Université de Montréal

Planification pour agents dans un environnement dynamique et incertain

par

Jean-François Bérubé

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Août 2003

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé:

Planification pour agents dans un environnement dynamique et incertain

présenté par:

Jean-François Bérubé

a été évalué par un jury composé des personnes suivantes:

Jean-Yves Potvin

(président-rapporteur)

Jean G. Vaucher

(directeur de recherche)

Jacques Robert

(co-directeur)

Esma Aïmeur

(membre du jury)

Mémoire accepté le:

4 novembre 2003

Résumé

Nous avons exploré des techniques de planification appliquées à l'organisation de voyages (avions et hôtels) optimaux en fonction des préférences d'un voyageur. Ces préférences portent sur trois aspects du voyage : le coût, les contraintes temporelles et le confort. Notre problème diffère des problèmes de planification classiques de plusieurs façons. D'abord, il nécessite l'utilisation d'une fonction d'utilité complexe pour représenter les préférences du voyageur et pour diriger le processus de recherche. Ensuite, il prend place dans un environnement dynamique et incertain très différent du monde statique et déterministe des planificateurs classiques. En fait, les planificateurs génériques d'aujourd'hui se sont avérés inadéquats face à ce genre de problème, spécialement lorsque des milliers d'opérateurs (avions et hôtels) doivent être pris en compte.

Nous avons développé une approche combinant le formalisme des planificateurs classiques et une recherche heuristique A^*. Nous montrons comment la fonction d'utilité peut servir à sous-estimer h, l'estimation du coût de la partie indéterminée d'un plan partiel, garantissant que la première solution trouvée soit la meilleure.

Nos tests montrent que cette approche est flexible et efficace ; flexible parce qu'elle permet facilement de trouver des itinéraires (toujours optimaux) répondant à différentes fonctions d'utilité et efficace parce qu'elle permet de répondre à des requêtes réalistes en quelques secondes. Nous croyons donc que notre stratégie pourrait servir dans d'autres applications.

Mots clés : planification, agents, incertitude, fonction d'utilité, voyage

Abstract

We explore agent planning techniques applied to the problem of making optimal travel arrangements involving flight and hotel reservations subject to the preferences of a customer. The preferences deal with 3 aspects of a trip : cost, timeliness and comfort. This problem differs from traditional planning in several ways. First, it uses a complex utility function to represent the customer preferences and to direct the planning process. Secondly, it caters to a dynamic and uncertain environment which is very different from the static and deterministic world of classical planning. Modern generic logical planners, especially when dealing with realistic data with thousands of applicable operators (flights and hotels), have been shown inadequate to solve this kind of problem.

We develop an approach combining the formalism of classical planners with the heuristic A^* search. The heuristic function is based on the utility function and we show how to apply it to underestimate h, the estimate of the cost of the plan that remains to be determined and thus garanty that the first complete solution found is the best.

Our tests show that this approach is flexible and efficient ; flexible in that it smoothly adapts to varying criteria to create quite different itineraries - each optimal - for different travellers. Efficient in that solutions for realistic requests - in an albeit synthetic world - are found in a few seconds. This suggests that the approach is valid for further application.

Key words : planning, agents, uncertainty, utility functions, travel

Table des matières

Introduction 1

1 Agents et planification : revue de l'état de l'art 3

1.1 Les agents 3

1.2 La planification logique classique 5

1.2.1 La représentation des états, des objectifs, des actions et des plans . . . 6

1.2.2 Algorithmie de la planification logique 11

1.2.3 L'algorithme de planification *POP* 13

1.2.4 Limites de la planification logique 14

1.2.5 La décomposition hiérarchique 15

1.2.6 Planification A^* 16

1.3 Planification en incertitude 19

1.3.1 Planification conditionnelle 20

1.3.2 Planification probabiliste 21

1.3.3 Planification conditionnelle probabiliste 23

1.3.4 Processus de décision markoviens 24

1.4 Planification et théorie de la décision 24

1.4.1 Théorie microéconomique et évaluation des plans 25

1.4.2 Intégration d'une évaluation continue des plans dans les algorithmes de planification 27

1.5 Résumé 27

2 Modélisation du problème de planification de voyages **29**

2.1 Mise en situation 29

2.2 Définition de l'environnement 31

2.2.1 Modèle géographique 31

2.2.2 Modèle du temps 32

2.2.3 Modélisation de la qualité 32

2.2.4 Les compagnies aériennes 33

2.2.5 Les hôtels 35

2.3 Modélisation de l'incertitude 36

2.3.1 Modèle général 36

2.3.2 Incertitude sur les prix 37

2.4 Description formelle d'un plan et de ses opérateurs 38

2.4.1 Les opérateurs 38

2.4.2 Les objectifs 39

2.4.3 Le plan 40

2.5 Modélisation de la fonction d'utilité d'un voyageur 44

2.5.1 La contrainte de budget 44

2.5.2 Les contraintes de temps 45

2.5.3 Les contraintes de qualité 46

2.5.4 L'espérance d'utilité d'un plan 47

2.5.5 La fonction de distribution de l'utilité d'un plan 47

2.5.6 Exemple d'évaluation de la distribution d'utilité d'un plan 48

2.6 Discussion sur le réalisme de l'environnement simulé et des modèles utilisés . 49

3 L'algorithme de planification **51**

3.1 Modèle général de l'algorithme 51

3.1.1 Les objectifs 52

3.1.2 La base de données 52

3.1.3 Fonctionnement général de l'algorithme 53

3.2 Évaluation des plans partiels et des opérateurs 57
3.2.1 La planification A^* 57
3.2.2 La sous-estimation du coût d'utilité espéré d'un plan partiel 57
3.2.3 Le choix d'un opérateur 60
3.3 Exemple complet de planification d'un voyage simple 61
3.4 Discussion 68

4 Analyse des résultats **70**
4.1 Les objectifs de l'évaluation 70
4.2 La base de données 71
4.3 Analyse des résultats 72
4.3.1 Sensibilité aux préférences du voyageur 73
4.3.2 Erreurs d'estimation de $h(n)$ et taille des solutions 79
4.3.3 Temps d'exécution et consommation de mémoire 81
4.3.4 Compromis entre le nombre de solutions et la qualité de ces solutions 84
4.4 Discussion 87

