d'échec, il faudrait étudier quelle partie pose problème pour modifier l'ordre, voire supprimer certains landmarks. Par exemple, on pourrait s'inspirer des sondes.

Enfin, on pourrait s'intéresser à utiliser un graphe de landmarks, ou une partie de celui-ci comme une donnée du problème. Ainsi, à la manière d'un HTN, l'utilisateur pourrait indiquer une façon de résoudre ce problème. Il indiquerait aussi certaines préférences, comme par exemple la résolution d'une partie du problème avant une autre.

Bibliographie

Alcázar, Vidal et Manuela Veloso (2011).
« A SAT compilation of the landmark graph ». Dans : *Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS)*.

Bäckström, Christer (1995).
« Expressive equivalence of planning formalisms ». Dans : *Artificial Intelligence* 76.1-2, p. 17–34 (cité page 14).

Bäckström, Christer, Yue Chen, Peter Jonsson, Sebastian Ordyniak et Stefan Szeider (2012).
« The Complexity of Planning Revisited - A Parameterized Analysis ». Dans : *Computing Classification System (CoRR)* abs/1208.2566 (cité page 14).

Bäckström, Christer et Bernhard Nebel (1995).
« Complexity results for SAS+ planning ». Dans : *Computational Intelligence* 11.4, p. 625–655 (cité pages 13, 14).

Bibaï, Jacques, Pierre Savéant, Marc Schoenauer et Vincent Vidal (mai 2010).
« An Evolutionary Metaheuristic based on State Decomposition for Domain-Independent Satisficing Planning ». Dans : *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS-2010)*. AAAI Press, p. 18–25 (cité pages 56, 86).

Blum, Avrim L. et Merrick L. Furst (1997).
« Fast planning through planning graph analysis ». Dans : *Artificial Intelligence* 90.1-2, p. 281–300 (cité page 20).

Bonet, Blai et Héctor Geffner (2001).
« Planning as heuristic search ». Dans : *Artificial Intelligence* 129.1, p. 5–33 (cité pages 23, 25, 27).

Bonet, Blai et Héctor Geffner (2008).
« Heuristics for planning with penalties and rewards formulated in logic and computed through circuits ». Dans : *Artificial Intelligence* 172.12-13, p. 1579–1604 (cité page 24).

Burns, Ethan, Seth Lemons, Wheeler Ruml et Rong Zhou (2009).
« Suboptimal and Anytime Heuristic Search on Multi-Core Machines ». Dans : *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)* (cité page 108).

Burns, Ethan, Sofia Lemons, Wheeler Ruml et Rong Zhou (2010).
« Best-first heuristic search for multicore machines ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 39.1, p. 689–743 (cité page 108).

Bylander, Tome (1994).
« The computational complexity of propositional STRIPS planning ». Dans : *Artificial Intelligence* 69.1-2, p. 165–204 (cité pages 13, 21, 46).

Chen, Yixin, Benjamin W Wah et Chih-Wei Hsu (2006).
« Temporal Planning using Subgoal Partitioning and Resolution in SGPlan ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 26, p. 323–369 (cité page 53).

Coles, Amanda J., Andrew Coles, Angel G. Olaya, Sergio Jiménez, Carlos L. Linares, Scott Sanner et Sungwook Yoon (2012).
« A Survey of the Seventh International Planning Competition ». Dans : *AI Magazine* 33.1 (cité pages 58, 59).

Coles, Andrew, Maria Fox, Derek Long et Amanda Smith (2008).
« Planning with Problems Requiring Temporal Coordination ». Dans : *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, p. 892–897 (cité page 16).

Cushing, William, Subbarao Kambhampati, Mausam et Daniel S. Weld (2007).
« When is Temporal Planning Really Temporal ? » Dans : *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, p. 1852–1859 (cité page 16).

Dechter, Rina et Judea Pearl (1985).
« Generalized Best-First Search Strategies and the Optimality of A* ». Dans : *Journal of the ACM* 32.3, p. 505–536 (cité page 34).

Dréo, Johann, Pierre Savéant, Marc Schoenauer et Vincent Vidal (juin 2014).
« Divide-and-Evolve : the Marriage of Descartes and Darwin ». Dans : *Proceedings of the 8th International Planning Competition (IPC-2014)* (cité page 56).

Edelkamp, Stefan et Jörg Hoffmann (2004).
« PDDL2.2 : The language for the classical part of the 4th international planning competition ». Dans : *4th International Planning Competition (IPC'04), at ICAPS-2004* (cité pages 15, 84).

Edelkamp, Stefan et Stefan Schrödl (2012).
Heuristic Search - Theory and Applications. Academic Press.

Ernits, Juhan, Charles Gretton et Richard Dearden (2011).
« Ay Also Plan : Bitstate Pruning for State-Based Planning on Massively Parallel Compute Clusters ». Dans : *Proceedings of the 7th International Planning Competition (IPC'11)* (cité page 60).

Ernst, George W. et Allen Newell (1969).
GPS : A Case Study in Generality and Problem Solving. ACM monograph series. Academic Press.

Erol, Kutluhan, Dana Nau et V. S. Subrahmanian (1995).
« Complexity, Decidability and Undecidability Results for Domain-Independent Planning ». Dans : *Artificial Intelligence* 76.1-2, p. 75–88 (cité page 13).

Fikes, Richard E. et Nils J. Nilsson (1972).
« STRIPS : A new approach to the application of theorem proving to problem solving ». Dans : *Artificial Intelligence* 2.3-4, p. 189–208 (cité pages xx, 10).

Forum, Message Passing Interface.
MPI : A Message-Passing Interface Standard. Rap. tech. (cité page 109).

Fox, Maria et Derek Long (2003).
« PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 20, p. 61–124 (cité pages 15, 84).

— (2011).
« The Automatic Inference of State Invariants in TIM ». Dans : *Computing Classification System (CoRR)* abs/1105.5451 (cité page 53).

Gerevini, Alfonso et Derek Long (2006).
« Preferences and soft constraints in PDDL3 ». Dans : *ICAPS workshop on planning with preferences and soft constraints*, p. 46–53 (cité pages 15, 84).

Ghallab, Malik, Dana Nau et Paolo Traverso (2004).
Automated Planning, theory and practice. Morgan-Kaufmann.

Green, Cordell (1969).
« Application of Theorem Proving to Problem Solving ». Dans : *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI'69)*. Morgan Kaufmann Publishers Inc., p. 219–239 (cité page xx).

Hart, Peter E., Nils J. Nilsson et Bertram Raphael (1968).
« A Formal Basis for the Heuristic Determination of Minimum Cost Paths ». Dans : *IEEE Transactions Systems Science and Cybernetics* 4.2, p. 100–107 (cité page 34).

HASLUM, Patrik (2006).
« Admissible Heuristics for Automated Planning ». Thèse de doctorat. Linköping University, Sweden (cité page 29).

— (2009).
« $h^m(P) = h^1(P^m)$: Alternative Characterisations of the Generalisation From h^{max} To h^m ». Dans : *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)* (cité page 30).

HASLUM, Patrik et Héctor GEFFNER (2000).
« Admissible heuristics for optimal planning ». Dans : *Proceedings of the 5th International Conference of Artificial Intelligence Planning Systems (AIPS)*, p. 140–149 (cité pages 27, 28).

HELMERT, Malte (2004).
« A Planning Heuristic Based on Causal Graph Analysis ». Dans : *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*. T. 16, p. 161–170 (cité page 37).

— (2006).
« The Fast Downward Planning System ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 26, p. 191–246 (cité pages 36, 37, 56, 93).

HELMERT, Malte et Carmel DOMSHLAK (2009).
« Landmarks, Critical Paths and Abstractions : What's the Difference Anyway ? » Dans : *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)*.

HOFFMANN, Jörg (2005).
« Where "Ignoring Delete Lists" Works : Local Search Topology in Planning Benchmarks ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 24, p. 685–758 (cité pages 29, 30, 96).

HOFFMANN, Jörg (2011).
« Analyzing Search Topology Without Running Any Search : On the Connection Between Causal Graphs and h+ ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 41, p. 155–229 (cité pages 29, 134).

HOFFMANN, Jörg, Stefan EDELKAMP, Sylvie THIÉBAUX, Roman ENGLERT, Frederico dos S LIPORACE et Sebastian TRÜG (2006).
« Engineering Benchmarks for Planning : the Domains Used in the Deterministic Part of IPC-4. » Dans : *Journal of Artificial Intelligence Research (JAIR)* 26, p. 453–541 (cité page 15).

HOFFMANN, Jörg et Bernhard NEBEL (2001).
« The FF Planning System : Fast Plan Generation Through Heuristic Search ». Dans : *Journal of Artificial Intelligence Research* 14, p. 253–302 (cité pages 26, 27, 35, 37, 56).

Hoffmann, Jörg, Julie Porteous et Laura Sebastia (2004).
« Ordered Landmarks in Planning ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 22, p. 215–278 (cité pages 46, 47, 51).

Karpas, Erez et Carmel Domshlak (2009).
« Cost-Optimal Planning with Landmarks ». Dans : *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, p. 1728–1733.

Katz, Michael et Jörg Hoffmann (2013).
« Red-Black Relaxed Plan Heuristics Reloaded ». Dans : *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SOCS 2013)* (cité page 31).

— (juin 2014).
« Mercury Planner : Pushing the Limits of Partial Delete Relaxation ». Dans : *Proceedings of the 8th International Planning Competition (IPC-2014)* (cité page 57).

Katz, Michael, Jörg Hoffmann et Carmel Domshlak (2013).
« Red-Black Relaxed Plan Heuristics ». Dans : *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013)* (cité page 31).

Keyder, Emil et Héctor Geffner (2008).
« Heuristics for Planning with Action Costs Revisited. » Dans : *Proceedings of European Conference on Artificial Intelligence (ECAI)*, p. 588–592 (cité page 27).

Keyder, Emil Ragip, Jörg Hoffmann et Patrik Haslum (2012).
« Semi-Relaxed Plan Heuristics ». Dans : *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)* (cité page 30).

Keyder, Emil, Silvia Richter et Malte Helmert (2010).
« Sound and Complete Landmarks for And/Or Graphs ». Dans : *Proceedings of European Conference on Artificial Intelligence (ECAI)*, p. 335–340 (cité pages 48, 86).

Kishimoto, Akihiro, Alex Fukunaga et Adi Botea (2009).
« Scalable, Parallel Best-First Search for Optimal Sequential Planning ». Dans : *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)* (cité pages 107, 108, 111).

Korf, Richard E. (1990).
« Real-Time Heuristic Search ». Dans : *Artificial Intelligence* 42.2-3, p. 189–211 (cité page 16).

Lipovetzky, Nir (2012).
« Structure and inference in classical planning ». Thèse de doctorat. Universitat Pompeu Fabra (cité page 38).

Lipovetzky, Nir et Héctor Geffner (2011).
« Searching for Plans with Carefully Designed Probes ». Dans : *Proceedings of the 21th International Conference on Automated Planning and Scheduling (ICAPS-2011)*, p. 154–161 (cité page 38).

Lipovetzky, Nir et Héctor Geffner (2012).
« Width and Serialization of Classical Planning Problems. » Dans : *Proceedings of European Conference on Artificial Intelligence (ECAI)*, p. 540–545.

McDermott, Drew, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld et David Wilkins (1998).
« PDDL - The Planning Domain Definition Language ». Dans : *Technical Report CVC TR98003/DCS TR-1165 (New Haven, CT : Yale Center for Computational Vision and Control)* (cité pages 14, 84).

Nakhost, Hootan et Martin Mueller (2013).
« Towards a Second Generation Random Walk Planner : An Experimental Exploration ». Dans : *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)* (cité page 57).

Nebel, Bernhard (2011).
« On the Compilability and Expressive Power of Propositional Planning Formalisms ». Dans : *Computing Classification System (CoRR)* abs/1106.0247 (cité page 15).

Newell, Allen et Herbert A Simon (1963).
« Computers & Thought ». Dans : chap. GPS, a Program That Simulates Human Thought, p. 279–293 (cité page xx).

Olaya, Angel García, Carlos Linares López et Sergio Jiménez (2011).
The Seventh International Planning Competition, Deterministic Track. `http://www.plg.inf.uc3m.es/ipc2011-deterministic/` (cité page 58).

Pearl, Judea (1984).
Heuristics - intelligent search strategies for computer problem solving. Addison-Wesley series in artificial intelligence. Addison-Wesley (cité page 31).

Pednault, Edwin PD (1989).
« ADL : Exploring the middle ground between STRIPS and the situation calculus ». Dans : *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Morgan Kaufmann Publishers Inc., p. 324–332 (cité page 14).

Pohl, Ira (1970).
« Heuristic Search Viewed as Path Finding in a Graph ». Dans : *Artificial Intelligence* 1.3, p. 193–204 (cité page 34).

Porteous, Julie et Stephen Cresswell (2002).
« Extending landmarks analysis to reason about resources and repetition ». Dans : *Proceedings of the 21st Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG'02)*.

Porteous, Julie, Laura Sebastia et Jörg Hoffmann (2001).
« On the Extraction, Ordering, and Usage of Landmarks in Planning ». Dans : *roceedings of the 6th European Conference on Planning (ECP-01)* (cité pages 42, 51).

Richter, Silvia (2010).
« Landmark-Based Heuristics and Search Control for Automated Planning ». Thèse de doctorat. Griffith University, Brisbane, Australia (cité pages 36, 37).

Richter, Silvia et Malte Helmert (2009).
« Preferred operators and deferred evaluation in satisficing planning ». Dans : *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-2009)*, p. 273–280 (cité pages 36, 93).

Richter, Silvia, Malte Helmert et Matthias Westphal (2008).
« Landmarks revisited ». Dans : *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, p. 975–982 (cité pages 43, 48, 50, 124).

Richter, Silvia et Matthias Westphal (2010).
« The LAMA planner : Guiding cost-based anytime planning with landmarks ». Dans : *Journal of Artificial Intelligence Research (JAIR)* 39.1, p. 127–177 (cité pages 36, 51, 56).

Rintanen, Jussi (2012).
« Planning as satisfiability : Heuristics ». Dans : *Artificial Intelligence* 193, p. 45–86 (cité page 57).