Conclusion **88**

Liste des figures

1.1 Exemple d'état dans le monde des blocs 6

1.2 Exemple d'opérateur *STRIPS* 7

1.3 L'opérateur *MOVE_TO_TABLE* 7

1.4 Exemples d'applications des opérateurs dans le monde des blocs 8

1.5 Exemple de plan minimal dans le monde des blocs 9

1.6 Exemple d'un plan complet et cohérent dans le monde des blocs 10

1.7 Exemple d'utilisation de A^* : graphe des distances 17

1.8 Exemple d'utilisation de A^* : étapes de la recherche 18

1.9 Exemple de plan conditionnel 21

1.10 Exemple d'opérateur probabiliste 22

2.1 Système automatisé d'aide à la planification de voyages 30

2.2 Graphe des liaisons aériennes entre les villes 34

2.3 Exemple d'un plan partiel et ses effets sur l'état du monde 41

3.1 Plan partiel à évaluer 59

3.2 Plan initial et ses effets (plan #0) 62

3.3 Plan #1 et ses effets sur l'état du monde 63

3.4 Plan #16 et ses effets sur l'état du monde 64

3.5 Plan #9 et ses effets sur l'état du monde 65

3.6 Plan #22 et ses effets sur l'état du monde 66

3.7 Plan #37 (complet et optimal) et ses effets sur l'état du monde 67

4.1 Graphe des distances entre les villes 73

4.2 Effet du facteur k_1 (importance du coût ($) du plan) sur le coût monétaire du plan 76

4.3 Effet du facteur k_2 (importance de l'écart à l'horaire) sur l'écart à l'horaire et le coût monétaire du plan 76

4.4 Effet du facteur k_3 (importance de la qualité) sur la perte de qualité moyenne et le coût monétaire du plan 77

4.5 Plan proposé lorsque le coût monétaire prime ($k_1 = 0.8$, $k_2 = 0.1$, $k_3 = 0.1$) . 78

4.6 Plan proposé lorsque le temps prime ($k_1 = 0.1$, $k_2 = 0.8$, $k_3 = 0.1$) 78

4.7 Plan proposé lorsque la qualité prime ($k_1 = 0.1$, $k_2 = 0.1$, $k_3 = 0.8$) 79

4.8 Erreur d'estimation relative moyenne en fonction de la taille de la solution . . 81

4.9 Erreur d'estimation absolue moyenne en fonction de la taille de la solution . . . 82

4.10 Temps d'exécution en fonction de la taille de la solution 83

4.11 Nombre de plans partiels explorés en fonction de la taille de la solution 84

4.12 Temps de planification en fonction du nombre de plans partiels explorés 85

4.13 Coût d'utilité des plans de différents rangs 86

4.14 Temps d'exécution en fonction du nombre de plans trouvés 86

4.15 Nombre de plans partiels explorés en fonction du nombre de plans trouvés . . . 87

Liste des tableaux

2.1 Caractéristiques des compagnies aériennes 33

2.2 Caractéristiques des types d'hôtels 35

2.3 Pré-conditions et effets des opérateurs 39

2.4 Description du voyage à planifier 40

2.5 Exemple de description de voyage avec contraintes temporelles 48

2.6 Plan complet pour le problème du tableau 2.5 48

3.1 Contraintes temporelles du voyageur correspondant au plan de la figure 3.1 . . 59

3.2 Description du voyage à planifier 61

4.1 Description des itinéraires 74

4.2 Influence des préférences sur le coût, la qualité et le respect des contraintes de temps des solutions proposées 75

Remerciements

Bien que cette recherche se veuille un travail individuel, elle n'aurait pas été la même sans la contribution de plusieurs personnes. J'aimerais remercier chacune d'elles en soulignant à quel point leur aide a été appréciée.

Je remercie Monsieur Jean Vaucher pour l'intérêt qu'il a accordé à mon travail. Au-delà d'un précieux soutien scientifique, il m'aura aidé à développer mes aptitudes et surtout mon intérêt pour la recherche. Je remercie Monsieur Jacques Robert d'avoir, pour une seconde fois, accepté de me superviser pour un travail de maîtrise. C'est en partie à lui que ce mémoire doit la teinte économique qui fait son originalité. Je remercie Monsieur Robert Gérin-Lajoie pour son soutien indéfectible et ses bonnes idées. Grâce à lui, ce mémoire propose une solution à un problème appliqué ayant de réels enjeux économiques.

J'aimerais aussi remercier le CIRANO et tout ceux et celles qui font qu'une telle institution existe. Son appui scientifique, logistique et financier est une véritable bénédiction pour un étudiant.

Merci finalement à Monsieur Jérôme Blanc et Madame Isabelle Therrien pour leur aide technique, à Messieurs Houssein Ben-Ameur et Sylvain Riopel pour ces petites discussions qui forcent à vulgariser nos idées et à Madame Lise Raymond pour la révision grammaticale et orthographique.

Introduction

L'intelligence artificielle incarne le rêve d'arriver un jour à concevoir des programmes capables de reproduire les capacités intellectuelles humaines. Avec un objectif aussi ambitieux, il n'est pas étonnant qu'un grand nombre des applications d'avant-garde de l'informatique en soit issu. Concevoir un programme capable d'entretenir une conversation soutenue avec un humain, réussir à traduire correctement des textes, ou encore, voir un grand maître incliner son roi face à une machine. C'est dans l'espoir de relever de tels défis que les pionniers de l'intelligence artificielle ont cherché à reproduire la puissance cognitive du cerveau humain. On a d'abord voulu donner à la machine la capacité de raisonner logiquement, comme un humain peut le faire. Mais sommes-nous vraiment des êtres logiques ? Ce débat philosophique, de même que la lenteur excessive des interpréteurs logiques face à des problèmes complexes, font que la tendance moderne en intelligence artificielle tourne autour de techniques dites "agents", délaissant quelque peu la programmation logique.

Avec les agents, on cherche à concevoir des programmes autonomes capables d'initiatives, d'apprendre, et de plus en plus, capables d'interagir avec d'autres agents. On se dirige en quelque sorte vers des programmes grégaires formant des communautés à l'intérieur desquelles se crée une collaboration autour d'un objectif commun. Les agents d'aujourd'hui sont utilisés principalement dans des applications en commerce électronique pour faire des achats sur Internet ou pour participer à des enchères, et en robotique pour diriger divers modules mécaniques.

Le désir de développer des agents de plus en plus autonomes fait de la planification un axe de recherche important en intelligence artificielle. L'autonomie nécessite une capacité à raisonner sur les meilleures actions à entreprendre afin d'atteindre un objectif précis. Dans le

cas d'un agent travaillant seul, ce processus de décision se résume souvent à l'établissement d'un plan définissant la séquence d'actions qui lui permettront de passer d'un état du monde quelconque vers l'état qu'il a pour objectif de voir se réaliser.

Avec l'équipe de recherche en commerce électronique du CIRANO*, nous utilisons les techniques agents dans le développement d'un système d'aide à l'achat de voyages sur Internet. Nous souhaitons offrir aux consommateurs un système capable de trouver et d'analyser une très grande quantité d'informations afin de leur proposer des plans de voyage correspondant à leurs préférences. La demande pour un tel système se fait sentir depuis que l'industrie du voyage, répondant à l'émergence des technologies de l'information par une modification de ses modèles d'affaire, a fait du Web une place de marché où il est possible d'acheter toutes sortes de produits de voyage. Les grandes firmes américaines spécialisées dans le commerce électronique telles que *Forrester Research, Jupiter Research* et *Phocys Wright* évaluent le chiffre d'affaire de l'industrie du voyage en ligne entre 27 et 56 milliards de dollars sur un marché de 215 milliards†. Bien que les leaders de la vente de voyage sur le Web (*Expedia, Travelocity, Orbitz, Hotel.com*) offrent des forfaits sur leurs sites, il est plus rare que des consommateurs complètent l'ensemble de leurs transactions sur Internet. Leurs capacités à analyser et à traiter la grande quantité d'information disponible étant limitée, ils préfèrent confier l'organisation de leurs voyages à des agences. Avec son système d'aide à l'achat électronique de voyage, le CIRANO espère trouver une solution intéressante à ce problème.

Notre mémoire de maîtrise traite du problème de planification inhérant à ce système. Nous avons développé un modèle d'incertitude pour le domaine du voyage, une modélisation des préférences des voyageurs et une stratégie de planification permettant de proposer des plans (avions, hôtels) complets et optimaux adaptés à ces préférences. Ce mémoire est divisé en quatre chapitres. Nous débuterons avec une revue des principales techniques de planification. Nous présenterons ensuite notre modélisation du problème, puis nous exposerons l'algorithme de recherche utilisé par notre planificateur et nous terminerons avec l'analyse des résultats d'un certain nombre de tests permettant d'évaluer notre stratégie.

---

*Centre Inter-universitaire de Recherche en ANalyse des Organisations

†*La Presse*, 13 août 2003

Chapitre 1

Agents et planification : revue de l'état de l'art

La recherche sur les agents fait appel à plusieurs spécialités en informatique telles que l'intelligence artificielle, la recherche opérationnelle, les systèmes distribués, la représentation des connaissances, la robotique, etc. Notre revue de l'état de l'art ne couvrira pas l'ensemble de la problématique des agents, mais seulement la recherche en planification. Ce chapitre débutera avec une description de ce qu'est un *agent*, mais nous entrerons rapidement dans le vif du sujet avec un survol des principales techniques de planification : logique, conditionnelle, probabiliste et non-catégorielle.

1.1 Les agents

Le terme *agent* est apparu dans la littérature informatique avec l'arrivée de programmes suffisamment sophistiqués en terme d'autonomie et d'intelligence pour que l'on puisse les considérer comme des entités différentes du logiciel traditionnel. Se rapprochant plus d'un robot virtuel que d'un programme classique, les agents suscitent depuis quelques années un grand engouement chez les chercheurs. Malgré la popularité grandissante de leur nom, il semble que tous ne s'entendent toujours pas sur une définition précise de ce qu'est un agent. Nous utilise-

rons une adaptation de celle proposée par Wooldridge [Wei99] : Un *agent* est un programme informatique *évoluant* dans un certain *environnement* et ayant la capacité d'agir de manière *autonome* afin d'atteindre des *objectifs* précis. Cette définition ne mentionne pas la notion d'intelligence, souvent associée au terme agent. Wooldridge et Jennings [WJ95] soutiennent que pour être qualifié d'intelligent, un agent doit être capable de *flexibilité* dans ses actions. Cette flexibilité implique que l'agent soit *réactif*, *pro-actif* et capable d'*interactions sociales*.

On dit d'un agent qu'il est *réactif* lorsqu'il est en mesure de réagir à des changements dans son environnement. Cette réaction peut être aussi simple qu'une réponse directe à un stimulus. La *pro-activité* de l'agent est nettement plus complexe puisqu'elle sous-entend que l'agent soit capable d'initiative. Il doit pouvoir élaborer et mettre en oeuvre des plans lui permettant d'atteindre ses objectifs. La pro-activité constitue la caractéristique centrale d'un agent intelligent puisqu'elle est à la base de la notion d'autonomie à laquelle on associe généralement les agents. Finalement, un agent flexible doit pouvoir interagir avec les humains et/ou d'autres agents, d'où l'importance qu'il soit capable d'interactions sociales.

Dans le livre sur les systèmes multi-agents édité par Weiss, Wooldridge [Wei99] présente quatre classes d'architecture pouvant servir de structure de base à un agent. L'agent le plus simple prend la forme d'un agent *réactif* pour lequel chaque décision est préprogrammée en fonction des différentes situations auxquelles il peut faire face. Vient ensuite l'agent issu d'une architecture *par couches (layered architecture)* prenant ses décisions via diverses couches logicielles et analysant l'environnement avec différents niveaux d'abstraction. L'agent *logique* dont les décisions sont issues d'un mécanisme d'inférence, est nettement plus complexe puisqu'il est véritablement capable de *raisonner*. Finalement, l'agent *belief-desire-intention*, qui manipule durant son processus décisionnel des structures de données représentant ses croyances, désirs et intentions, correspond à l'architecture agent la plus évoluée à ce jour. Dans tous les cas, l'agent doit disposer de connaissances sur son environnement, des capacités à le modifier et d'objectifs guidant son comportement.

La principale motivation du développement des techniques agents vient de la nécessité d'améliorer la robustesse et l'efficacité des programmes en situation d'incertitude. L'approche multi-agents permet de construire des systèmes composés d'un ensemble d'entités autonomes

(les agents) collaborant à un objectif commun. Cette façon de faire permet de distribuer la charge de calcul lorsque les agents se trouvent sur des machines différentes et d'obtenir un programme robuste, dans la mesure où les agents s'adaptent bien au dynamisme de leur environnement et que des stratégies de collaboration efficaces leur permettent de faire un bon travail de groupe.

Pris individuellement, les agents doivent exercer un processus cognitif leur permettant d'analyser l'environnement et de déterminer les meilleures actions à entreprendre. C'est ce processus par lequel l'agent développe sa stratégie d'action que nous appelons la *planification.* L'élaboration d'un plan ou d'une solution probable dans un univers virtuel constitue la base d'un raisonnement autonome. Par le fait même, la planification s'affiche comme un aspect fondamental de *l'intelligence* d'un agent.

1.2 La planification logique classique

Nous présenterons plusieurs approches possibles à la planification, en commençant par l'approche logique qui fut la première à être développée. Bien qu'elle soit limitée par de nombreuses et fortes hypothèses, la planification logique demeure la base de la plupart des autres techniques. Dans cette section, nous nous appliquerons à bien expliquer les tenants et aboutissants de la planification logique afin d'asseoir solidement les bases sur lesquelles repose ce mémoire. Notez que notre présentation de la planification logique est largement basée sur les chapitres 11 à 13 du livre de Russell et Norvig sur l'intelligence artificielle [RN95].

La planification est l'opération par laquelle un agent détermine la séquence d'actions *(le plan)* lui permettant d'atteindre un objectif précis en transformant son environnement. La représentation des états du monde, des actions et des plans est par conséquent un pré-requis à toute forme planification.

1.2.1 La représentation des états, des objectifs, des actions et des plans

Les états et les objectifs

Les états du monde sont représentés par un ensemble de propositions. La figure 1.1 donne un exemple d'état dans le monde des blocs. Cet environnement école représente un monde composé d'une table sur laquelle sont disposés des blocs qu'un robot est capable de déplacer un à la fois. Les états du monde des blocs sont définis à partir de deux types de propositions : *on(X,Y)* qui signifie que le bloc X est sur le bloc Y et *clear(X)* qui signifie qu'il n'y a rien sur le bloc X.

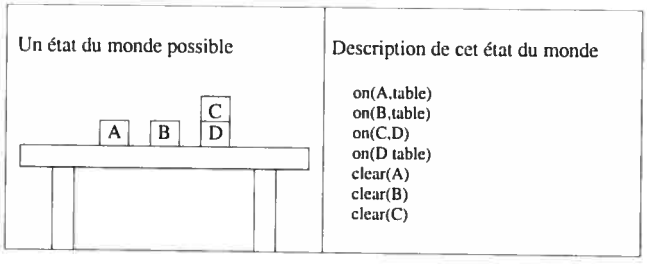

FIG. 1.1 – Exemple d'état dans le monde des blocs

Un objectif est simplement un état du monde que l'agent aimerait voir se réaliser. Par exemple, la figure 1.1 pourrait très bien représenter l'objectif d'un problème de planification.

Les actions

La norme en planification logique veut que les actions y soient représentées avec le langage *STRIPS** [FN71]. Ce langage datant du début des années 1970, définit les actions sous forme d'opérateurs, chacun étant composé des pré-conditions nécessaires à l'exécution de l'action et des effets engendrés par sa mise en application. La section de gauche de la figure 1.2 illustre un opérateur *STRIPS* générique composé de son action, de ses pré-conditions et de ses effets. À droite, un opérateur provenant du monde des blocs définit l'action *MOVE(X,Y,Z)* qui permet de déplacer le bloc X de Y vers Z. Pour que cette action puisse être appliquée, les blocs X et

*STanford Research Institute Problem Solver

Z doivent être libres et le bloc X doit être placé sur le bloc Y. Une fois exécutée, l'action aura pour effet de libérer le bloc Y, de placer le bloc X sur Z et de faire en sorte que Z ne soit plus libre.

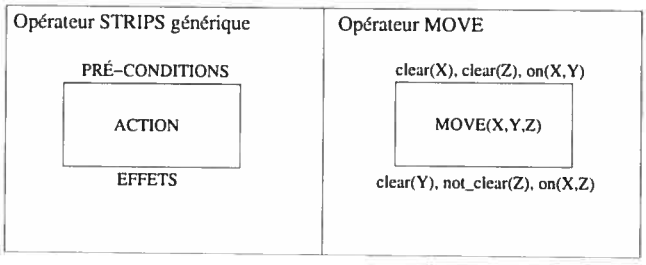

FIG. 1.2 – Exemple d'opérateur *STRIPS*

L'opérateur *MOVE* ne suffit pas pour représenter toutes les actions possibles dans le monde des blocs. Il ne définit pas correctement les opérations de déplacement lorsque la table entre en ligne de compte. Pour déplacer un bloc vers la table, il faudrait que $Z = table$. Or ceci nécessiterait d'avoir $clear(table)$ avant l'application de l'opérateur, chose qui ne devrait jamais arriver. Afin de résoudre ce problème, nous devons ajouter l'opérateur $MOVE\_TO\_TABLE$ qui permet de placer un bloc sur la table. La figure 1.3 illustre les pré-conditions et les effets de cet opérateur.

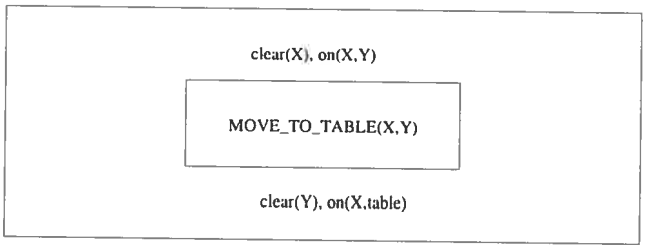

FIG. 1.3 – L'opérateur *MOVE_TO_TABLE*

Un autre problème survient si nous voulons déplacer un bloc provenant de la table. Dans ce cas, l'opérateur *MOVE* verrait sa variable $Y = table$ et aurait par conséquent $clear(table)$ comme effet. Nous pouvons résoudre ce problème sans avoir à créer un nouvel opérateur. Il suffit simplement de redéfinir le prédicat $clear(X)$ comme suit : "il est possible de placer un bloc sur X". Comme il est toujours possible de placer un bloc sur la table, cette pré-condition sera toujours vérifiée.