Romein, John W., Aske Plaat, Henri E. Bal et Jonathan Schaeffer (1999).
« Transposition Table Driven Work Scheduling in Distributed Search ». Dans : *Proceedings of (North American) National Conference on Artificial Intelligence (AAAI)* (cité page 111).

Ruml, Wheeler et Minh Binh Do (2007).
« Best-First Utility-Guided Search ». Dans : *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, p. 2378–2384 (cité page 16).

Sebastia, Laura, Eva Onaindia et Eliseo Marzal (2002).
« SteLLa v2.0 : planning with intermediate goals ». Dans : *Advances in Artificial Intelligence - IBERAMIA 2002, 8th Ibero-American Conference on AI, Seville, Spain, November 12-15, 2002, Proceedings*. T. 2527. Springer, p. 805–814 (cité page 52).

— (2006).
« Decomposition of planning problems ». Dans : *AI Communications* 19, p. 49–81 (cité page 52).

Sutton, Richard S. et Andrew G. Barto (1998).
Reinforcement Learning : An Introduction. A Bradford book. MIT Press (cité page 16).

Thiébaux, Sylvie, Jörg Hoffmann et Bernhard Nebel (2005).
« In defense of PDDL axioms ». Dans : *Artificial Intelligence* 168.1, p. 38–69 (cité page 15).

Valenzano, Richard Anthony, Hootan Nakhost, Martin Müller, Jonathan Schaeffer et Nathan R. Sturtevant (2012).
« ArvandHerd : Parallel Planning with a Portfolio ». Dans : *Proceedings of European Conference on Artificial Intelligence (ECAI)*, p. 786–791 (cité page 60).

Vernhes, Simon, Guillaume Infantes et Vincent Vidal (août 2012).
« The Landmark-based Meta Best-First Search Algorithm for Classical Planning ». Dans : *Proceedings of the 5th European Starting AI Researcher Symposium (STAIRS-2012)*. T. 241. Frontiers in Artificial Intelligence and Applications. IOS Press, p. 336–347 (cité pages xxi, 134).

— (juin 2013a).
« Landmark-based Meta Best-First Search Algorithm : First Parallelization Attempt and Evaluation ». Dans : *Proceedings of the 5th ICAPS Workshop on Heuristics and Search for Domain-independent Planning (HSDIP-2013)*, p. 44–52 (cité pages xxi, 135).

— (août 2013b).
« Problem Splitting using Heuristic Search in Landmark Orderings ». Dans : *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-2013)*. AAAI Press (cité pages xxi, 135).

Vidal, Vincent (juin 2004).
« A Lookahead Strategy for Heuristic Search Planning ». Dans : *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*. AAAI Press, p. 150–159 (cité pages 37, 38).

— (juin 2011).
« YAHSP2 : Keep It Simple, Stupid ». Dans : *Proceedings of the 7th International Planning Competition (IPC-2011)*, p. 83–90 (cité pages 38, 56, 86).

— (juin 2014).
« YAHSP3 and YAHSP3-MT in the 8th International Planning Competition ». Dans : *Proceedings of the 8th International Planning Competition (IPC-2014)* (cité page 56).

Vidal, Vincent et Héctor Geffner (2005).
« Solving simple planning problems with more inference and no search ». Dans : *Principles and Practice of Constraint Programming (CP 2005)*, p. 682–696.

Vidal, Vincent, Simon Vernhes et Guillaume Infantes (2011).
« Parallel AI Planning on the SCC ». Dans : *Proceedings of the 4th Symposium of the Many-core Applications Research Community (MARC)* (cité pages 107, 108, 111).

Wehrle, Martin, Sebastian Kupferschmid et Andreas Podelski (2008).
« Useless Actions Are Useful ». Dans : *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS-2008)*, p. 388–395 (cité page 37).

Xie, Fan, Martin Müller et Robert Holte (2014a).
« Adding Local Exploration to Greedy Best-First Search in Satisficing Planning ». Dans : *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, p. 2388–2394 (cité page 57).

— (juin 2014b).
« Jasper : the Art of Exploration in Greedy Best First Search ». Dans : *Proceedings of the 8th International Planning Competition (IPC-2014)* (cité page 57).

Xie, Fan, Martin Müller, Robert Holte et Tatsuya Imai (2014).
« Type-Based Exploration with Multiple Search Queues for Satisficing Planning ». Dans : *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, p. 2395–2402 (cité page 57).

Zhou, Rong et Eric A Hansen (2007).
« Parallel structured duplicate detection ». Dans : *Proceedings of (North American) National Conference on Artificial Intelligence (AAAI)*, p. 1217–1224 (cité page 108).

Zhu, Lin et Robert Givan (2003).
« Landmark Extraction via Planning Graph Propagation ». Dans : *ICAPS Doctoral Consortium*, p. 156–160 (cité pages 42, 47, 50, 86, 126).

Appendix

NOTATIONS ET ACRONYMES

Cette annexe présente brièvement l'ensemble des notations utilisées dans cette thèse.

Sommaire

A.1 STRIPS

Π	$\triangleq$	Problème de planification STRIPS (défininition 2)
Π	$=$	$\langle F, A, I, G \rangle$
F	$\triangleq$	Ensemble des fluents instanciés
A	$\triangleq$	Ensemble des actions instanciées
I	$\triangleq$	État initial
G	$\triangleq$	But
f	$\triangleq$	Fluent instancié
s	$\triangleq$	État
a	$\triangleq$	Action (définition 4)
$pre(a)$	$\triangleq$	Ensemble des fluents nécessaires pour l'exécution de a
$add(a)$	$\triangleq$	Ensemble des fluents ajoutés pour l'exécution de a
$del(a)$	$\triangleq$	Ensemble des fluents supprimés pour l'exécution de a
$apply(s, a)$	$\triangleq$	Application de l'action a à l'état s
$s[a]$	$=$	$apply(s, a)$
ρ	$\triangleq$	Plan, Séquence d'actions (définition 7)
$\lvert\rho\rvert$	$\triangleq$	Nombre d'actions du plan ρ
$s[\rho]$	$\triangleq$	Application du plan ρ à l'état s
$\rho_{i \to i'}$	$\triangleq$	Sous-séquence du plan ρ
$\mathcal{P}(\Pi)$	$\triangleq$	Ensemble des plans de Π
$\mathcal{P}^\star(\Pi)$	$\triangleq$	Ensemble des plans solutions de Π

A.2 Heuristiques

Π^+	$\triangleq$	Problème relaxé (définition 11)
$h^\star$	$\triangleq$	Fonction heuristique optimale (définition 13)
h^+	$\triangleq$	Fonction heuristique optimale du problème relaxé (définition 14)
h^{add}	$\triangleq$	Fonction heuristique basées sur l'indépendance entre fluents (définition 17)
h^{ff}	$\triangleq$	Fonction heuristique correspondant à la taille d'un plan relaxé (section 2.2.4)
h^1	$\triangleq$	Fonction heuristique basées sur la distance du fluent le plus difficile à obtenir (définition 18)
h^{max}	$=$	h^1
h^m	$\triangleq$	Fonction heuristique basées sur la distance de la conjonction de taille m de fluents la plus difficile à obtenir (définition 19)
h^{LM}	$\triangleq$	Fonction heuristique basées sur les landmarks (définition 29)
$h^{\mathcal{L}_{left}}$	$\triangleq$	Fonction heuristique basées sur les landmarks (définition 41)

A.3 Landmarks

$\mathcal{L}_f$	$\triangleq$	Ensemble des landmarks d'un problèmes (définition 20)
$\mathcal{L}_c$	$\triangleq$	Ensemble des landmarks *causaux* d'un problèmes (définition 21)
l	$\triangleq$	Landmark
$\rightarrow$	$\triangleq$	Relation d'ordre partielle entre landmarks (définition 22)
$\rightarrow_n$	$\triangleq$	Relation d'ordre partielle *nécessaire* entre landmarks (définition 27)
$\rightarrow_{gn}$	$\triangleq$	Relation d'ordre partielle *glouton nécessaire* entre landmarks (définition 27)
Γ	$\triangleq$	Graphe des landmarks (définition 23)
Γ	$=$	$\langle \mathcal{L}, \rightarrow \rangle$
$\mathcal{R}(\Gamma)$	$\triangleq$	Ensemble des landmarks racines de Γ (définition 24)
$\mathcal{P}(l)$	$\triangleq$	Landmarks parents de l (définition 25)
$\mathrm{Ch}(l)$	$\triangleq$	Landmarks fils direct de l (définition 25)

A.4 Algorithmes

$\mathcal{O}$	$\triangleq$	Liste ouverte (ensemble des nœuds sur la frontière d'exploration)
$\mathcal{C}$	$\triangleq$	Liste fermé (ensemble des nœuds déjà exploré)
n	$\triangleq$	Nœud de la recherche (associé à un état du problème de planification)
m	$\triangleq$	Metanœud (définition 31)
Π_m	$\triangleq$	Sous-problème associé au metanœud m (définition 33)
$\mathcal{A}$	$\triangleq$	Ensemble de landmarks déjà atteints
A_{tt}	$\triangleq$	Fonction de mise à jour des landmarks atteints (définition 34)

A.5 Acronymes

DSC	Disjunctive Search Control
FF	Fast-Forward
HDA*	Hash Distributed A*
IA	Intelligence Artificielle
IPC	International Planning Competition
LMBFS	Landmark-based Meta Best-First Search
MPI	Message Passing Interface
PDDL	Planning Domain Definition Language
RPG	Relaxed Planning Graph (Graphe de planification relaxé)
STRIPS	Stanford Research Institute Problem Solver (représentation)
SAS	Simplified Action Structures
YAHSP	Yet Another Heuristic Search Planner

DÉFINITIONS DES DOMAINES ET PROBLÈMES EN PDDL

B.1 Domaine Zenotravel en PDDL (pure STRIPS)

B.1.1 Domaine

```
(define (domain zenotravel)
  (:requirements :strips)

  (:types avion passager ville)

  (:predicates
    (dans ?a - avion ?p - passager)
    (au_sol ?a - avion ?v - ville)
    (a ?p - passager ?v - ville)
    (route ?v1 ?v2 - ville)
  )

  (:action deplace
    :parameters
      (?a - avion ?v1 ?v2 - ville)
    :precondition
      (and
        (au_sol ?a ?v1)
        (route ?v1 ?v2)
      )
    :effect
      (and
        (au_sol ?a ?v2)
        (not (au_sol ?a ?v1))
      )
  )
```

```
27
28    (:action embarquer
29      :parameters
30        (?a - avion ?p - passager ?v - ville)
31      :precondition
32        (and
33          (au_sol ?a ?v)
34          (a ?p ?v)
35        )
36      :effect
37        (and
38          (dans ?a ?p)
39          (not (a ?p ?v))
40        )
41    )
42
43    (:action debarquer
44      :parameters
45        (?a - avion ?p - passager ?v - ville)
46      :precondition
47        (and
48          (au_sol ?a ?v)
49          (dans ?a ?p)
50        )
51      :effect
52        (and
53         (a ?p ?v)
54         (not (dans ?a ?p))
55        )
56    )
57  )
```

B.1.2 Problème

```
1  (define (problem zenotravel-1)
2    (:domain zenotravel)
3    (:objects
4      av - avion
5      p1 p2 - passager
6      toulouse paris berlin rome - ville
7    )
8
9    (:init
10     (au_sol av toulouse)
11     (a p1 berlin)
12     (a p2 rome)
13     (route toulouse paris)
14     (route paris toulouse)
15     (route paris berlin)
```

```
16      (route berlin paris)
17      (route paris rome)
18      (route rome paris)
19      (route berlin rome)
20      (route rome berlin)
21    )
22
23    (:goal
24     (and
25      (a p1 toulouse)
26      (a p2 toulouse)
27     )
28    )
29  )
```

C

DÉTAILS DES BENCHMARKS

Sommaire

C.1 IPC 7 : instances résolues

barman

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
010	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
013	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
014	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
017	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
018	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
020	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**

elevators

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	1.00	∅	1.00	1.00	1.00	**1.00**
020	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**

floortile

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	∅	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	∅	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
008	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
009	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
010	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
011	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
012	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
013	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
014	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
015	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
016	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
017	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
018	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
019	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
020	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**

nomystery

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	∅	1.00	1.00	**1.00**
004	1.00	1.00	1.00	∅	∅	∅	1.00	**1.00**
005	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
006	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
007	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
008	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
009	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
010	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	1.00	∅	∅	1.00	1.00	**1.00**
014	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
015	1.00	∅	∅	∅	∅	∅	∅	**1.00**
016	1.00	∅	∅	∅	∅	∅	∅	**1.00**
017	1.00	∅	∅	∅	∅	∅	∅	**1.00**
018	1.00	∅	∅	∅	∅	∅	∅	**1.00**
019	1.00	∅	1.00	∅	∅	∅	∅	**1.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

openstacks

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
016	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
017	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
020	1.00	1.00	∅	∅	∅	∅	1.00	**1.00**

parcprinter

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
004	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
006	1.00	1.00	1.00	∅	1.00	∅	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
010	1.00	1.00	1.00	∅	1.00	∅	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
013	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
014	1.00	1.00	1.00	∅	∅	∅	1.00	**1.00**
015	1.00	1.00	1.00	∅	∅	∅	1.00	**1.00**
016	1.00	1.00	1.00	∅	∅	∅	1.00	**1.00**
017	1.00	1.00	1.00	∅	1.00	∅	1.00	**1.00**
018	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
019	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
020	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**

parking

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
010	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
012	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
013	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
014	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
015	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
016	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
017	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**
018	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
019	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
020	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**

pegsol

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
020	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**

scanalyzer

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
019	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**
020	1.00	1.00	∅	1.00	∅	∅	∅	**1.00**

sokoban

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
004	1.00	1.00	∅	1.00	1.00	∅	1.00	**1.00**
005	1.00	1.00	∅	1.00	1.00	∅	1.00	**1.00**
006	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
008	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	∅	1.00	∅	1.00	1.00	**1.00**
011	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
012	1.00	1.00	∅	1.00	∅	1.00	1.00	**1.00**
013	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
014	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
015	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
016	1.00	1.00	1.00	∅	1.00	∅	1.00	**1.00**
017	1.00	1.00	∅	1.00	∅	∅	1.00	**1.00**
018	∅	∅	∅	1.00	∅	∅	1.00	**1.00**
019	1.00	∅	∅	∅	∅	∅	1.00	**1.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

transport

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
016	∅	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
020	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**

visitall

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
020	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**

woodworking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
003	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**
012	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
013	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
014	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
015	1.00	1.00	1.00	1.00	∅	∅	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
018	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
019	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
020	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**