La figure 1.4 donne deux exemples d'application des opérateurs du monde des blocs. Dans la partie du haut, l'opérateur *MOVE* est utilisé pour déplacer le bloc A du bloc B vers le bloc C. Dans la partie du bas, l'opérateur *MOVE_TO_TABLE* est utilisé pour déplacer le bloc C de D vers la table. Dans les deux cas, remarquez que toutes les pré-conditions des opérateurs sont vérifiées dans l'état du monde dans lequel l'action est appliquée. Remarquez aussi que l'état résultant de l'application des opérateurs correspond parfaitement à leurs effets.

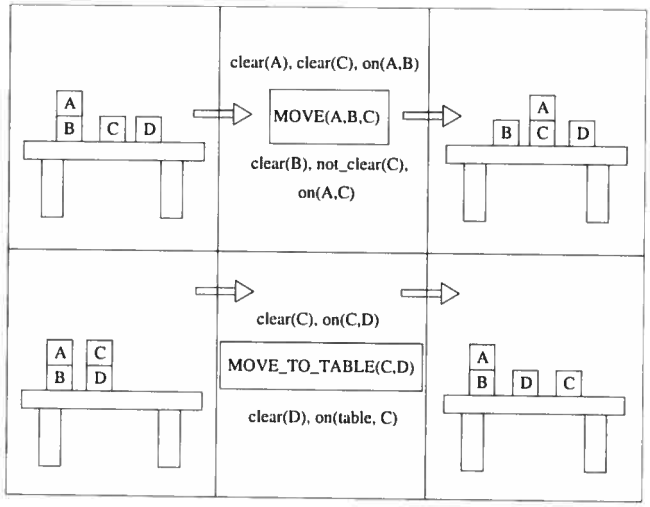

FIG. 1.4 – Exemples d'applications des opérateurs dans le monde des blocs

Bien qu'il date de plus de 30 ans, le langage *STRIPS* demeure *le* standard pour représenter les actions et, indirectement, les plans. Sa simplicité et sa souplesse semblent lui avoir permis de s'imposer. À preuve, tous les planificateurs des compétitions de planification de l'*International Artificial Intelligence Planning Systems Conference* [La00] utilisaient *STRIPS*. Plutôt que d'être mis de côté suite à l'émergence d'une nouvelle forme de représentation, on s'en inspire pour développer des extensions permettant d'exprimer plus de complexité. C'est par exemple le cas de *ADL* et *PDDL* [McD00] qui permettent d'exprimer des contraintes temporelles ainsi que la consommation des ressources utilisées par les opérateurs.

Le plan

Un plan est une abstraction d'un ensemble d'opérateurs qui, appliqués en ordre, permettent de passer d'un état du monde initial à un état du monde souhaité. Toute planification débute avec un *plan minimal* ne contenant que les opérateurs *start* et *finish*. L'opérateur *start*, dont les effets correspondent aux propositions définissant l'état initial, donne un point de départ au planificateur, tandis que l'opérateur *finish*, dont les pré-conditions correspondent aux propositions définissant l'état à atteindre, permet de guider le planificateur vers son objectif. La figure 1.5 donne un exemple de plan minimal dans le monde des blocs. Ce plan est évidemment incomplet puisque les effets de *start* ne correspondent pas aux pré-conditions de *finish*. Par l'ajout d'opérateurs, le planificateur doit combler ce vide et ainsi produire un plan complet.

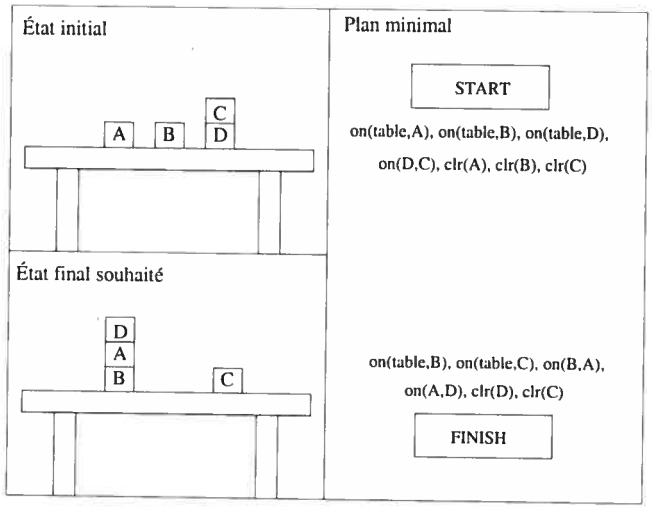

FIG. 1.5 – Exemple de plan minimal dans le monde des blocs

La figure 1.6 illustre un plan complet pour le problème de la figure 1.5. Dans ce schéma, les flèches indiquent les liens causaux entre les effets et les pré-conditions des opérateurs. Ce plan compte cinq opérateurs sujets à des *contraintes d'ordonnancement : start* doit être le premier opérateur à être appliqué et *finish* doit être appliqué à la fin. *MOVE(D,...)* doit être exécuté après *MOVE_TO_TABLE(C,D)* car un effet (*clr(D)*) de ce dernier est nécessaire à l'application de *MOVE(D,...)*. *MOVE(D,...)* doit aussi venir après *MOVE(A,...)* car un de ses effets (*not_clr(A)*) est en conflit avec une pré-condition de *MOVE(A,...)*. Formellement, les contraintes d'ordonnancement sont définies comme une relation entre deux étapes du plan $(S_i \prec S_j)$ spécifiant

que S_i doit être appliquée avant S_j, mais pas nécessairement immédiatement avant.

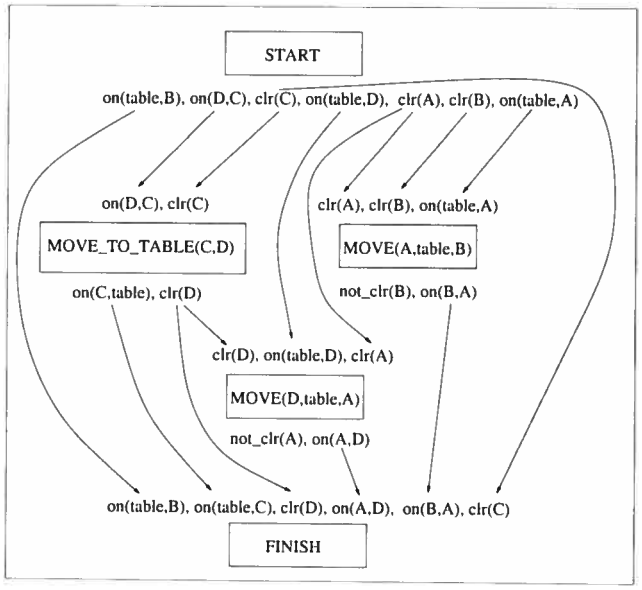

FIG. 1.6 – Exemple d'un plan complet et cohérent dans le monde des blocs

Ce plan est en fait un plan *partiellement ordonné* car l'ensemble des ordonnancements du plan ne couvre pas toutes ses étapes. Ainsi, *MOVE_TO_TABLE(C,D)* et *MOVE(A,table,B)* peuvent être appliqués dans n'importe quel ordre. L'agent choisira arbitrairement l'ordre d'application en *linéarisant* le plan au moment de l'exécution. La production de plans partiellement ordonnés s'inscrit dans une logique de *moindre engagement* (*least commitment*) qui évite au planificateur de prendre des décisions contraignantes lorsque ce n'est pas absolument nécessaire. Cette façon de faire accroît l'efficacité du planificateur en lui évitant de prendre une mauvaise décision d'ordonnancement qui l'obligerait à refaire une partie de la planification pour corriger son erreur.

Le plan de la figure 1.6 comporte aussi plusieurs *liens causaux*. En fait, chaque flèche représente un lien causal signifiant que l'effet d'un opérateur permet de satisfaire la pré-condition d'un autre. Par exemple, la proposition *on(B,A)*, effet de l'opérateur *MOVE(A,table,B)*, est liée à une pré-condition de l'opérateur *finish*. Formellement, un lien causal a la forme $S_i \xrightarrow{c} S_j$, c'est-à-dire que S_i produit l'effet c nécessaire à l'application de S_j.

Remarquez que toutes les pré-conditions de tous les opérateurs du plan sont vérifiées, fai-

sant de lui un plan *complet*. Remarquez aussi qu'il n'y a aucune contradiction dans l'ordonnancement des opérateurs ; le plan est donc *cohérent*. Ces deux caractéristiques sont les conditions nécessaires et suffisantes pour qu'un plan puisse être considéré comme une solution valable à un problème de planification.

1.2.2 Algorithmie de la planification logique

Le processus de planification peut se résumer à la *recherche* d'une séquence d'opérateurs permettant de passer de l'état initial à l'état final souhaité. Les choix qui sont faits lors du développement de l'algorithme de recherche influenceront à la fois la qualité des plans produits et le temps nécessaire à leur production. Planifier, c'est avant tout, faire l'exploration d'un espace afin d'y trouver un plan. Le temps qu'un algorithme passe à explorer une bonne solution constitue toujours un bon investissement puisqu'il permettra d'obtenir un bon résultat. Par contre, chaque seconde passée à explorer une mauvaise solution est une seconde de perdue. Éviter d'explorer de mauvaises solutions permet donc de réduire le temps de calcul. Autrement dit, le temps nécessaire à la recherche d'un plan est proportionnel au temps perdu à explorer de mauvaises solutions.

La taille de l'espace de recherche et la manière dont l'exploration est faite sont donc cruciales. Pour mesurer l'ordre de grandeur d'un espace de recherche, il faut connaître son *facteur de branchement* (b), c'est-à-dire le nombre d'opérateurs applicables pour chacune des étapes d'un plan. L'ordre de grandeur de la taille d'un espace de recherche pour un plan de n étapes sera $O(b^n)$.

La stratégie de recherche la plus simple serait une recherche *avant* dans *l'espace des états*. Ce type de recherche part de l'état initial et considère tous les états pouvant être atteints à partir de là. La recherche continue ainsi jusqu'à l'état final. Dans l'exemple de la figure 1.5, 7 états différents peuvent être atteints à partir de l'état initial (2 pour A et B et 3 pour C). Si nous considérons que ce nombre est représentatif du facteur de branchement moyen et que l'état final peut être atteint en 3 étapes, alors le planificateur devra, en pire cas (exploration de tout l'espace), explorer 7^3 (donc 343) solutions avant d'arriver à l'état final. Ce chiffre peut sembler petit, mais remarquez que le plan compte très peu d'étapes. Avec un plan de seulement 5

étapes, l'espace de recherche passerait à plus de 16 000 solutions. De plus, dans cet exemple le planificateur n'a accès qu'à des opérateurs pertinents pour le problème à résoudre. Mais dans certaines situations, il pourrait exister d'autres opérateurs comme, par exemple, peinturer un bloc, couper un bloc en deux, etc. Ce genre d'opérateurs impertinents peuvent faire augmenter considérablement le facteur de branchement. Dans notre exemple, si b valait 12 plutôt que 7, le nombre de plans à explorer en pire cas serait de 1 728 lorsque n vaut 3. C'est 5 fois plus qu'avec un facteur de branchement de 7.

Afin de réduire la taille de l'espace de recherche, nous pouvons faire une recherche *arrière* dans l'espace des états. Cette façon de faire a l'avantage de réduire le facteur de branchement en éliminant tous les opérateurs qui n'ont rien à voir avec l'objectif à atteindre. Dans notre exemple, une recherche dirigée par l'état final permettrait d'éliminer d'emblée des états où un bloc a été peinturé par exemple. Toutefois, il se peut qu'une recherche arrière ne permette pas de réduire le facteur de branchement. C'est le cas dans le monde des blocs où les seuls opérateurs disponibles sont pertinents au problème. Ainsi, une recherche arrière dans l'espace des états pour le problème de la figure 1.5 ne présenterait aucun avantage par rapport à une recherche avant.

La solution adoptée par les planificateurs modernes consiste à faire une recherche arrière dans *l'espace des plans* plutôt que dans l'espace des états. La recherche est d'abord faite parmi les opérateurs permettant de satisfaire une pré-condition de l'état final. L'ajout d'un opérateur dans le plan crée un plan partiel qui limite la recherche aux pré-conditions de ses opérateurs. En guise de comparaison, la taille de l'espace de recherche du problème de la figure 1.5 serait de 64 états (4^3, le facteur de branchement moyen étant de 4) avec une recherche arrière dans l'espace des états et de 8 (2^3, le facteur de branchement étant égal au nombre d'opérateurs possibles) avec une recherche arrière dans l'espace des plans. Malgré le petit ordre de grandeur de notre exemple, la différence entre les deux approches est très significative.

Peut-être vous demandez-vous comment le facteur de branchement peut être seulement de 2 pour une recherche dans l'espace des plans du monde des blocs ? Il faut voir que les opérateurs ne sont en fait que des coquilles pouvant être *partiellement instanciées*. Par exemple, l'opérateur *MOVE(X,Y,Z)* est un opérateur pouvant être appliqué pour n'importe quelle valeurs

(blocs ou table) de X, Y et Z. Éventuellement, il faudra que toutes les variables des opérateurs du plan soient liées à une constante (condition nécessaire à la cohérence du plan), mais au moment du raffinement du plan, rien n'empêche les opérateurs d'être partiellement instanciés. Voilà pourquoi, dans le monde des blocs, le facteur de branchement d'une recherche dans l'espace des plans est limité à 2.

1.2.3 L'algorithme de planification *POP*

L'algorithme *POP*[†] [Wel94], basé sur l'algorithme *SNLP*[‡] [MR91], est un algorithme produisant des plans partiellement ordonnés, cohérents et complets. *POP* est un exemple d'algorithme faisant une recherche arrière dans l'espace des plans. Le fonctionnement général de *POP* suit les étapes suivantes :

1. On donne à l'algorithme l'état initial, l'objectif et l'ensemble des opérateurs dont il dispose.
2. Un plan minimal ne contenant que l'état final et l'état initial est construit *(start et finish)*.
3. Tant que le plan n'est pas une solution au problème de planification, l'algorithme choisit une condition à satisfaire parmi les pré-conditions des opérateurs du plan qui ne le sont pas. Si cette condition peut être remplie par l'établissement de liens causaux avec des effets d'opérateurs déjà présents dans le plan, alors aucun nouvel opérateur ne sera ajouté. Autrement, un nouvel opérateur dont l'un des effets permet de satisfaire la condition choisie sera ajouté au plan. Cet ajout sera suivi d'un mécanisme de vérification et de résolution des conflits.

Le mécanisme *résolution des conflits* est très important en planification logique. Rappelons qu'un plan contient toujours un ensemble de liens causaux liant l'effet d'un opérateur avec une pré-condition d'un autre. Ces liens sont essentiels à la cohérence du plan et doivent donc être protégés contre des opérateurs qui pourraient les briser. La manière usuelle de protéger les liens causaux contre des conflits emploie un système de *promotions* et de *démotions*. Une

[†] **P**artial **O**rder **P**lanning

[‡] **S**ystematic **N**on**L**inear **P**lanning

promotion consiste à forcer l'opérateur créant le conflit à être exécuté avant un autre et une démotion consiste à forcer cet opérateur à être exécuté plus tard. La résolution des conflits par promotions/démotions crée donc des modifications à l'ensemble des ordonnancements du plan.

1.2.4 Limites de la planification logique

Bien qu'ils aient permis d'établir les bases de la planification en intelligence artificielle, les planificateurs logiques ne sont pas à la hauteur des besoins actuels. Malgré l'utilisation d'une recherche arrière dans l'espace des plans, la complexité algorithmique des planificateurs logiques demeure trop élevée pour qu'ils soient appliqués à des problèmes un peu plus complexes que ceux du monde des blocs. Dans leur revue de l'état de l'art sur les efforts en planification depuis une quinzaine d'années, Zimmerman et Kambhampati [ZK03] soulignent que les problèmes dans le monde des blocs dont le nombre de blocs dépassait 10, de même que plusieurs problèmes de logistique s'avéraient au-delà des capacités de la plupart des planificateurs du début des années 90. La faiblesse des heuristiques servant à diriger la recherche semble avoir été la cause de ces piètres performances. S'appuyant sur la démonstration de Bylander [Byl92] selon laquelle même des problèmes de planification très simples ne sont pas calculables en temps polynomial, Zimmerman et Kambhampati [ZK03] insistent sur la nécessité d'aider les planificateurs à réduire considérablement leurs espaces de recherche. Cette nécessité de bâtir des heuristiques permettant de réduire les espaces de recherche modifient présentement la tendance voulant qu'un planificateur générique soit préférable à une solution limitée à un type particulier d'application.