Nombre d'instances résolues

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
jasper	20.00	20.00	5.00	19.00	20.00	20.00	20.00
mercury	20.00	20.00	6.00	14.00	20.00	20.00	20.00
probe	20.00	19.00	5.00	5.00	14.00	14.00	19.00
$LMBFS_g$	20.00	20.00	20.00	4.00	19.00	15.00	13.00
yahsp	8.00	20.00	6.00	7.00	20.00	20.00	10.00
mpc	19.00	20.00	20.00	15.00	11.00	20.00	20.00
LMBFS	15.00	20.00	20.00	6.00	19.00	2.00	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
jasper	20.00	20.00	18.00	19.00	20.00	20.00	**241.00**
mercury	20.00	20.00	17.00	20.00	20.00	20.00	**237.00**
probe	20.00	20.00	17.00	20.00	20.00	20.00	**213.00**
$LMBFS_g$	19.00	17.00	8.00	20.00	20.00	15.00	**210.00**
yahsp	20.00	18.00	19.00	20.00	20.00	16.00	**204.00**
mpc	19.00	19.00	4.00	4.00	9.00	20.00	**200.00**
LMBFS	19.00	17.00	5.00	20.00	20.00	1.00	**164.00**

C.2 IPC 7 : métrique m_{time}

barman

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	0.59	0.62	0.51	1.00	0.43	0.48	0.27	**0.00**
002	0.59	0.62	0.47	1.00	0.59	0.62	∅	**0.00**
003	0.59	0.59	1.00	1.00	0.62	0.77	0.25	**0.00**
004	0.59	0.62	1.00	1.00	0.56	0.68	0.31	**0.00**
005	0.62	0.62	0.46	1.00	0.68	0.62	∅	**0.00**
006	0.56	0.62	0.36	1.00	0.47	0.56	0.29	**0.00**
007	0.91	1.00	0.64	0.85	0.72	∅	∅	**4.00**
008	0.62	0.59	0.47	1.00	0.46	0.45	∅	**0.00**
009	0.53	0.54	0.34	1.00	0.40	0.46	∅	**0.00**
010	0.56	0.59	0.25	1.00	0.35	∅	0.26	**0.00**
011	0.48	0.53	0.37	1.00	0.41	0.24	0.30	**0.00**
012	0.94	1.00	0.32	0.46	0.60	0.75	∅	**6.00**
013	0.50	0.56	0.29	1.00	0.39	∅	0.31	**0.00**
014	0.54	0.59	0.25	1.00	0.38	0.43	∅	**0.00**
015	0.80	1.00	0.32	0.65	0.66	0.87	∅	**5.00**
016	0.49	0.47	0.26	1.00	0.33	0.37	∅	**0.00**
017	0.51	0.56	0.28	1.00	0.31	∅	∅	**0.00**
018	0.49	0.50	0.25	1.00	0.32	∅	∅	**0.00**
019	0.59	0.59	∅	1.00	0.32	0.36	0.34	**0.00**
020	0.53	0.51	0.25	1.00	0.31	0.34	∅	**0.00**

elevators

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	0.77	0.68	1.00	1.00	1.00	1.00	1.00	**0.00**
002	0.59	0.59	0.68	0.62	1.00	1.00	1.00	**0.00**
003	0.54	0.56	0.59	0.54	0.77	1.00	1.00	**1.00**
004	0.62	0.62	1.00	1.00	1.00	1.00	1.00	**0.00**
005	0.62	0.62	1.00	1.00	1.00	1.00	1.00	**0.00**
006	0.53	0.56	0.56	0.42	0.68	0.77	1.00	**1.00**
007	0.56	0.59	0.62	0.54	1.00	1.00	1.00	**0.00**
008	0.50	0.56	0.54	0.44	0.68	0.77	1.00	**1.00**
009	0.49	0.56	0.54	0.41	0.62	0.77	1.00	**1.00**
010	0.48	0.54	0.53	0.40	0.62	0.77	1.00	**1.00**
011	0.45	0.57	0.53	0.38	0.62	0.72	1.00	**2.00**
012	0.47	0.56	0.49	0.35	0.60	0.72	1.00	**2.00**
013	0.50	0.61	0.54	0.35	0.64	0.77	1.00	**3.00**
014	0.48	0.60	0.54	0.36	0.61	0.73	1.00	**3.00**
015	0.49	0.59	0.47	0.35	0.60	0.68	1.00	**3.00**
016	0.62	0.84	0.50	0.45	0.78	0.89	1.00	**9.00**
017	0.59	0.89	0.51	0.41	0.79	0.91	1.00	**12.00**
018	0.59	0.85	0.50	0.40	0.75	0.87	1.00	**12.00**
019	0.55	0.83	0.49	∅	0.72	0.84	1.00	**13.00**
020	0.58	0.84	0.48	0.37	0.72	0.84	1.00	**15.00**

floortile

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	0.62	0.62	1.00	0.37	1.00	1.00	0.53	**0.00**
002	0.45	0.56	1.00	0.37	1.00	1.00	0.51	**0.00**
003	0.27	0.45	1.00	∅	1.00	1.00	0.33	**0.00**
004	0.30	0.26	1.00	0.26	1.00	1.00	0.31	**0.00**
005	0.24	0.31	1.00	1.00	1.00	1.00	0.28	**0.00**
006	∅	0.36	1.00	1.00	1.00	1.00	0.27	**0.00**
007	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
008	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
009	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
010	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
011	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
012	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
013	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
014	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
015	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
016	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
017	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
018	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
019	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
020	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**

nomystery

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	0.68	0.77	0.68	1.00	1.00	1.00	1.00	**0.00**
002	0.68	0.68	0.54	1.00	0.26	1.00	1.00	**0.00**
003	0.62	0.62	0.33	1.00	∅	0.24	0.33	**0.00**
004	1.00	0.93	0.35	∅	∅	∅	0.68	**5.00**
005	0.89	1.00	0.31	∅	∅	∅	∅	**6.00**
006	0.58	1.00	0.37	∅	∅	∅	∅	**6.00**
007	0.54	1.00	0.34	∅	∅	∅	∅	**7.00**
008	0.52	1.00	0.40	∅	∅	∅	∅	**5.00**
009	0.92	1.00	0.31	∅	∅	∅	∅	**9.00**
010	1.00	1.00	0.33	∅	∅	∅	∅	**7.00**
011	0.68	0.68	1.00	0.68	0.37	0.77	1.00	**0.00**
012	0.68	0.68	0.77	0.62	0.32	0.42	1.00	**0.00**
013	0.68	1.00	0.51	∅	∅	0.33	0.56	**3.00**
014	1.00	1.00	0.42	∅	∅	∅	∅	**4.00**
015	1.00	∅	∅	∅	∅	∅	∅	**37.00**
016	1.00	∅	∅	∅	∅	∅	∅	**28.00**
017	1.00	∅	∅	∅	∅	∅	∅	**301.00**
018	1.00	∅	∅	∅	∅	∅	∅	**147.00**
019	1.00	∅	0.90	∅	∅	∅	∅	**1206.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

openstacks

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.62	0.62	0.77	1.00	1.00	1.00	1.00	**0.00**
002	0.62	0.59	0.68	1.00	0.77	0.77	1.00	**0.00**
003	0.59	0.59	0.68	1.00	0.77	1.00	1.00	**0.00**
004	0.53	0.53	0.53	0.68	0.54	0.56	1.00	**0.00**
005	0.53	0.53	0.53	0.77	0.59	0.62	1.00	**0.00**
006	0.53	0.53	0.53	0.56	0.59	0.62	1.00	**0.00**
007	0.68	0.62	0.77	1.00	1.00	1.00	1.00	**0.00**
008	0.48	0.48	0.43	0.47	0.44	0.46	1.00	**0.00**
009	0.47	0.48	0.43	0.49	0.45	0.47	1.00	**0.00**
010	0.47	0.48	0.43	0.51	0.45	0.47	1.00	**0.00**
011	0.42	0.43	0.29	0.44	0.38	0.40	1.00	**1.00**
012	0.40	0.41	∅	0.34	0.34	0.35	1.00	**1.00**
013	0.47	0.48	∅	0.44	0.37	0.38	1.00	**3.00**
014	0.46	0.47	∅	0.36	0.36	0.36	1.00	**4.00**
015	0.48	0.50	∅	∅	0.38	0.38	1.00	**6.00**
016	0.46	0.49	∅	∅	0.34	0.35	1.00	**7.00**
017	0.48	0.51	∅	∅	0.35	0.36	1.00	**10.00**
018	0.47	0.50	∅	∅	0.34	0.34	1.00	**13.00**
019	0.48	0.52	∅	∅	0.34	0.35	1.00	**16.00**
020	0.47	0.51	∅	∅	∅	∅	1.00	**20.00**

parcprinter

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.77	0.77	1.00	1.00	1.00	0.38	1.00	**0.00**
002	0.77	0.77	1.00	1.00	1.00	∅	1.00	**0.00**
003	0.77	0.77	1.00	1.00	1.00	∅	1.00	**0.00**
004	0.77	0.77	1.00	0.26	∅	∅	1.00	**0.00**
005	0.77	0.77	1.00	1.00	1.00	∅	1.00	**0.00**
006	0.77	0.77	1.00	∅	1.00	∅	1.00	**0.00**
007	0.77	0.68	1.00	1.00	1.00	0.28	1.00	**0.00**
008	0.77	0.77	1.00	1.00	1.00	∅	1.00	**0.00**
009	0.68	1.00	1.00	1.00	0.46	∅	1.00	**0.00**
010	0.77	0.77	1.00	∅	0.25	∅	1.00	**0.00**
011	0.77	0.62	1.00	1.00	1.00	∅	1.00	**0.00**
012	0.77	1.00	1.00	1.00	1.00	∅	1.00	**0.00**
013	0.68	0.68	1.00	0.77	1.00	∅	1.00	**0.00**
014	0.68	0.77	1.00	∅	∅	∅	1.00	**0.00**
015	0.68	0.68	1.00	∅	∅	∅	1.00	**0.00**
016	0.48	0.68	1.00	∅	∅	∅	0.77	**0.00**
017	0.68	0.77	1.00	∅	0.36	∅	1.00	**0.00**
018	0.77	0.77	1.00	1.00	0.32	∅	1.00	**0.00**
019	0.77	0.68	1.00	0.62	0.30	∅	1.00	**0.00**
020	0.68	0.77	1.00	0.77	∅	∅	1.00	**0.00**

parking

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	1.00	1.00	0.86	0.75	0.51	∅	0.45	**15.00**
002	1.00	0.97	0.79	0.49	0.48	∅	0.47	**13.00**
003	0.98	1.00	0.75	0.49	0.42	∅	0.35	**20.00**
004	0.86	1.00	0.78	0.48	0.40	∅	0.35	**19.00**
005	0.88	1.00	0.79	0.61	0.44	∅	0.49	**19.00**
006	0.92	1.00	0.73	0.48	0.45	∅	0.39	**19.00**
007	0.93	1.00	0.73	0.51	0.43	∅	0.39	**26.00**
008	0.85	0.89	0.71	1.00	0.41	∅	0.38	**23.00**
009	0.98	1.00	0.73	0.91	0.39	∅	∅	**25.00**
010	0.89	1.00	0.52	0.94	0.40	∅	0.35	**26.00**
011	0.94	1.00	0.70	0.54	0.36	∅	∅	**31.00**
012	0.96	1.00	0.77	0.65	0.41	∅	∅	**43.00**
013	1.00	1.00	0.69	0.49	∅	∅	∅	**33.00**
014	0.99	1.00	0.72	0.58	0.39	∅	∅	**36.00**
015	0.84	1.00	0.68	0.41	∅	∅	∅	**53.00**
016	0.81	1.00	0.70	0.82	∅	∅	∅	**46.00**
017	0.95	1.00	0.69	0.44	∅	∅	0.41	**51.00**
018	0.75	1.00	0.70	0.60	∅	∅	∅	**46.00**
019	0.78	1.00	0.63	∅	∅	∅	∅	**57.00**
020	1.00	0.82	0.61	0.59	∅	∅	∅	**55.00**

pegsol

	jasper	mercury	mpc	probe	$LMBFS_g$	LMBFS	yahsp	**best**
001	0.68	0.77	0.31	1.00	0.50	0.77	1.00	**0.00**
002	0.77	0.77	0.59	1.00	0.54	1.00	1.00	**0.00**
003	0.68	0.77	1.00	1.00	1.00	0.50	1.00	**0.00**
004	0.77	0.77	∅	1.00	1.00	0.62	1.00	**0.00**
005	0.77	0.77	1.00	1.00	1.00	0.49	1.00	**0.00**
006	0.68	0.68	1.00	1.00	0.54	0.77	1.00	**0.00**
007	0.77	0.77	1.00	1.00	1.00	1.00	1.00	**0.00**
008	1.00	0.77	1.00	1.00	1.00	0.68	1.00	**0.00**
009	1.00	1.00	0.77	1.00	0.46	0.68	1.00	**0.00**
010	0.77	0.77	1.00	1.00	1.00	1.00	1.00	**0.00**
011	0.77	0.77	1.00	1.00	0.50	0.59	1.00	**0.00**
012	0.77	0.77	0.77	1.00	0.62	0.62	1.00	**0.00**
013	0.77	0.77	1.00	1.00	1.00	1.00	1.00	**0.00**
014	1.00	0.77	0.36	1.00	0.44	0.54	1.00	**0.00**
015	0.68	0.77	1.00	1.00	0.43	0.62	1.00	**0.00**
016	0.77	0.77	1.00	1.00	1.00	0.68	1.00	**0.00**
017	0.68	0.77	1.00	1.00	0.62	1.00	1.00	**0.00**
018	1.00	1.00	1.00	1.00	0.29	0.30	1.00	**0.00**
019	0.59	0.77	1.00	0.56	1.00	1.00	1.00	**0.00**
020	1.00	0.27	0.57	0.34	∅	∅	0.33	**2.00**

scanalyzer

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.44	0.45	0.51	1.00	1.00	1.00	0.77	**0.00**
002	0.62	0.62	1.00	1.00	1.00	1.00	1.00	**0.00**
003	0.77	0.68	1.00	1.00	1.00	1.00	1.00	**0.00**
004	0.68	0.68	1.00	1.00	1.00	1.00	1.00	**0.00**
005	0.54	0.54	1.00	1.00	1.00	1.00	1.00	**0.00**
006	0.54	0.55	0.72	1.00	0.85	1.00	1.00	**2.00**
007	0.50	0.49	0.68	0.77	1.00	1.00	1.00	**0.00**
008	0.49	0.48	0.62	0.54	0.62	0.68	1.00	**1.00**
009	0.62	0.62	1.00	1.00	1.00	1.00	1.00	**0.00**
010	0.59	0.59	1.00	1.00	1.00	1.00	0.37	**0.00**
011	0.59	0.62	0.62	1.00	1.00	1.00	1.00	**0.00**
012	0.68	0.68	1.00	1.00	1.00	1.00	0.51	**0.00**
013	0.77	0.68	1.00	1.00	1.00	1.00	1.00	**0.00**
014	0.54	0.54	1.00	1.00	1.00	1.00	0.31	**0.00**
015	0.54	0.54	0.25	1.00	1.00	1.00	1.00	**0.00**
016	0.50	0.50	0.68	0.77	1.00	1.00	0.27	**0.00**
017	0.50	0.50	0.24	0.77	1.00	1.00	1.00	**0.00**
018	0.55	0.78	0.56	1.00	∅	∅	∅	**119.00**
019	0.71	1.00	0.72	0.50	∅	∅	0.61	**158.00**
020	0.64	1.00	∅	0.66	∅	∅	∅	**133.00**

sokoban

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.77	0.70	∅	1.00	0.28	0.34	0.82	**3.00**
002	0.82	1.00	∅	1.00	0.51	0.66	0.82	**3.00**
003	0.50	0.68	0.28	1.00	0.68	∅	0.62	**0.00**
004	1.00	0.74	∅	0.87	0.40	∅	0.87	**5.00**
005	0.88	0.96	∅	0.96	0.48	∅	1.00	**11.00**
006	0.68	0.59	∅	1.00	∅	∅	1.00	**3.00**
007	0.68	0.68	0.33	1.00	0.32	∅	0.68	**1.00**
008	0.63	0.71	∅	1.00	∅	∅	0.87	**19.00**
009	0.68	0.68	0.37	1.00	0.33	0.33	0.34	**1.00**
010	0.70	0.66	∅	1.00	∅	0.35	0.73	**3.00**
011	0.42	1.00	∅	0.51	∅	∅	0.73	**24.00**
012	0.53	0.45	∅	0.62	∅	0.80	1.00	**45.00**
013	0.74	0.45	∅	0.68	∅	∅	1.00	**30.00**
014	0.63	0.87	∅	0.85	∅	∅	1.00	**14.00**
015	0.78	0.91	∅	0.81	∅	∅	1.00	**155.00**
016	0.71	0.33	0.36	∅	1.00	∅	0.69	**14.00**
017	0.53	0.75	∅	0.88	∅	∅	1.00	**114.00**
018	∅	∅	∅	0.72	∅	∅	1.00	**158.00**
019	0.93	∅	∅	∅	∅	∅	1.00	**451.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

transport

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.49	0.51	0.30	0.59	1.00	1.00	1.00	**0.00**
002	0.48	0.49	0.37	0.51	1.00	1.00	1.00	**0.00**
003	0.44	0.47	∅	0.44	0.77	1.00	1.00	**0.00**
004	0.49	0.54	0.35	0.68	1.00	1.00	1.00	**0.00**
005	0.45	0.53	∅	0.51	1.00	1.00	1.00	**0.00**
006	0.43	0.49	∅	0.41	1.00	1.00	1.00	**0.00**
007	0.35	0.47	∅	0.40	0.77	1.00	1.00	**0.00**
008	0.45	0.47	0.38	0.46	0.77	1.00	1.00	**0.00**
009	0.34	0.45	∅	0.39	0.68	0.77	1.00	**0.00**
010	0.42	0.45	∅	0.38	0.68	0.77	1.00	**0.00**
011	0.42	0.47	∅	0.42	0.77	1.00	1.00	**0.00**
012	0.43	0.47	∅	0.43	0.68	1.00	1.00	**0.00**
013	0.37	0.46	∅	0.40	0.68	0.77	1.00	**0.00**
014	0.24	0.39	∅	0.25	0.49	0.56	1.00	**1.00**
015	0.27	0.44	∅	0.28	0.59	0.68	1.00	**2.00**
016	∅	0.53	∅	0.29	0.72	0.87	1.00	**5.00**
017	0.30	0.43	∅	0.28	0.55	0.62	1.00	**3.00**
018	0.35	0.43	∅	0.33	0.56	0.64	1.00	**3.00**
019	0.32	0.42	∅	0.30	0.53	0.60	1.00	**3.00**
020	0.37	0.55	∅	0.37	0.74	0.88	1.00	**11.00**

visitall

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.77	0.77	0.77	1.00	1.00	1.00	1.00	**0.00**
002	0.68	0.77	0.56	1.00	1.00	1.00	1.00	**0.00**
003	0.68	0.68	0.44	1.00	1.00	1.00	1.00	**0.00**
004	0.68	0.68	0.40	0.77	1.00	1.00	1.00	**0.00**
005	0.62	0.62	0.34	0.62	1.00	1.00	1.00	**0.00**
006	0.62	0.68	0.30	0.54	1.00	1.00	1.00	**0.00**
007	0.56	0.62	0.29	0.47	1.00	1.00	1.00	**0.00**
008	0.53	0.59	0.26	0.43	1.00	1.00	1.00	**0.00**
009	0.51	0.50	0.25	0.40	1.00	1.00	1.00	**0.00**
010	0.49	0.47	∅	0.37	0.77	1.00	1.00	**0.00**
011	0.44	0.45	∅	0.35	0.68	1.00	1.00	**0.00**
012	0.45	0.43	∅	0.33	0.62	0.77	1.00	**0.00**
013	0.43	0.41	∅	0.32	0.59	0.77	1.00	**1.00**
014	0.42	0.41	∅	0.31	0.56	0.68	1.00	**1.00**
015	0.39	0.37	∅	0.30	0.50	0.62	1.00	**1.00**
016	0.43	0.40	∅	0.31	0.54	0.72	1.00	**2.00**
017	0.44	0.41	∅	0.31	0.58	0.77	1.00	**3.00**
018	0.45	0.42	∅	0.32	0.59	0.77	1.00	**4.00**
019	0.43	0.40	∅	0.31	0.56	0.72	1.00	**4.00**
020	0.42	0.40	∅	0.30	0.57	0.74	1.00	**5.00**

woodworking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.72	0.53	0.85	1.00	0.65	∅	0.68	**2.00**
002	0.77	0.53	0.77	1.00	0.45	∅	∅	**3.00**
003	0.85	0.56	1.00	0.80	0.59	∅	0.74	**4.00**
004	0.94	0.58	1.00	0.79	0.62	∅	0.29	**6.00**
005	1.00	0.73	1.00	0.70	0.50	∅	0.32	**8.00**
006	0.95	0.63	1.00	0.95	0.35	∅	0.69	**7.00**
007	0.77	0.50	1.00	0.89	0.55	∅	0.29	**3.00**
008	0.77	0.53	1.00	0.66	0.70	∅	0.58	**3.00**
009	0.82	0.61	1.00	0.89	0.82	∅	0.89	**3.00**
010	0.77	0.77	1.00	1.00	1.00	1.00	1.00	**0.00**
011	1.00	0.69	1.00	0.60	∅	∅	0.34	**7.00**
012	0.95	0.66	1.00	0.73	0.43	∅	0.46	**8.00**
013	0.89	0.51	1.00	0.69	∅	∅	∅	**6.00**
014	0.95	0.55	1.00	0.61	∅	∅	∅	**7.00**
015	1.00	0.54	0.86	0.55	∅	∅	0.48	**11.00**
016	0.97	0.57	1.00	0.72	0.42	∅	0.43	**12.00**
017	0.77	0.69	1.00	0.77	0.49	∅	0.51	**5.00**
018	0.80	0.68	1.00	0.80	∅	∅	∅	**5.00**
019	1.00	0.80	0.96	0.80	0.30	∅	0.68	**9.00**
020	0.78	0.73	1.00	0.72	0.54	∅	0.73	**11.00**

somme métrique m_{time}

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
yahsp	2.34	20.00	2.24	5.56	20.00	19.77	4.04
jasper	12.03	11.02	1.89	15.47	10.09	14.53	18.32
mercury	12.74	13.08	2.57	12.35	10.25	15.23	19.67
probe	18.96	9.79	3.00	4.30	9.06	12.42	11.82
$LMBFS_g$	9.34	15.21	20.00	1.95	9.81	11.69	5.49
mpc	8.10	12.12	20.00	7.56	6.06	20.00	14.28
LMBFS	8.01	17.04	20.00	3.77	10.24	0.66	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
yahsp	19.33	14.84	16.17	20.00	20.00	9.10	**173.38**
jasper	15.89	11.82	12.59	7.43	10.43	17.44	**158.94**
mercury	15.25	12.54	12.14	9.47	10.49	12.37	**158.16**
probe	18.90	18.01	14.92	8.14	9.77	15.66	**154.76**
$LMBFS_g$	13.96	16.47	3.99	14.96	15.55	8.41	**146.84**
mpc	16.38	14.60	1.35	1.40	3.61	19.44	**144.88**
LMBFS	13.86	16.68	2.49	17.16	17.55	1.00	**128.46**

C.3 IPC 7 : métrique $m_{quality}$

barman

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	0.85	0.47	0.91	0.35	0.35	0.72	**262.00**
002	1.00	0.89	0.66	0.90	0.36	0.36	∅	**251.00**
003	1.00	0.84	0.69	0.91	0.40	0.42	0.67	**261.00**
004	1.00	0.90	0.58	0.92	0.41	0.41	0.68	**263.00**
005	1.00	0.89	0.68	0.95	0.34	0.34	∅	**283.00**
006	1.00	0.89	0.70	0.90	0.32	0.32	0.71	**295.00**
007	1.00	0.90	0.69	0.75	0.35	∅	∅	**295.00**
008	1.00	0.90	0.50	0.93	0.30	0.30	∅	**285.00**
009	1.00	0.82	0.49	0.96	0.32	0.34	∅	**333.00**
010	1.00	0.86	0.58	0.96	0.26	∅	0.66	**317.00**
011	1.00	0.89	0.62	0.91	0.32	0.36	0.59	**333.00**
012	1.00	0.86	0.57	0.55	0.31	0.31	∅	**331.00**
013	1.00	0.90	0.65	0.91	0.34	∅	0.79	**383.00**
014	1.00	0.89	0.69	0.97	0.36	0.36	∅	**368.00**
015	1.00	0.88	0.57	0.77	0.35	0.35	∅	**376.00**
016	1.00	0.72	0.53	0.96	0.35	0.35	∅	**378.00**
017	1.00	0.83	0.68	0.95	0.31	∅	∅	**378.00**
018	1.00	0.78	0.65	0.93	0.30	∅	∅	**367.00**
019	1.00	0.85	∅	0.93	0.34	0.34	0.75	**384.00**
020	1.00	0.94	0.47	1.00	0.28	0.27	∅	**356.00**

elevators

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.64	0.80	1.00	0.37	0.36	0.36	0.19	**147.00**
002	0.58	0.68	1.00	0.22	0.24	0.25	0.14	**295.00**
003	0.51	0.79	1.00	0.37	0.30	0.30	0.11	**385.00**
004	0.79	0.68	1.00	0.74	0.20	0.20	0.14	**204.00**
005	0.78	0.98	1.00	0.79	0.28	0.28	0.16	**201.00**
006	0.51	0.67	1.00	0.26	0.18	0.18	0.13	**384.00**
007	0.59	0.71	1.00	0.34	0.21	0.21	0.15	**351.00**
008	0.46	0.59	1.00	0.28	0.15	0.15	0.11	**387.00**
009	0.49	0.50	1.00	0.22	0.16	0.16	0.12	**404.00**
010	0.54	0.72	1.00	0.30	0.21	0.21	0.12	**418.00**
011	0.48	0.51	1.00	0.21	0.19	0.19	0.11	**489.00**
012	0.90	0.89	1.00	0.45	0.27	0.27	0.29	**864.00**
013	0.65	0.77	1.00	0.25	0.22	0.22	0.16	**751.00**
014	0.64	0.72	1.00	0.22	0.20	0.20	0.11	**637.00**
015	0.69	0.64	1.00	0.24	0.16	0.16	0.12	**756.00**
016	0.56	0.53	1.00	0.24	0.23	0.23	0.14	**734.00**
017	0.74	0.92	1.00	0.34	0.24	0.24	0.17	**1034.00**
018	0.61	0.69	1.00	0.23	0.20	0.20	0.15	**1049.00**
019	0.62	0.67	1.00	∅	0.17	0.17	0.14	**1218.00**
020	0.86	0.70	1.00	0.38	0.23	0.23	0.18	**1229.00**

floortile

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.76	0.76	0.38	0.42	1.00	1.00	0.95	**37.00**
002	0.69	0.69	0.35	0.37	0.95	1.00	0.90	**36.00**
003	0.41	0.73	0.38	∅	0.92	1.00	0.96	**46.00**
004	0.62	0.35	0.36	0.30	0.96	1.00	0.86	**48.00**
005	0.38	0.83	0.42	0.50	0.95	0.98	1.00	**52.00**
006	∅	0.78	0.38	0.49	1.00	1.00	0.94	**50.00**
007	∅	∅	0.43	∅	0.97	1.00	∅	**68.00**
008	∅	∅	0.39	∅	1.00	1.00	∅	**61.00**
009	∅	∅	0.42	∅	0.98	1.00	∅	**82.00**
010	∅	∅	0.42	∅	0.98	1.00	∅	**81.00**
011	∅	∅	0.43	∅	1.00	1.00	∅	**81.00**
012	∅	∅	0.40	∅	0.99	1.00	∅	**89.00**
013	∅	∅	0.42	∅	0.99	1.00	∅	**102.00**
014	∅	∅	0.39	∅	0.97	1.00	∅	**100.00**
015	∅	∅	0.41	∅	0.98	1.00	∅	**124.00**
016	∅	∅	0.40	∅	0.97	1.00	∅	**121.00**
017	∅	∅	0.44	∅	0.99	1.00	∅	**138.00**
018	∅	∅	0.39	∅	0.99	1.00	∅	**137.00**
019	∅	∅	0.39	∅	0.99	1.00	∅	**169.00**
020	∅	∅	0.42	∅	0.99	1.00	∅	**169.00**