Les planificateurs logiques continuent tout de même d'être utilisés. Par exemple, tous les planificateurs des compétitions de planification de l'*International Artificial Intelligence Planning Systems Conference* [La00] devaient être des planificateurs génériques capables de résoudre des problèmes très différents. Même l'usage de décomposition hiérarchique, dont nous discuterons plus tard, était vu comme une forme de tricherie en raison de son fort aspect heuristique. Plusieurs des planificateurs de cette compétition étaient des extensions de *Graphplan* [BF97], un planificateur générique datant de la fin des années 1990. Nous avons essayé *Graphplan* sur notre problème de planification de voyages mais il n'a pas été à la hauteur, même

pour des voyages très simples, en raison de l'immense facteur de branchement du problème.

Soulignons finalement que les planificateurs logiques sont difficilement applicables à des problématiques réelles car ils ne permettent pas de gérer l'incertitude causée par le dynamisme de plusieurs environnements réalistes. De plus, ils limitent l'évaluation des plans à une fonction binaire basée sur le succès ou l'échec face à l'atteinte de l'objectif. Or, comme nous le verrons plus loin, certaines situations nécessiteraient une évaluation plus nuancée des plans.

1.2.5 La décomposition hiérarchique

Nous avons vu qu'un algorithme de recherche arrière dans l'espace des plans (comme *POP*) est préférable à une recherche dans l'espace des états. Mais un tel algorithme demeure toutefois limité dans la taille des problèmes qu'il peut résoudre. La recherche effectuée par *POP* n'étant absolument pas dirigée, l'algorithme ne tiendra pas la route sur des problèmes plus réalistes que les traditionnels exemples confinés au monde des blocs. Pour des problèmes complexes, la banque d'opérateurs est immense, rendant impraticable une recherche efficace dans l'espace des plans. Russell et Norvig [RN95] donnent l'exemple de la construction d'une maison comme problème trop complexe pour qu'une planification logique classique puisse le résoudre.

Introduite en 1974 par Sacerdoti [Sac74], la décomposition hiérarchique permet de limiter le domaine de recherche du planificateur. Dans leur ouvrage sur l'intelligence artificielle Russell et Norvig [RN95] présentent le concept de décomposition hiérarchique en définissant deux types d'opérateurs : les *opérateurs abstraits* et les *opérateurs primitifs*. Les opérateurs primitifs sont équivalents aux opérateurs standards dont nous avons parlé précédemment. Les opérateurs abstraits sont des opérateurs de plus haut niveau qui sont eux-mêmes composés d'autres opérateurs abstraits ou d'opérateurs primitifs. Dans l'exemple de la construction d'une maison, l'action "planter un clou" pourrait être un opérateur primitif alors que l'action "construire la charpente" serait un opérateur abstrait.

Le gain d'efficacité que permet une décomposition hiérarchique est considérable. Supposons un problème de planification pouvant être résolu en n étapes et supposons aussi que le

temps consacré à la résolution des conflits et au traitement des contraintes soit négligeable. Soient b le facteur de branchement (nombre de méthodes de décomposition par étape), s le le nombre d'étapes par méthode de décomposition et d la profondeur du plan hiérarchique. Nous en discutions plus tôt, la complexité associée au choix des opérateurs dans un plan non-hiérarchique est d'ordre $O(b^n)$ en pire cas. Un planificateur hiérarchique pourra travailler sur des plans abstraits où exactement une de chaque b décomposition est une solution. Le planificateur devra donc analyser sb étapes pour la profondeur $d = 1$, il devra analyser sb étapes à la profondeur $d = 2$ mais en décomposer seulement $\frac{1}{b}$ pour un total de bs^2. Ainsi, le nombre de plans à analyser sera de $\sum_{i-1}^{d} bs^i = O(bs^d)$. Si $b = 3$, $s = 4$ et $d = 3$, alors $n = s^d = 64$. Une planification non-hiérarchique devra donc considérer $3 \cdot 10^{30}$ plans alors qu'une planification hiérarchique n'en considérera que 576.

1.2.6 Planification A^*

La décomposition hiérarchique n'est pas applicable à tous les types de problèmes. Imaginez un monde des blocs dans lequel les opérateurs ne peuvent être partiellement instanciés parce que seulement quelques déplacements comme *MOVE(A,B,C)* sont possibles. Si l'environnement compte plusieurs milliers de blocs et quelques dizaines de milliers d'opérateurs complètement instanciés, l'espace de recherche sera démesuré. Compte tenu du fait qu'il n'y a en fait que deux types d'opérateurs très simples, une décomposition hiérarchique du problème n'est pas possible. Comme aucune solution apparente ne permet de réduire l'espace de recherche, il faudra concentrer nos efforts pour trouver une stratégie permettant de limiter l'exploration de l'espace.

Il existe plusieurs techniques permettant de diriger une recherche : recherche en largeur, recherche en profondeur, *hill climbing*, algorithmes gloutons, etc. Nous avons choisi de présenter l'algorithme A^*, d'abord parce qu'il trouve toujours la meilleure solution lorsqu'une solution existe. Ensuite, parce qu'il est optimal en ce sens qu'aucun autre algorithme ne peut trouver la meilleure solution en explorant moins de noeuds et, finalement, parce que le prix à payer pour cette optimalité est assez faible pour les problèmes ayant une solution de petite taille, la complexité en pire cas de A^* étant exponentielle dans la taille de la solution.

A^* a été proposé pour la première fois par Hart, Nilsson et Raphael en 1968 [HNR68], puis corrigé lors d'une autre publication de Hart et al. en 1972 [HNR72]. Le principe de A^* est de diriger la recherche grâce à une sous-estimation du coût du chemin à parcourir. Soit $f(n)$ le coût total d'un plan passant par l'état n[§]. Avant le début de la planification, ce coût ne peut être qu'une estimation du coût réel, appelons cet estimé $h(n)$ et sa vraie valeur $h^*(n)$. La recherche débute avec $f(n) = h(n)$, puis, lorsqu'il y a au moins un opérateur dans le plan, le coût estimé du plan devient $f(n) = g(n) + h(n)$. Ici, $g(n)$ représente le coût des opérateurs ajoutés jusqu'à maintenant alors que $h(n)$ représente une sous-estimation du coût des opérateurs qui devront être ajoutés étant donnés ceux déjà présents dans le plan. Tous les plans partiels produits sont conservés en mémoire et le raffinement du plan se fait toujours en commençant par le plan partiel comportant le coût estimé le plus faible jusqu'à l'obtention d'un plan dont le coût sera nécessairement minimal. La découverte du meilleur plan est conditionnelle au fait que les estimés soient toujours inférieurs ou égaux au coût réel du meilleur plan possible.

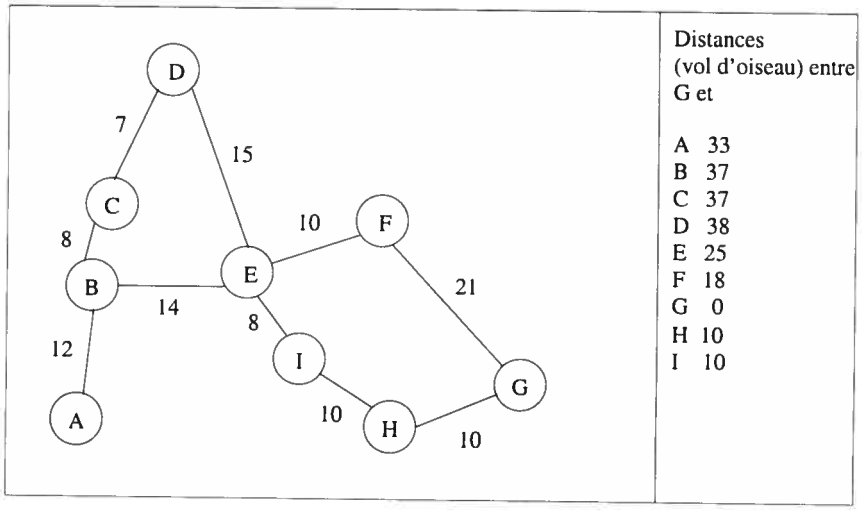

FIG. 1.7 – Exemple d'utilisation de A^* : graphe des distances

La nécessité de diriger la recherche par des sous-estimations de la fonction objective est intuitivement assez facile à comprendre. Si un plan avait un coût estimé supérieur à son coût réel, il serait possible que ce plan soit faussement considéré comme moins bon qu'un autre et que le raffinement de ce plan en soit compromis. En mettant ainsi de côté un plan qui aurait pu

[§]On associe souvent les états des problèmes de recherche à des noeuds, d'où la lettre n.