nomystery

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	0.95	1.00	1.00	**18.00**
002	1.00	1.00	1.00	1.00	0.95	1.00	1.00	**21.00**
003	1.00	1.00	0.93	0.86	∅	0.86	0.86	**25.00**
004	1.00	0.97	0.90	∅	∅	∅	0.88	**28.00**
005	1.00	0.97	0.87	∅	∅	∅	∅	**34.00**
006	1.00	1.00	0.88	∅	∅	∅	∅	**36.00**
007	0.95	1.00	0.93	∅	∅	∅	∅	**40.00**
008	1.00	0.95	0.91	∅	∅	∅	∅	**40.00**
009	0.94	1.00	0.91	∅	∅	∅	∅	**50.00**
010	0.96	1.00	0.96	∅	∅	∅	∅	**49.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**18.00**
012	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**21.00**
013	1.00	1.00	0.89	∅	∅	0.93	0.96	**25.00**
014	1.00	1.00	1.00	∅	∅	∅	∅	**29.00**
015	1.00	∅	∅	∅	∅	∅	∅	**34.00**
016	1.00	∅	∅	∅	∅	∅	∅	**36.00**
017	1.00	∅	∅	∅	∅	∅	∅	**41.00**
018	1.00	∅	∅	∅	∅	∅	∅	**40.00**
019	1.00	∅	1.00	∅	∅	∅	∅	**47.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

openstacks

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.86	1.00	0.33	0.86	0.12	0.12	0.50	**6.00**
002	1.00	1.00	0.44	0.94	0.25	0.25	0.83	**15.00**
003	0.82	1.00	0.32	0.90	0.15	0.15	0.50	**9.00**
004	1.00	0.95	0.54	0.90	0.24	0.24	0.63	**19.00**
005	1.00	0.88	0.39	0.88	0.17	0.17	0.54	**14.00**
006	1.00	1.00	0.54	0.95	0.25	0.25	0.69	**20.00**
007	1.00	0.91	0.45	0.83	0.20	0.20	0.67	**10.00**
008	1.00	1.00	0.77	0.94	0.33	0.33	0.70	**33.00**
009	0.97	1.00	0.71	0.97	0.30	0.30	0.67	**30.00**
010	0.94	1.00	0.78	0.94	0.32	0.32	0.62	**32.00**
011	0.94	1.00	0.62	1.00	0.25	0.25	0.32	**32.00**
012	1.00	1.00	∅	0.95	0.77	0.77	0.79	**100.00**
013	0.88	1.00	∅	0.85	0.28	0.28	0.34	**45.00**
014	1.00	1.00	∅	0.93	0.80	0.80	0.82	**128.00**
015	0.97	1.00	∅	∅	0.39	0.39	0.44	**75.00**
016	0.98	1.00	∅	∅	0.82	0.82	0.82	**155.00**
017	0.96	1.00	∅	∅	0.47	0.47	0.51	**104.00**
018	0.99	1.00	∅	∅	0.86	0.86	0.87	**189.00**
019	0.98	1.00	∅	∅	0.54	0.54	0.58	**135.00**
020	1.00	1.00	∅	∅	∅	∅	0.90	**221.00**

parcprinter

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	0.82	0.73	0.73	0.73	0.73	0.73	**1383121.00**
002	1.00	0.95	0.86	0.86	0.86	∅	0.86	**1852217.00**
003	1.00	0.96	0.93	0.93	0.93	∅	0.93	**2490322.00**
004	1.00	0.99	0.98	0.85	∅	∅	0.77	**2754187.00**
005	1.00	1.00	0.87	0.87	1.00	∅	1.00	**1216462.00**
006	1.00	0.93	0.88	∅	1.00	∅	1.00	**1270874.00**
007	1.00	0.91	0.88	0.88	0.88	0.88	0.88	**2121255.00**
008	1.00	1.00	0.90	0.90	1.00	∅	1.00	**1681282.00**
009	1.00	1.00	0.96	0.96	0.97	∅	0.96	**2388158.00**
010	1.00	0.88	0.91	∅	1.00	∅	1.00	**2021893.00**
011	1.00	0.95	0.86	0.86	0.86	∅	0.86	**1891203.00**
012	1.00	1.00	0.91	0.91	0.91	∅	0.91	**2928369.00**
013	0.90	1.00	0.85	0.85	0.85	∅	0.85	**3435396.00**
014	1.00	0.99	0.98	∅	∅	∅	0.73	**3119803.00**
015	1.00	0.93	0.94	∅	∅	∅	0.75	**3160821.00**
016	1.00	0.97	0.97	∅	∅	∅	0.74	**3671077.00**
017	1.00	0.99	0.85	∅	0.90	∅	0.89	**1566450.00**
018	0.99	0.92	0.99	0.92	1.00	∅	0.99	**2376643.00**
019	1.00	1.00	0.98	0.95	0.99	∅	0.97	**3122628.00**
020	1.00	0.93	0.90	0.76	∅	∅	1.00	**2318715.00**

parking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	0.68	0.55	0.29	0.42	∅	0.59	**38.00**
002	1.00	0.94	0.46	0.27	0.31	∅	0.57	**31.00**
003	1.00	1.00	0.29	0.25	0.37	∅	0.54	**42.00**
004	1.00	0.74	0.45	0.26	0.34	∅	0.58	**46.00**
005	0.83	1.00	0.40	0.27	0.38	∅	0.51	**40.00**
006	1.00	0.81	0.41	0.27	0.43	∅	0.60	**44.00**
007	1.00	0.82	0.45	0.26	0.38	∅	0.77	**58.00**
008	1.00	0.79	0.56	0.39	0.42	∅	0.61	**46.00**
009	1.00	0.74	0.47	0.40	0.35	∅	∅	**46.00**
010	0.72	1.00	0.45	0.43	0.35	∅	0.59	**46.00**
011	1.00	0.63	0.57	0.35	0.28	∅	∅	**45.00**
012	1.00	0.77	0.60	0.40	0.44	∅	∅	**65.00**
013	1.00	0.83	0.43	0.23	∅	∅	∅	**60.00**
014	1.00	0.91	0.32	0.30	0.28	∅	∅	**48.00**
015	0.83	1.00	0.35	0.27	∅	∅	∅	**58.00**
016	0.77	1.00	0.53	0.47	∅	∅	∅	**57.00**
017	1.00	0.68	0.37	0.27	∅	∅	0.61	**52.00**
018	0.93	1.00	0.64	0.47	∅	∅	∅	**67.00**
019	0.93	1.00	0.37	∅	∅	∅	∅	**53.00**
020	1.00	0.55	0.35	0.32	∅	∅	∅	**53.00**

pegsol

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	0.83	1.00	0.91	0.91	0.83	**10.00**
002	1.00	1.00	0.64	0.88	0.70	0.64	0.58	**7.00**
003	1.00	1.00	0.67	0.89	0.73	0.67	0.73	**8.00**
004	1.00	1.00	∅	1.00	0.86	0.92	0.86	**12.00**
005	1.00	1.00	0.69	1.00	1.00	0.75	0.75	**9.00**
006	1.00	1.00	0.54	1.00	0.78	0.70	0.64	**7.00**
007	1.00	1.00	0.62	1.00	0.80	0.62	0.62	**8.00**
008	1.00	1.00	0.92	1.00	0.92	0.85	0.92	**11.00**
009	1.00	1.00	0.53	1.00	0.80	0.62	0.57	**8.00**
010	1.00	1.00	0.58	1.00	0.88	0.70	0.58	**7.00**
011	1.00	1.00	0.57	1.00	0.73	0.67	0.67	**8.00**
012	1.00	1.00	0.67	0.89	0.67	0.62	0.53	**8.00**
013	1.00	1.00	0.90	1.00	0.75	0.75	0.82	**9.00**
014	1.00	1.00	0.44	1.00	0.67	0.67	0.53	**8.00**
015	1.00	1.00	0.58	1.00	0.58	0.58	0.78	**7.00**
016	1.00	1.00	0.50	1.00	0.82	0.75	0.56	**9.00**
017	1.00	1.00	0.35	1.00	0.54	0.44	0.58	**7.00**
018	1.00	1.00	0.57	0.60	0.67	0.63	0.57	**12.00**
019	1.00	0.52	0.50	0.65	0.58	0.61	0.61	**11.00**
020	1.00	0.95	0.61	0.70	∅	∅	0.73	**19.00**

scanalyzer

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	0.54	0.54	0.09	**26.00**
002	1.00	1.00	1.00	1.00	0.40	0.40	0.38	**36.00**
003	1.00	1.00	0.60	0.75	0.95	0.95	0.58	**36.00**
004	1.00	1.00	0.55	0.79	0.88	0.96	0.56	**46.00**
005	1.00	1.00	1.00	1.00	0.35	0.35	0.47	**42.00**
006	1.00	1.00	0.83	0.88	0.83	0.83	0.17	**30.00**
007	1.00	1.00	1.00	1.00	0.33	0.33	0.47	**48.00**
008	1.00	1.00	0.74	0.81	0.55	0.55	0.23	**34.00**
009	1.00	0.97	0.43	0.76	0.93	0.93	0.58	**56.00**
010	1.00	0.62	0.44	0.72	0.76	0.76	0.61	**56.00**
011	1.00	0.97	0.32	0.77	0.89	0.89	0.43	**66.00**
012	1.00	0.63	0.63	0.73	0.83	0.89	0.52	**48.00**
013	1.00	0.80	1.00	0.87	0.95	0.95	0.41	**40.00**
014	1.00	0.65	0.46	0.76	0.76	0.76	0.64	**68.00**
015	1.00	0.76	0.29	0.78	0.93	0.97	0.46	**76.00**
016	1.00	0.71	0.57	0.78	0.80	0.80	0.65	**78.00**
017	1.00	0.75	0.29	0.78	0.90	0.91	0.37	**86.00**
018	1.00	0.87	1.00	0.83	∅	∅	∅	**39.00**
019	1.00	0.86	0.93	0.50	∅	∅	0.29	**51.00**
020	1.00	0.92	∅	0.66	∅	∅	∅	**65.00**

sokoban

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.93	1.00	∅	0.72	0.81	0.74	0.87	**52.00**
002	1.00	1.00	∅	0.89	0.86	0.86	0.86	**50.00**
003	0.79	1.00	0.62	0.66	0.79	∅	0.79	**23.00**
004	1.00	1.00	∅	0.60	0.43	∅	0.85	**33.00**
005	0.96	1.00	∅	0.86	0.56	∅	0.83	**50.00**
006	0.96	1.00	∅	0.52	∅	∅	0.73	**44.00**
007	1.00	1.00	0.84	0.94	1.00	∅	1.00	**31.00**
008	0.97	1.00	∅	0.81	∅	∅	0.94	**60.00**
009	1.00	1.00	0.08	1.00	0.33	0.33	1.00	**2.00**
010	0.96	1.00	∅	0.77	∅	0.56	0.88	**46.00**
011	1.00	0.94	∅	0.81	∅	∅	0.89	**101.00**
012	0.88	1.00	∅	0.69	∅	0.71	0.88	**45.00**
013	1.00	1.00	∅	1.00	∅	∅	1.00	**24.00**
014	1.00	1.00	∅	0.96	∅	∅	0.96	**47.00**
015	0.83	1.00	∅	0.72	∅	∅	0.82	**98.00**
016	1.00	0.56	0.39	∅	0.64	∅	0.82	**18.00**
017	1.00	0.77	∅	0.79	∅	∅	1.00	**61.00**
018	∅	∅	∅	1.00	∅	∅	0.86	**64.00**
019	1.00	∅	∅	∅	∅	∅	0.58	**39.00**
020	∅	∅	∅	∅	∅	∅	∅	∅

transport

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.19	0.22	1.00	0.08	0.09	0.09	0.08	**166.00**
002	0.22	0.20	1.00	0.09	0.10	0.10	0.09	**180.00**
003	0.89	1.00	∅	0.46	0.36	0.36	0.27	**2514.00**
004	0.06	0.06	1.00	0.02	0.02	0.02	0.03	**150.00**
005	0.76	1.00	∅	0.34	0.36	0.36	0.42	**2656.00**
006	0.67	1.00	∅	0.32	0.34	0.34	0.42	**2603.00**
007	0.58	1.00	∅	0.30	0.32	0.32	0.43	**3382.00**
008	0.17	0.19	1.00	0.09	0.09	0.10	0.07	**196.00**
009	0.58	1.00	∅	0.36	0.33	0.33	0.43	**3409.00**
010	0.83	1.00	∅	0.38	0.40	0.41	0.42	**3815.00**
011	0.79	1.00	∅	0.41	0.43	0.43	0.33	**1128.00**
012	0.96	1.00	∅	0.52	0.59	0.59	0.39	**1356.00**
013	0.88	1.00	∅	0.49	0.48	0.48	0.30	**1673.00**
014	0.73	1.00	∅	0.32	0.41	0.41	0.52	**5044.00**
015	0.66	1.00	∅	0.36	0.38	0.38	0.41	**4688.00**
016	∅	1.00	∅	0.25	0.34	0.34	0.49	**4573.00**
017	0.80	1.00	∅	0.37	0.49	0.49	0.37	**4477.00**
018	0.77	1.00	∅	0.48	0.46	0.46	0.44	**3320.00**
019	0.88	1.00	∅	0.34	0.52	0.52	0.42	**3773.00**
020	0.76	1.00	∅	0.42	0.47	0.47	0.35	**3516.00**