être meilleur qu'un autre, nous ne trouverions possiblement pas la meilleure solution.

On illustre habituellement A^* avec un problème de plus court chemin. Nous vous présentons un exemple adapté du livre de Russell et Norvig [RN95]. La figure 1.7 donne les distances routières entre des villes (étiquette des arcs) et les distances à vol d'oiseau entre G et toutes les autres villes. Vous aurez compris que nous utiliserons la distance à vol d'oiseau comme sous-estimation de la distance réelle entre les villes. Supposons que nous sommes en B et que nous souhaitions nous rendre en G. La figure 1.8 illustre l'exploration qui serait faire avec l'algorithme A^*. Remarquez que la décision de la prochaine ville à laquelle se rendre se fait toujours en comparant la somme de $g(n)$ et $h(n)$ à toutes les alternatives possibles. Comme la valeur choisie pour $h(n)$ (la distance à vol d'oiseau) ne surestime jamais la vraie valeur $h^*(n)$, le chemin le plus court sera trouvé : $B - E - I - H - G$. Une contrainte implicite est utilisée dans cet exemple : à chaque niveau d'exploration, seules les villes directement reliées sont prises en compte.

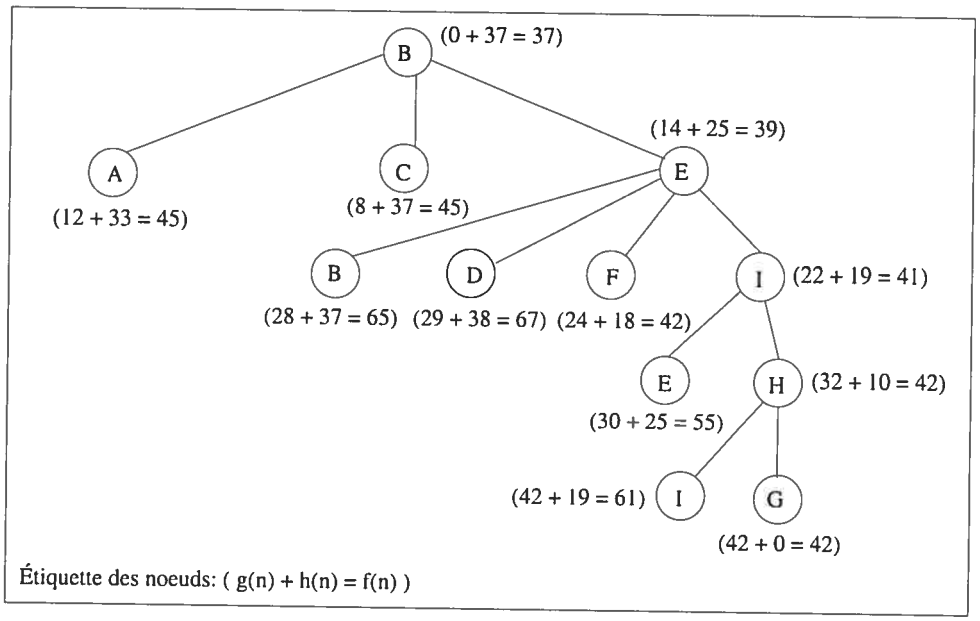

FIG. 1.8 – Exemple d'utilisation de A^* : étapes de la recherche

La complexité de A^* a fait couler beaucoup d'encre vers le début des années 1980. Pohl [Poh70, Poh77] et Gaschnig [Gas79] ont montré que la complexité de A^* est, en pire cas, exponentielle dans la taille de la solution lorsque, pour tout état n, l'erreur relative $\left(\frac{h(n)-h^*(n)}{h^*(n)}\right)$

demeure constante dans la taille de la solution. Ils ont aussi montré que lorsque l'erreur absolue $(h(n) - h^*(n))$ est constante, la complexité en pire cas est linéaire dans la taille de la solution. En 1984, Pearl [Pea84] a démontré un résultat intermédiaire : il est possible de limiter la complexité de A^* au logarithme de la taille de la solution si $(h(n) - h^*(n)) \leq O(\log(h^*(n)))$.

La difficulté de la planification A^*, c'est de donner un coût aux plans et de calculer une sous-estimation de ce coût pour des plans partiels. C'est sans doute la raison pour laquelle la recherche A^* n'a pas fait sa marque en planification. Nous croyons donc qu'une démonstration de la puissance de A^* dans certaines classes de problème de planification constitue une contribution intéressante à la recherche. C'est d'ailleurs l'approche que nous utiliserons pour résoudre notre problème de planification de voyages. Le chapitre 3 sera consacré à une explication complète de la manière dont nous avons utilisé la planification A^*.

1.3 Planification en incertitude

La planification logique fait de nombreuses hypothèses simplificatrices dont certaines sont très fortes. Pollack et Horty [PH99] font ressortir six hypothèses faites par les techniques de planification classiques :

- L'agent est omniscient, il connaît tout de son environnement.
- Les actions pouvant être exécutées par l'agent ont des conséquences déterministes.
- Les objectifs de l'agent sont catégoriels, c'est-à-dire qu'ils sont complètement atteints ou qu'ils ne le sont pas du tout ; il n'y a pas de notion d'atteinte partielle des objectifs.
- L'agent est la seule source de changement dans son environnement ; il est seul dans un monde statique.
- Les objectifs de l'agent sont fixes tout au long de la planification et de l'exécution du plan.
- Les actions faites par l'agent ont un effet instantané.

Les techniques de planification en incertitude permettent de lever certaines de ces hypothèses, proposant ainsi des solutions de planification applicables à des problèmes plus réalistes que ceux de la planification logique classique. Plusieurs approches ont été proposées dans la littérature depuis une dizaine d'années. Nous présenterons les principales en faisant ressortir

leurs forces et leurs faiblesses.

1.3.1 Planification conditionnelle

Plutôt que de produire une séquence d'actions fixe, la procédure de planification conditionnelle génère un plan dans lequel la séquence d'actions peut s'adapter aux observations faites par l'agent lors de l'exécution du plan. Cette technique peut tenir compte d'une incertitude aussi bien au niveau de l'état de l'environnement que des effets de certaines actions. Par exemple, un robot cuisinier pourrait concevoir le plan de préparation d'un repas sur le BBQ, mais s'il pleut au moment de la mise en oeuvre du plan, ce dernier ne pourra pas être exécuté en raison d'un changement imprévu de l'état du monde. Un robot jouant au casino devrait de son côté avoir à gérer une incertitude face à ses actions, les conséquences d'une mise étant toujours imprévisibles.

Plutôt que de refaire une planification au fur et à mesure que l'incertitude se dissipe, la planification conditionnelle génère un plan qui tient compte de toutes les possibilités. Il s'agit d'un plan sous forme d'arbre où chaque changement de branche représente les observations devant être faites lors de l'exécution. La figure 1.9 illustre un exemple de plan conditionnel pour un robot devant préparer le repas. Dès le début de l'exécution du plan, une observation de l'environnement est prévue afin de vérifier le temps qu'il fait. Le plan comporte une branche pour chaque résultat possible de cette observation. Ainsi, le robot pourra préparer le repas sur le BBQ s'il fait beau et il devra cuisiner à l'intérieur s'il pleut.