visitall

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.98	1.00	0.49	0.99	0.99	0.99	0.91	**153.00**
002	0.86	1.00	0.63	0.97	0.82	0.82	0.87	**201.00**
003	0.84	1.00	0.63	0.97	0.81	0.81	0.86	**261.00**
004	0.82	1.00	0.53	0.96	0.66	0.66	0.85	**327.00**
005	0.81	1.00	0.57	0.97	0.68	0.69	0.86	**405.00**
006	0.83	1.00	0.55	0.97	0.88	0.88	0.82	**490.00**
007	0.82	1.00	0.49	0.97	0.89	0.89	0.83	**581.00**
008	0.79	1.00	0.59	0.82	0.77	0.77	0.84	**682.00**
009	0.79	1.00	0.49	0.82	0.77	0.77	0.84	**793.00**
010	0.79	1.00	∅	0.81	0.79	0.79	0.83	**908.00**
011	0.80	1.00	∅	0.80	0.74	0.74	0.84	**1033.00**
012	0.78	1.00	∅	0.81	0.76	0.76	0.86	**1165.00**
013	0.78	1.00	∅	0.80	0.73	0.73	0.87	**1305.00**
014	0.76	1.00	∅	0.80	0.66	0.66	0.84	**1453.00**
015	0.76	1.00	∅	0.82	0.93	0.93	0.89	**1609.00**
016	0.77	1.00	∅	0.79	0.93	0.93	0.89	**1773.00**
017	0.79	1.00	∅	0.79	0.80	0.80	0.85	**1945.00**
018	0.77	1.00	∅	0.81	0.81	0.81	0.87	**2125.00**
019	0.78	1.00	∅	0.79	0.81	0.81	0.89	**2313.00**
020	0.77	1.00	∅	0.79	0.73	0.73	0.88	**2509.00**

woodworking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.49	0.36	1.00	0.51	0.39	∅	0.34	**630.00**
002	0.44	0.36	1.00	0.47	0.36	∅	∅	**700.00**
003	0.49	0.42	1.00	0.50	0.36	∅	0.31	**670.00**
004	0.55	0.41	1.00	0.56	0.50	∅	0.46	**800.00**
005	0.46	0.39	1.00	0.51	0.40	∅	0.38	**860.00**
006	0.48	0.43	1.00	0.55	0.38	∅	0.43	**860.00**
007	0.41	0.33	1.00	0.42	0.32	∅	0.29	**690.00**
008	0.49	0.39	1.00	0.52	0.37	∅	0.35	**730.00**
009	0.49	0.47	1.00	0.48	0.37	∅	0.47	**620.00**
010	0.10	0.10	1.00	0.10	0.06	0.06	0.09	**5.00**
011	0.41	0.41	1.00	0.44	∅	∅	0.33	**790.00**
012	0.51	0.44	1.00	0.55	0.38	∅	0.38	**970.00**
013	0.44	0.38	1.00	0.47	∅	∅	∅	**880.00**
014	0.49	0.50	1.00	0.55	∅	∅	∅	**770.00**
015	0.41	0.33	1.00	0.48	∅	∅	0.39	**1010.00**
016	0.47	0.40	1.00	0.51	0.38	∅	0.41	**950.00**
017	0.59	0.63	1.00	0.65	0.61	∅	0.56	**610.00**
018	0.37	0.31	1.00	0.37	∅	∅	∅	**490.00**
019	0.46	0.48	1.00	0.50	0.37	∅	0.43	**630.00**
020	0.46	0.46	1.00	0.50	0.41	∅	0.38	**670.00**

Somme métrique $m_{quality}$

	barman	elevators	floortile	nomystery	openstacks	parcprinter	parking
jasper	20.00	12.64	2.86	18.86	19.30	19.87	19.01
mercury	17.29	14.15	4.13	13.89	19.73	19.14	16.89
probe	17.97	6.48	2.07	4.86	12.85	12.23	6.16
mpc	11.45	20.00	8.01	14.18	5.90	18.12	9.03
yahsp	5.58	2.95	5.61	6.70	12.74	17.81	5.99
$LMBFS_g$	6.65	4.37	19.54	3.90	7.51	13.88	4.75
LMBFS	5.18	4.40	19.98	5.79	7.51	1.61	0.00

	pegsol	scanalyzer	sokoban	transport	visitall	woodworking	**total**
jasper	20.00	20.00	17.28	12.19	16.08	9.01	**207.10**
mercury	19.47	17.51	16.28	16.68	20.00	8.01	**203.19**
probe	18.60	16.16	13.75	6.41	17.25	9.66	**144.44**
mpc	11.72	13.08	1.93	4.00	4.97	20.00	**142.39**
yahsp	13.46	7.91	16.57	6.69	17.19	6.00	**125.19**
$LMBFS_g$	14.36	12.57	5.43	6.99	15.96	5.68	**121.59**
LMBFS	13.07	12.78	3.21	7.01	15.97	0.06	**96.58**

C.4 IPC 8 : instances résolues

barman

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
003	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
005	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
006	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
007	1.00	1.00	∅	1.00	1.00	∅	1.00	**1.00**
008	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	∅	∅	**1.00**
013	1.00	1.00	∅	1.00	1.00	∅	∅	**1.00**
014	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
015	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
016	1.00	1.00	1.00	1.00	1.00	∅	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	1.00	∅	**1.00**
018	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
020	1.00	1.00	∅	1.00	∅	∅	∅	**1.00**

childsnack

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
002	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
003	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
004	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
005	∅	∅	∅	∅	∅	1.00	∅	**1.00**
006	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
007	∅	∅	1.00	∅	∅	1.00	∅	**1.00**
008	∅	∅	∅	∅	∅	1.00	∅	**1.00**
009	∅	∅	∅	∅	∅	1.00	∅	**1.00**
010	∅	∅	∅	∅	∅	1.00	∅	**1.00**
011	∅	∅	∅	∅	∅	∅	∅	∅
012	∅	∅	∅	∅	∅	1.00	∅	**1.00**
013	∅	∅	∅	∅	∅	1.00	∅	**1.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	∅	∅	∅	∅	∅
016	∅	∅	1.00	∅	∅	∅	∅	**1.00**
017	∅	∅	∅	∅	∅	∅	∅	∅
018	∅	∅	∅	∅	∅	∅	∅	∅
019	∅	∅	∅	∅	∅	∅	∅	∅
020	∅	∅	∅	∅	∅	∅	∅	∅

floortile

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
003	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
004	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
005	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
006	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
007	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
008	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
009	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
010	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
011	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
012	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
013	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
014	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
015	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
016	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
017	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
018	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
019	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**
020	∅	∅	1.00	∅	1.00	1.00	∅	**1.00**

ged

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	1.00	1.00	∅	1.00	1.00	**1.00**
013	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
014	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
020	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**

hiking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
007	1.00	1.00	∅	1.00	1.00	1.00	∅	**1.00**
008	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
013	1.00	∅	∅	1.00	1.00	1.00	1.00	**1.00**
014	1.00	∅	∅	1.00	1.00	1.00	∅	**1.00**
015	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
017	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
020	∅	∅	∅	1.00	1.00	1.00	1.00	**1.00**

parking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
002	1.00	1.00	1.00	1.00	1.00	∅	∅	**1.00**
003	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
004	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
005	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
006	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
007	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
008	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
009	1.00	1.00	1.00	1.00	∅	∅	∅	**1.00**
010	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
011	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
012	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
013	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
014	1.00	1.00	1.00	∅	∅	∅	∅	**1.00**
015	1.00	1.00	∅	∅	∅	∅	∅	**1.00**
016	1.00	1.00	∅	∅	∅	∅	∅	**1.00**
017	1.00	1.00	∅	1.00	∅	∅	∅	**1.00**
018	1.00	1.00	∅	∅	∅	∅	∅	**1.00**
019	1.00	1.00	∅	1.00	∅	∅	∅	**1.00**
020	1.00	1.00	∅	1.00	∅	∅	∅	**1.00**

thoughtful

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	∅	∅	1.00	1.00	1.00	1.00	1.00	**1.00**
002	∅	∅	1.00	1.00	1.00	1.00	1.00	**1.00**
003	∅	∅	1.00	1.00	1.00	1.00	1.00	**1.00**
004	∅	∅	1.00	1.00	1.00	1.00	1.00	**1.00**
005	∅	∅	1.00	1.00	1.00	1.00	1.00	**1.00**
006	∅	∅	∅	1.00	1.00	1.00	1.00	**1.00**
007	∅	∅	∅	1.00	∅	∅	∅	**1.00**
008	∅	∅	∅	1.00	∅	∅	1.00	**1.00**
009	∅	∅	∅	1.00	∅	∅	∅	**1.00**
010	∅	∅	∅	1.00	∅	∅	∅	**1.00**
011	∅	∅	∅	1.00	1.00	1.00	∅	**1.00**
012	∅	∅	∅	1.00	1.00	1.00	∅	**1.00**
013	∅	∅	∅	1.00	∅	∅	∅	**1.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	1.00	1.00	1.00	∅	**1.00**
016	∅	∅	∅	∅	∅	∅	1.00	**1.00**
017	∅	∅	∅	1.00	∅	∅	∅	**1.00**
018	∅	∅	∅	1.00	∅	∅	∅	**1.00**
019	∅	∅	∅	1.00	1.00	1.00	1.00	**1.00**
020	∅	∅	∅	1.00	∅	∅	1.00	**1.00**

transport

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
005	∅	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
009	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
011	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
013	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
014	∅	1.00	∅	∅	1.00	1.00	1.00	**1.00**
015	∅	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
016	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
017	∅	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
020	∅	1.00	∅	1.00	1.00	1.00	1.00	**1.00**

visitall

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
002	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
003	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
004	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
005	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
006	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
007	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
008	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
009	1.00	1.00	∅	1.00	1.00	1.00	1.00	**1.00**
010	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
011	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
012	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
013	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
014	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
015	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
016	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
017	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
018	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
019	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**
020	1.00	1.00	∅	∅	1.00	1.00	1.00	**1.00**

Nombre d'instances résolues

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
LMBFS	13.00	12.00	20.00	20.00	20.00	0.00	10.00	20.00	20.00	**135.00**
$LMBFS_g$	19.00	0.00	20.00	19.00	20.00	1.00	10.00	20.00	20.00	**129.00**
mercury	20.00	0.00	2.00	20.00	17.00	20.00	0.00	20.00	20.00	**119.00**
probe	20.00	0.00	2.00	20.00	20.00	12.00	18.00	18.00	9.00	**119.00**
jasper	20.00	0.00	2.00	20.00	19.00	20.00	0.00	15.00	20.00	**116.00**
yahsp	5.00	0.00	2.00	20.00	17.00	0.00	10.00	20.00	20.00	**94.00**
mpc	11.00	7.00	20.00	18.00	9.00	14.00	5.00	0.00	0.00	**84.00**

C.5 IPC 8 : métrique m_{time}

barman

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.59	0.59	0.25	1.00	0.45	0.53	0.24	**0.00**
002	0.94	1.00	0.37	0.50	0.37	0.42	∅	**6.00**
003	1.00	0.80	∅	0.80	0.38	0.39	∅	**4.00**
004	0.68	0.62	0.26	1.00	0.38	0.44	∅	**0.00**
005	0.51	0.49	∅	1.00	0.34	0.38	∅	**0.00**
006	0.54	0.35	∅	1.00	0.54	0.53	∅	**0.00**
007	0.37	1.00	∅	0.74	0.31	∅	0.29	**5.00**
008	0.53	0.85	∅	1.00	0.39	0.44	∅	**10.00**
009	0.59	0.53	0.29	1.00	0.36	0.28	∅	**0.00**
010	0.49	0.48	0.24	1.00	0.31	0.34	∅	**1.00**
011	0.64	0.73	0.30	1.00	0.49	0.55	0.40	**3.00**
012	0.51	0.49	∅	1.00	0.33	∅	∅	**0.00**
013	0.50	0.35	∅	1.00	0.33	∅	∅	**0.00**
014	0.68	0.68	0.24	1.00	0.33	∅	∅	**0.00**
015	0.56	0.53	0.26	1.00	0.32	∅	∅	**0.00**
016	0.49	0.53	0.33	1.00	0.30	∅	0.25	**0.00**
017	0.56	0.54	0.24	1.00	0.50	0.59	∅	**0.00**
018	0.59	0.59	0.25	1.00	0.31	0.34	0.26	**0.00**
019	0.48	0.49	∅	1.00	0.36	0.24	∅	**1.00**
020	0.65	1.00	∅	0.44	∅	∅	∅	**4.00**

childsnack

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	∅	∅	1.00	∅	∅	1.00	∅	**0.00**
002	∅	∅	1.00	∅	∅	1.00	∅	**0.00**
003	∅	∅	1.00	∅	∅	0.77	∅	**0.00**
004	∅	∅	1.00	∅	∅	0.53	∅	**0.00**
005	∅	∅	∅	∅	∅	1.00	∅	**9.00**
006	∅	∅	1.00	∅	∅	0.38	∅	**0.00**
007	∅	∅	1.00	∅	∅	0.36	∅	**0.00**
008	∅	∅	∅	∅	∅	1.00	∅	**8.00**
009	∅	∅	∅	∅	∅	1.00	∅	**946.00**
010	∅	∅	∅	∅	∅	1.00	∅	**173.00**
011	∅	∅	∅	∅	∅	∅	∅	∅
012	∅	∅	∅	∅	∅	1.00	∅	**411.00**
013	∅	∅	∅	∅	∅	1.00	∅	**1437.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	∅	∅	∅	∅	∅
016	∅	∅	1.00	∅	∅	∅	∅	**167.00**
017	∅	∅	∅	∅	∅	∅	∅	∅
018	∅	∅	∅	∅	∅	∅	∅	∅
019	∅	∅	∅	∅	∅	∅	∅	∅
020	∅	∅	∅	∅	∅	∅	∅	∅

floortile

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.77	0.50	1.00	0.33	1.00	1.00	0.43	**0.00**
002	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
003	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
004	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
005	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
006	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
007	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
008	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
009	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
010	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
011	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
012	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
013	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
014	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
015	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
016	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
017	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
018	0.43	1.00	1.00	1.00	1.00	1.00	0.49	**0.00**
019	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**
020	∅	∅	1.00	∅	1.00	1.00	∅	**0.00**

ged

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.62	0.56	0.49	1.00	0.51	0.59	1.00	**1.00**
002	0.62	0.56	0.42	0.77	0.51	0.62	1.00	**1.00**
003	0.62	0.54	0.41	1.00	1.00	1.00	1.00	**0.00**
004	0.68	0.62	0.37	0.68	0.34	0.38	1.00	**0.00**
005	0.54	0.54	0.26	0.42	0.45	0.54	1.00	**0.00**
006	0.68	0.62	0.34	0.38	0.33	0.37	1.00	**2.00**
007	0.62	0.54	0.36	0.45	0.56	0.68	1.00	**0.00**
008	0.62	0.56	0.39	1.00	0.68	0.77	1.00	**1.00**
009	0.82	0.70	0.44	0.82	0.48	0.58	1.00	**3.00**
010	0.62	0.62	0.27	0.56	0.53	0.62	1.00	**1.00**
011	0.72	0.72	0.40	0.46	0.43	0.51	1.00	**2.00**
012	0.51	0.51	0.26	0.38	∅	0.24	1.00	**0.00**
013	1.00	1.00	0.35	0.71	0.36	0.41	0.85	**10.00**
014	1.00	0.96	0.38	0.46	0.37	0.44	0.74	**11.00**
015	0.77	0.72	0.29	0.41	0.30	0.34	1.00	**4.00**
016	1.00	0.92	0.31	0.45	0.34	0.39	0.84	**9.00**
017	0.97	1.00	∅	0.48	0.48	0.60	0.74	**14.00**
018	0.98	1.00	∅	0.42	0.38	0.45	0.83	**17.00**
019	0.62	0.62	0.38	0.62	0.68	1.00	1.00	**0.00**
020	0.62	0.59	0.32	0.50	0.77	1.00	1.00	**1.00**

hiking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.77	0.77	1.00	1.00	1.00	1.00	1.00	**0.00**
002	0.68	0.68	0.42	1.00	1.00	1.00	1.00	**0.00**
003	0.53	0.42	0.25	0.68	1.00	1.00	1.00	**0.00**
004	0.45	0.33	∅	0.47	1.00	1.00	0.45	**0.00**
005	0.39	0.31	∅	0.40	1.00	1.00	0.30	**0.00**
006	0.34	0.25	∅	0.36	1.00	1.00	∅	**1.00**
007	0.30	0.24	∅	0.34	0.77	1.00	∅	**1.00**
008	0.59	0.59	1.00	0.77	1.00	1.00	1.00	**0.00**
009	0.48	0.50	0.77	0.62	1.00	1.00	1.00	**0.00**
010	0.39	0.45	0.28	0.37	1.00	1.00	1.00	**0.00**
011	0.34	0.36	∅	0.34	0.77	1.00	0.45	**1.00**
012	0.31	0.29	∅	0.35	0.77	1.00	0.29	**2.00**
013	0.30	∅	∅	0.33	0.77	1.00	0.28	**3.00**
014	0.32	∅	∅	0.33	0.80	1.00	∅	**5.00**
015	0.46	0.47	0.68	1.00	1.00	1.00	1.00	**0.00**
016	0.36	0.42	0.40	0.36	1.00	1.00	1.00	**0.00**
017	0.35	0.31	0.35	0.34	0.85	1.00	0.48	**2.00**
018	0.29	0.27	∅	0.33	0.80	1.00	0.51	**4.00**
019	0.33	0.31	∅	0.35	0.83	1.00	0.79	**8.00**
020	∅	∅	∅	0.33	0.83	1.00	0.33	**13.00**

parking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.89	1.00	0.67	0.45	∅	∅	∅	**31.00**
002	0.98	1.00	0.72	0.54	0.40	∅	∅	**35.00**
003	0.94	1.00	0.65	0.62	∅	∅	∅	**40.00**
004	0.87	1.00	0.54	0.43	∅	∅	∅	**48.00**
005	0.70	1.00	0.62	0.46	∅	∅	∅	**42.00**
006	0.86	1.00	0.61	0.42	∅	∅	∅	**50.00**
007	1.00	0.98	0.76	0.54	∅	∅	∅	**109.00**
008	1.00	0.98	0.71	0.58	∅	∅	∅	**81.00**
009	0.97	1.00	0.65	0.51	∅	∅	∅	**86.00**
010	0.82	1.00	0.65	∅	∅	∅	∅	**79.00**
011	0.79	1.00	0.59	∅	∅	∅	∅	**78.00**
012	0.69	1.00	0.59	∅	∅	∅	∅	**79.00**
013	0.71	1.00	0.65	∅	∅	∅	∅	**125.00**
014	0.72	1.00	0.60	∅	∅	∅	∅	**90.00**
015	0.85	1.00	∅	∅	∅	∅	∅	**163.00**
016	0.98	1.00	∅	∅	∅	∅	∅	**220.00**
017	1.00	0.95	∅	0.51	∅	∅	∅	**145.00**
018	0.78	1.00	∅	∅	∅	∅	∅	**168.00**
019	0.73	1.00	∅	0.62	∅	∅	∅	**200.00**
020	0.82	1.00	∅	0.68	∅	∅	∅	**212.00**

thoughtful

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	∅	∅	1.00	1.00	1.00	1.00	1.00	**0.00**
002	∅	∅	1.00	1.00	1.00	1.00	1.00	**0.00**
003	∅	∅	0.77	1.00	1.00	1.00	1.00	**0.00**
004	∅	∅	0.77	1.00	1.00	1.00	1.00	**0.00**
005	∅	∅	1.00	1.00	1.00	1.00	1.00	**0.00**
006	∅	∅	∅	1.00	0.38	0.59	0.43	**5.00**
007	∅	∅	∅	1.00	∅	∅	∅	**6.00**
008	∅	∅	∅	1.00	∅	∅	1.00	**9.00**
009	∅	∅	∅	1.00	∅	∅	∅	**7.00**
010	∅	∅	∅	1.00	∅	∅	∅	**7.00**
011	∅	∅	∅	1.00	0.66	0.67	∅	**6.00**
012	∅	∅	∅	1.00	0.71	0.71	∅	**7.00**
013	∅	∅	∅	1.00	∅	∅	∅	**11.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	1.00	0.44	0.43	∅	**1.00**
016	∅	∅	∅	∅	∅	∅	1.00	**17.00**
017	∅	∅	∅	1.00	∅	∅	∅	**8.00**
018	∅	∅	∅	1.00	∅	∅	∅	**7.00**
019	∅	∅	∅	1.00	0.56	0.56	0.57	**3.00**
020	∅	∅	∅	1.00	∅	∅	0.82	**17.00**

transport

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.40	0.46	∅	0.39	0.62	0.77	1.00	**0.00**
002	0.32	0.43	∅	0.34	0.56	0.68	1.00	**0.00**
003	0.39	0.45	∅	0.35	0.62	0.77	1.00	**0.00**
004	0.29	0.45	∅	0.29	0.54	0.62	1.00	**3.00**
005	∅	0.49	∅	0.29	0.59	0.72	1.00	**6.00**
006	0.29	0.45	∅	0.29	0.54	0.62	1.00	**3.00**
007	0.34	0.43	∅	0.28	0.52	0.60	1.00	**4.00**
008	0.30	0.44	∅	∅	0.53	0.62	1.00	**6.00**
009	0.34	0.43	∅	0.28	0.52	0.60	1.00	**4.00**
010	0.31	0.43	∅	0.31	0.53	0.61	1.00	**4.00**
011	0.33	0.43	∅	0.34	0.54	0.68	1.00	**0.00**
012	0.36	0.42	∅	0.31	0.53	0.62	1.00	**1.00**
013	0.40	0.46	∅	0.42	0.62	0.77	1.00	**0.00**
014	∅	0.48	∅	∅	0.58	0.69	1.00	**4.00**
015	∅	0.49	∅	0.30	0.58	0.69	1.00	**5.00**
016	0.30	0.45	∅	0.29	0.56	0.65	1.00	**2.00**
017	∅	0.43	∅	0.29	0.51	0.58	1.00	**4.00**
018	0.34	0.45	∅	0.32	0.54	0.63	1.00	**6.00**
019	0.35	0.44	∅	0.33	0.54	0.61	1.00	**3.00**
020	∅	0.44	∅	0.32	0.53	0.62	1.00	**5.00**

visitall

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.47	0.49	∅	0.37	0.77	1.00	1.00	**0.00**
002	0.46	0.47	∅	0.36	0.77	1.00	1.00	**0.00**
003	0.47	0.46	∅	0.35	0.68	1.00	1.00	**0.00**
004	0.46	0.45	∅	0.34	0.68	0.77	1.00	**1.00**
005	0.50	0.46	∅	0.33	0.68	0.96	1.00	**10.00**
006	0.49	0.45	∅	0.33	0.66	0.93	1.00	**10.00**
007	0.48	0.44	∅	0.32	0.64	0.87	1.00	**10.00**
008	0.48	0.45	∅	0.33	0.64	0.87	1.00	**12.00**
009	0.48	0.45	∅	0.32	0.62	0.88	1.00	**13.00**
010	0.45	0.45	∅	∅	0.63	0.86	1.00	**13.00**
011	0.48	0.44	∅	∅	0.65	0.89	1.00	**16.00**
012	0.44	0.44	∅	∅	0.63	0.86	1.00	**15.00**
013	0.45	0.44	∅	∅	0.67	0.93	1.00	**20.00**
014	0.49	0.44	∅	∅	0.62	0.88	1.00	**21.00**
015	0.46	0.45	∅	∅	0.63	0.86	1.00	**22.00**
016	0.47	0.44	∅	∅	0.65	0.90	1.00	**26.00**
017	0.50	0.45	∅	∅	0.66	0.91	1.00	**30.00**
018	0.46	0.45	∅	∅	0.64	0.88	1.00	**30.00**
019	0.50	0.47	∅	∅	0.68	0.95	1.00	**40.00**
020	0.51	0.46	∅	∅	0.65	0.90	1.00	**39.00**

somme métrique m_{time}

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
LMBFS	5.46	10.03	20.00	11.53	20.00	0.00	7.97	13.15	18.08	**106.21**
$LMBFS_g$	7.09	0.00	20.00	9.49	18.18	0.40	7.74	11.11	13.26	**87.27**
yahsp	1.44	0.00	0.92	18.99	11.90	0.00	8.83	20.00	20.00	**82.09**
probe	18.50	0.00	1.33	11.97	10.08	6.36	18.00	5.76	3.07	**75.07**
mercury	12.63	0.00	1.50	13.93	6.97	19.91	0.00	8.95	9.08	**72.97**
jasper	11.90	0.00	1.20	14.65	7.98	17.10	0.00	5.06	9.52	**67.40**
mpc	3.03	7.00	20.00	6.45	5.13	9.01	4.54	0.00	0.00	**55.16**

C.6 IPC 8 : métrique $m_{quality}$

barman

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	0.80	0.49	0.77	0.32	0.32	0.45	**164.00**
002	1.00	0.66	0.51	0.70	0.31	0.31	∅	**160.00**
003	0.97	1.00	∅	0.71	0.24	0.23	∅	**151.00**
004	1.00	0.69	0.46	0.67	0.27	0.27	∅	**139.00**
005	1.00	0.76	∅	0.89	0.26	0.26	∅	**155.00**
006	1.00	0.65	∅	0.84	0.31	0.31	∅	**152.00**
007	1.00	0.79	∅	0.61	0.26	∅	0.40	**157.00**
008	1.00	0.60	∅	0.59	0.31	0.31	∅	**175.00**
009	1.00	0.56	0.40	0.80	0.27	0.29	∅	**137.00**
010	1.00	0.63	0.40	0.63	0.26	0.26	∅	**155.00**
011	1.00	0.73	0.42	0.87	0.31	0.32	0.45	**163.00**
012	1.00	0.83	∅	0.85	0.26	∅	∅	**167.00**
013	1.00	0.70	∅	0.88	0.29	∅	∅	**175.00**
014	1.00	0.74	0.43	0.94	0.29	∅	∅	**153.00**
015	1.00	0.56	0.54	0.81	0.26	∅	∅	**156.00**
016	1.00	0.67	0.36	0.78	0.26	∅	0.44	**160.00**
017	1.00	0.67	0.48	0.83	0.29	0.29	∅	**141.00**
018	1.00	0.70	0.44	0.84	0.27	0.27	0.46	**158.00**
019	1.00	0.54	∅	0.81	0.31	0.31	∅	**166.00**
020	1.00	0.77	∅	0.75	∅	∅	∅	**169.00**

childsnack

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	∅	∅	1.00	∅	∅	0.68	∅	**36.00**
002	∅	∅	1.00	∅	∅	0.72	∅	**38.00**
003	∅	∅	1.00	∅	∅	0.66	∅	**38.00**
004	∅	∅	1.00	∅	∅	0.68	∅	**43.00**
005	∅	∅	∅	∅	∅	1.00	∅	**68.00**
006	∅	∅	1.00	∅	∅	0.69	∅	**47.00**
007	∅	∅	1.00	∅	∅	0.64	∅	**47.00**
008	∅	∅	∅	∅	∅	1.00	∅	**73.00**
009	∅	∅	∅	∅	∅	1.00	∅	**78.00**
010	∅	∅	∅	∅	∅	1.00	∅	**78.00**
011	∅	∅	∅	∅	∅	∅	∅	∅
012	∅	∅	∅	∅	∅	1.00	∅	**83.00**
013	∅	∅	∅	∅	∅	1.00	∅	**88.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	∅	∅	∅	∅	∅
016	∅	∅	1.00	∅	∅	∅	∅	**67.00**
017	∅	∅	∅	∅	∅	∅	∅	∅
018	∅	∅	∅	∅	∅	∅	∅	∅
019	∅	∅	∅	∅	∅	∅	∅	∅
020	∅	∅	∅	∅	∅	∅	∅	∅

floortile

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.64	0.64	0.38	0.35	0.90	1.00	0.90	**36.00**
002	∅	∅	0.39	∅	1.00	1.00	∅	**67.00**
003	∅	∅	0.46	∅	0.99	1.00	∅	**91.00**
004	∅	∅	0.43	∅	0.98	1.00	∅	**82.00**
005	∅	∅	0.43	∅	1.00	1.00	∅	**98.00**
006	∅	∅	0.44	∅	0.99	1.00	∅	**98.00**
007	∅	∅	0.37	∅	0.99	1.00	∅	**66.00**
008	∅	∅	0.38	∅	1.00	1.00	∅	**79.00**
009	∅	∅	0.42	∅	1.00	1.00	∅	**105.00**
010	∅	∅	0.41	∅	0.99	1.00	∅	**96.00**
011	∅	∅	0.37	∅	1.00	1.00	∅	**80.00**
012	∅	∅	0.40	∅	1.00	1.00	∅	**100.00**
013	∅	∅	0.38	∅	0.98	1.00	∅	**95.00**
014	∅	∅	0.39	∅	1.00	1.00	∅	**69.00**
015	∅	∅	0.42	∅	1.00	1.00	∅	**89.00**
016	∅	∅	0.43	∅	0.99	1.00	∅	**112.00**
017	∅	∅	0.42	∅	0.98	1.00	∅	**96.00**
018	0.75	0.75	0.38	0.47	1.00	1.00	0.90	**36.00**
019	∅	∅	0.44	∅	1.00	1.00	∅	**99.00**
020	∅	∅	0.41	∅	0.99	1.00	∅	**99.00**

ged

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	0.18	0.53	0.03	0.03	0.07	**18.00**
002	1.00	0.95	0.08	0.27	0.03	0.03	0.09	**18.00**
003	0.88	1.00	0.16	0.75	0.03	0.03	0.05	**21.00**
004	1.00	0.95	0.09	0.24	0.04	0.04	0.08	**18.00**
005	0.84	1.00	0.14	0.50	0.03	0.03	0.08	**26.00**
006	0.91	1.00	0.21	0.18	0.03	0.03	0.08	**29.00**
007	0.86	1.00	0.16	0.32	0.03	0.03	0.06	**18.00**
008	1.00	1.00	0.27	0.74	0.03	0.03	0.07	**20.00**
009	0.78	1.00	0.18	0.54	0.02	0.02	0.05	**21.00**
010	0.88	1.00	0.13	0.43	0.03	0.03	0.07	**23.00**
011	0.96	1.00	0.12	0.25	0.03	0.03	0.06	**23.00**
012	0.90	1.00	0.19	0.22	∅	0.02	0.08	**26.00**
013	1.00	0.91	0.14	0.35	0.02	0.02	0.05	**32.00**
014	0.88	1.00	0.12	0.18	0.01	0.01	0.06	**30.00**
015	0.71	1.00	0.13	0.22	0.01	0.02	0.05	**27.00**
016	0.71	1.00	0.11	0.21	0.01	0.02	0.05	**25.00**
017	0.82	1.00	∅	0.23	0.01	0.01	0.04	**31.00**
018	1.00	0.98	∅	0.28	0.02	0.02	0.07	**41.00**
019	0.94	1.00	0.15	0.19	0.02	0.02	0.06	**15.00**
020	1.00	0.81	0.10	0.20	0.03	0.03	0.05	**17.00**

hiking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	1.00	0.67	1.00	0.92	0.92	0.41	**12.00**
002	1.00	1.00	0.41	0.68	1.00	1.00	0.39	**19.00**
003	1.00	0.96	0.35	0.57	0.70	0.70	0.15	**26.00**
004	1.00	0.97	∅	0.61	0.66	0.66	0.18	**35.00**
005	0.83	1.00	∅	0.60	0.59	0.59	0.03	**40.00**
006	1.00	0.86	∅	0.73	0.69	0.72	∅	**57.00**
007	0.91	0.64	∅	1.00	0.81	0.81	∅	**79.00**
008	1.00	1.00	0.48	1.00	0.93	0.93	0.82	**14.00**
009	1.00	1.00	0.45	0.95	0.60	0.60	0.22	**21.00**
010	1.00	0.91	0.69	0.58	0.56	0.57	0.13	**29.00**
011	0.89	1.00	∅	0.67	0.57	0.60	0.13	**42.00**
012	0.68	1.00	∅	0.90	0.59	0.60	0.04	**55.00**
013	0.73	∅	∅	1.00	0.79	0.79	0.01	**80.00**
014	1.00	∅	∅	0.93	0.84	0.84	∅	**101.00**
015	1.00	1.00	0.44	0.89	0.89	0.89	0.33	**16.00**
016	1.00	0.92	0.42	0.73	0.55	0.55	0.35	**24.00**
017	1.00	0.89	0.42	0.60	0.49	0.51	0.12	**33.00**
018	0.62	1.00	∅	0.90	0.73	0.73	0.17	**66.00**
019	0.76	1.00	∅	0.80	0.55	0.55	0.04	**64.00**
020	∅	∅	∅	1.00	0.84	0.84	0.02	**103.00**

parking

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	1.00	0.97	0.52	0.36	∅	∅	∅	**66.00**
002	1.00	0.75	0.67	0.29	0.41	∅	∅	**58.00**
003	1.00	0.72	0.37	0.50	∅	∅	∅	**50.00**
004	1.00	0.80	0.52	0.30	∅	∅	∅	**66.00**
005	0.90	1.00	0.47	0.38	∅	∅	∅	**77.00**
006	0.95	1.00	0.46	0.46	∅	∅	∅	**74.00**
007	0.83	1.00	0.51	0.50	∅	∅	∅	**71.00**
008	0.80	1.00	0.37	0.39	∅	∅	∅	**66.00**
009	1.00	0.93	0.61	0.41	∅	∅	∅	**81.00**
010	0.98	1.00	0.62	∅	∅	∅	∅	**81.00**
011	0.93	1.00	0.59	∅	∅	∅	∅	**85.00**
012	0.55	1.00	0.39	∅	∅	∅	∅	**71.00**
013	0.80	1.00	0.61	∅	∅	∅	∅	**103.00**
014	0.85	1.00	0.46	∅	∅	∅	∅	**77.00**
015	0.84	1.00	∅	∅	∅	∅	∅	**87.00**
016	1.00	0.91	∅	∅	∅	∅	∅	**105.00**
017	1.00	0.85	∅	0.61	∅	∅	∅	**88.00**
018	0.67	1.00	∅	∅	∅	∅	∅	**78.00**
019	1.00	0.98	∅	0.52	∅	∅	∅	**86.00**
020	0.94	1.00	∅	0.52	∅	∅	∅	**98.00**

thoughtful

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	∅	∅	0.69	1.00	0.76	0.88	0.94	**29.00**
002	∅	∅	0.81	1.00	0.64	0.64	0.81	**25.00**
003	∅	∅	0.63	1.00	0.82	0.82	0.97	**31.00**
004	∅	∅	0.59	1.00	0.69	0.69	0.90	**27.00**
005	∅	∅	0.46	1.00	0.89	0.91	0.86	**31.00**
006	∅	∅	∅	1.00	0.63	0.63	0.44	**120.00**
007	∅	∅	∅	1.00	∅	∅	∅	**129.00**
008	∅	∅	∅	1.00	∅	∅	0.57	**178.00**
009	∅	∅	∅	1.00	∅	∅	∅	**171.00**
010	∅	∅	∅	1.00	∅	∅	∅	**132.00**
011	∅	∅	∅	1.00	0.67	0.67	∅	**161.00**
012	∅	∅	∅	1.00	0.91	0.91	∅	**170.00**
013	∅	∅	∅	1.00	∅	∅	∅	**176.00**
014	∅	∅	∅	∅	∅	∅	∅	∅
015	∅	∅	∅	1.00	0.62	0.62	∅	**134.00**
016	∅	∅	∅	∅	∅	∅	1.00	**196.00**
017	∅	∅	∅	1.00	∅	∅	∅	**177.00**
018	∅	∅	∅	1.00	∅	∅	∅	**170.00**
019	∅	∅	∅	1.00	0.56	0.56	0.50	**113.00**
020	∅	∅	∅	1.00	∅	∅	0.61	**150.00**

transport

	jasper	mercury	mpc	probe	LMBFS_g	LMBFS	yahsp	**best**
001	0.79	1.00	∅	0.39	0.41	0.42	0.29	**1309.00**
002	0.98	1.00	∅	0.44	0.50	0.51	0.40	**2123.00**
003	0.84	1.00	∅	0.36	0.41	0.41	0.36	**1539.00**
004	0.84	1.00	∅	0.37	0.40	0.40	0.58	**5595.00**
005	∅	1.00	∅	0.27	0.36	0.36	0.40	**6201.00**
006	0.84	1.00	∅	0.37	0.40	0.40	0.58	**5595.00**
007	0.94	1.00	∅	0.34	0.41	0.41	0.40	**4713.00**
008	0.78	1.00	∅	∅	0.39	0.40	0.32	**4014.00**
009	0.94	1.00	∅	0.34	0.41	0.41	0.40	**4713.00**
010	0.77	1.00	∅	0.30	0.38	0.39	0.46	**4618.00**
011	0.74	1.00	∅	0.41	0.44	0.44	0.38	**1336.00**
012	0.89	1.00	∅	0.31	0.38	0.41	0.33	**1641.00**
013	0.73	1.00	∅	0.41	0.38	0.38	0.19	**1140.00**
014	∅	1.00	∅	∅	0.40	0.40	0.51	**5974.00**
015	∅	1.00	∅	0.31	0.34	0.34	0.43	**5320.00**
016	0.63	1.00	∅	0.31	0.35	0.35	0.34	**4639.00**
017	∅	1.00	∅	0.36	0.35	0.35	0.40	**4255.00**
018	0.74	1.00	∅	0.34	0.41	0.41	0.38	**4840.00**
019	0.78	1.00	∅	0.43	0.37	0.37	0.34	**3862.00**
020	∅	1.00	∅	0.37	0.43	0.43	0.42	**3831.00**

visitall

	jasper	mercury	mpc	probe	LMBFS$_g$	LMBFS	yahsp	**best**
001	0.80	1.00	∅	0.81	0.79	0.79	0.81	**908.00**
002	0.82	1.00	∅	0.84	0.79	0.79	0.81	**980.00**
003	0.78	1.00	∅	0.80	0.74	0.74	0.84	**1033.00**
004	0.80	1.00	∅	0.80	0.80	0.80	0.80	**1108.00**
005	0.74	1.00	∅	0.79	0.73	0.73	0.85	**2509.00**
006	0.81	1.00	∅	0.80	0.73	0.73	0.86	**2620.00**
007	0.80	1.00	∅	0.77	0.75	0.75	0.86	**2713.00**
008	0.78	1.00	∅	0.80	0.84	0.84	0.88	**2828.00**
009	0.77	1.00	∅	0.79	0.83	0.83	0.87	**2925.00**
010	0.74	1.00	∅	∅	0.81	0.81	0.87	**3044.00**
011	0.76	1.00	∅	∅	0.80	0.80	0.88	**3145.00**
012	0.75	1.00	∅	∅	0.75	0.75	0.85	**3268.00**
013	0.74	1.00	∅	∅	0.76	0.76	0.90	**3373.00**
014	0.78	1.00	∅	∅	0.88	0.89	0.89	**3500.00**
015	0.75	1.00	∅	∅	0.87	0.88	0.89	**3609.00**
016	0.75	1.00	∅	∅	0.72	0.73	0.87	**3740.00**
017	0.75	1.00	∅	∅	0.74	0.75	0.90	**3853.00**
018	0.74	1.00	∅	∅	0.80	0.81	0.85	**3988.00**
019	0.73	1.00	∅	∅	0.81	0.82	0.85	**4105.00**
020	0.78	1.00	∅	∅	0.65	0.66	0.85	**4244.00**

Somme métrique $m_{quality}$

	barman	childsnack	floortile	ged	hiking	parking	thoughtful	transport	visitall	**total**
mercury	14.06	0.00	1.39	19.59	16.16	18.91	0.00	20.00	20.00	**110.12**
jasper	19.97	0.00	1.39	18.05	17.43	18.03	0.00	12.24	15.37	**102.49**
LMBFS	3.75	10.07	20.00	0.51	14.41	0.00	7.34	8.00	15.65	**79.72**
probe	15.55	0.00	0.81	6.82	16.14	5.24	18.00	6.43	7.21	**76.21**
$LMBFS_g$	5.34	0.00	19.77	0.46	14.30	0.41	7.19	7.92	15.59	**70.98**
yahsp	2.20	0.00	1.80	1.25	3.53	0.00	7.59	7.91	17.19	**41.48**
mpc	4.93	7.00	8.14	2.67	4.33	7.19	3.17	0.00	0.00	**37.44**

ONERA
THE FRENCH AEROSPACE LAB
retour sur innovation

Décomposition des problèmes de planification de tâches basée sur les landmarks

Simon Vernhes

sous la direction de

Gérard Verfaillie Vincent Vidal Guillaume Infantes

Résumé

Les algorithmes permettant la création de stratégies efficaces pour la résolution d'ensemble de problèmes hétéroclites ont toujours été un des piliers de la recherche en Intelligence Artificielle. Dans cette optique, la planification de tâches a pour objectif de fournir à un système la capacité de raisonner pour interagir avec son environnement de façon autonome afin d'atteindre les buts qui lui ont été assignés. À partir d'une description de l'état initial du monde, des actions que le système peut exécuter, et des buts qu'il doit atteindre, un *planificateur* calcule une séquence d'actions dont l'exécution permet de faire passer l'état du monde dans lequel évolue le système vers un état qui satisfait les buts qu'on lui a fixés. Le problème de planification est en général difficile à résoudre (PSPACE-difficile), cependant certaines propriétés des problèmes peuvent être automatiquement extraites permettant ainsi une résolution efficace.

Dans un premier temps, nous avons développé l'algorithme LMBFS *(Landmark-based Meta Best-First Search)*. À contre-courant des planificateurs *state-of-the-art*, basés sur la recherche heuristique dans l'espace d'états, LMBFS est un algorithme qui réactualise la technique de décomposition des problèmes de planification basés sur les landmarks. Un landmark est un fluent qui doit être vrai à un certain moment durant l'exécution de n'importe quel plan solution. L'algorithme LMBFS découpe le problème principal en un ensemble de sous-problèmes et essaie de trouver une solution globale grâce aux solutions trouvées pour ces sous-problèmes. Dans un second temps, nous avons adapté un ensemble de techniques pour améliorer les performances de l'algorithme. Enfin, nous avons testé et comparé chacune de ces méthodes permettant ainsi la création d'un planificateur efficace.

Mots-Clés : planification de tâches, landmarks, algorithmes de recherche, intelligence artificielle.