Développé en 1976, *Warplan-C* [War76] fut le premier planificateur conditionnel. Il créait des plans comme celui de la figure 1.9, mais limitait les actions d'observation à deux : une proposition p et sa négation ($\neg p$). En 1992, Peot et Smith [PS92] proposaient *CNLP*[¶], un autre planificateur conditionnel basé sur les opérateurs *STRIPS* mais ne limitant pas l'expression des résultats des actions d'observations comme c'était le cas pour *Warplan-C*. Peot et Smith soulignent la nécessité de contraindre le nombre de ces actions d'observation afin d'éviter une explosion de la taille du plan. Ils précisent d'ailleurs que *CNLP* pourrait difficilement être utilisé tel quel, la taille des plans produits étant justement soumise à une croissance exponentielle

[¶] Conditional Nonlinear Planner

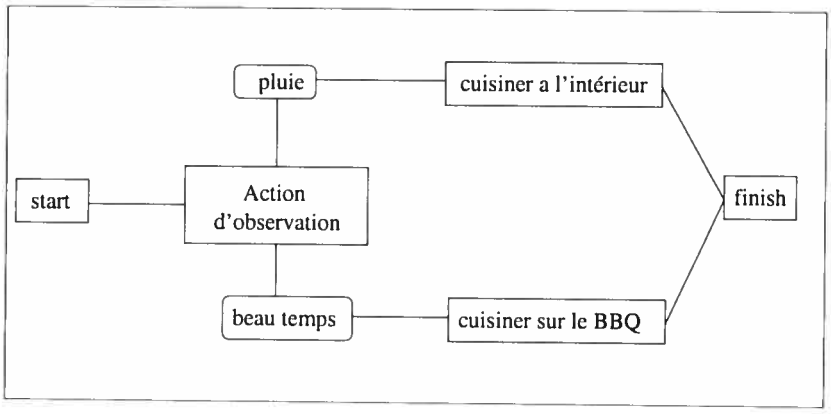

FIG. 1.9 – Exemple de plan conditionnel

dans le nombre d'actions d'observation.

Ce problème de croissance exponentielle fait de la planification conditionnelle une stratégie difficilement utilisable. Elle a toutefois le mérite d'avoir ouvert la voie à une planification plus réaliste en admettant que l'agent n'ait pas toujours une connaissance parfaite de son environnement, même s'il doit le connaître suffisamment bien pour prévoir un plan valide dans toutes les circonstances possibles.

1.3.2 Planification probabiliste

La planification conditionnelle permettait de lever partiellement l'hypothèse d'omniscience des agents en admettant qu'il soit possible qu'ils ne connaissent pas parfaitement l'état du monde quand ils élaborent leurs plans. Par contre, cette approche nécessite que l'information sur l'environnement soit déterministe afin que l'agent planificateur puisse produire un plan valide dans toutes les circonstances. La planification probabiliste permet de lever les deux hypothèses : l'information sur les états du monde n'a à être ni complète, ni déterministe. De plus, l'agent n'a pas à connaître parfaitement les effets de ses actions. Ces incertitudes l'empêchant de générer un plan toujours valide, le planificateur probabiliste produira un plan *ayant de fortes chances de réussir*, c'est-à-dire un plan dont la probabilité de succès est supérieure à un seuil quelconque.

Kushmerick, Hanks et Weld [KHW95] présentent le système de planification probabiliste *BURIDAN* basé lui aussi sur une extension des opérateurs *STRIPS*. Cette extension permet aux opérateurs de *BURIDAN* d'avoir des conséquences non-déterministes. Les opérateurs sont représentés par des arbres binaires dans lesquels les feuilles correspondent aux conséquences possibles de l'action qui est à la racine de l'arbre. Les noeuds internes décrivent l'état du monde permettant d'obtenir l'effet défini dans la feuille située au bout de sa branche et une probabilité est associée à chaque transition vers une feuille. La somme des probabilités d'une pré-condition vers ses deux enfants doit être de 1. Les chemins pour tous les effets de l'action doivent être *mutuellement exclusifs et exhaustifs*, c'est-à-dire qu'exactement une conséquence sera réalisée à la suite de l'exécution de l'action.

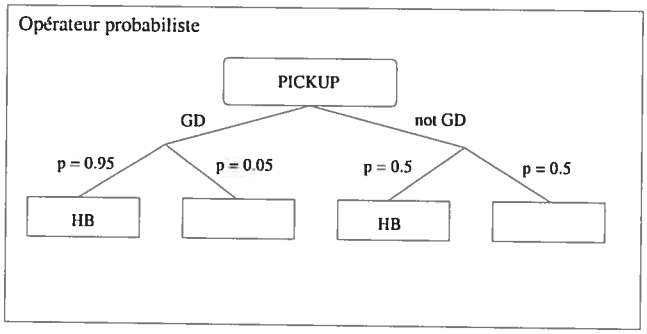

FIG. 1.10 – Exemple d'opérateur probabiliste

La figure 1.10 illustre un exemple d'opérateur probabiliste qui s'apparente au monde des blocs. Nous supposons que les opérateurs de déplacement de blocs définis précédemment sont des abstractions d'actions plus fines comme de prendre un bloc. L'opérateur *PICKUP* de la figure 1.7 représente l'action de prendre un bloc en vue de le déplacer. Elle peut être exécutée lorsque le bras du robot est sec (GD) ou mouillé ($\neg GD$). L'effet de cette action sera de tenir un bloc (HB) si l'opération réussit. Les probabilités de réussite dépendent de l'état s du bras du robot au moment où l'action sera exécutée. Par exemple, si $s = \{\neg HB, GD\}$ et que l'état à atteindre $s' = \{HB\}$, alors $P(s'|s, PICKUP) = 0.95 \cdot 1 = 0.95$.

Les états du monde sont représentés par des distributions de probabilités sur les valeurs des variables qui les définissent. L'objectif du planificateur probabiliste est de générer une séquence d'actions permettant de passer d'une distribution de probabilités initiale vers une autre distri-

bution dans laquelle le but à atteindre a de fortes chances de se réaliser. La planification débute avec un plan contenant la distribution de probabilités initiale et l'état final à atteindre et se fait par une recherche dans l'espace des plans partiels. On vérifie d'abord si la probabilité que l'objectif soit atteint est supérieure au seuil au-delà duquel un plan est acceptable. Si oui, la planification est terminée. Autrement, le planificateur cherche à augmenter cette probabilité en raffinant le plan, c'est-à-dire en ajoutant des opérateurs ou en résolvant des conflits dans le but d'augmenter la probabilité de succès du plan.

Cette approche n'apporte rien au niveau de la stratégie de planification qui demeure celle d'un planificateur classique. L'incertitude exprimée par les opérateurs et l'évaluation des plans sur la base de leur probabilité de succès constitue l'intérêt principal de la planification probabiliste. Toutefois, l'application des stratégies de recherche de la planification logique sur un problème complexifié par l'ajout d'incertitude accentue les problèmes de performances. De plus, la planification probabiliste fait l'hypothèse très forte que le planificateur connaît les distributions de probabilités sur les variables définissant les états du monde, ce qui est très rare dans des applications réelles.

1.3.3 Planification conditionnelle probabiliste

La planification conditionnelle et la planification probabiliste ont chacune leurs avantages lorsque l'agent évolue dans un environnement incertain. La planification conditionnelle permet à l'agent de choisir la branche du plan correspondant à l'état du monde qu'il observe lors de l'exécution. Alternativement, la planification probabiliste permet de générer un plan qui a de fortes chances de réussir lorsqu'il est impossible d'observer l'état du monde avec certitude. Les deux approches font clairement des hypothèses opposées. La planification conditionnelle suppose que l'agent ne connaît pas la distribution de probabilités des variables de son environnement mais peut observer avec certitude l'état du monde lorsqu'il exécute son plan. Par opposition, la planification probabiliste suppose que l'agent connaît la distribution de probabilités des variables de son environnement mais est incapable de faire une observation sûre de l'état du monde.

Il semble donc assez naturel de penser qu'une combinaison des deux approches pourrait

donner des résultats intéressants. Les conclusions des auteurs des deux planificateurs conditionnels probabilistes les plus cités (*C-BURIDAN* [DHW94] et *Mahinur* [OP99]) indiquent que cette voie est effectivement prometteuse. Son principal avantage serait de permettre un usage parcimonieux des actions d'observation (qui tendent à faire exploser la taille des solutions) tout en capturant l'incertitude (opérateurs probabilistes) sur les éléments ne nécessitant pas absolument une observation de l'environnment. Autrement dit, la planification conditionnelle probabiliste permet d'inclure des actions d'observation lorsque c'est critique au succès du plan et de gérer les autres éléments incertains avec des distributions de probabilités.

1.3.4 Processus de décision markoviens

Certaines classes de problèmes de planification peuvent être modélisées comme des processus de décision markoviens. C'est le cas, par exemple, des problèmes dans lesquels l'agent évolue dans un environnement dynamique pouvant être modélisé par un processus stochastique et où les actions peuvent influencer le comportement de ce processus. Ainsi, l'état dans lequel se trouve l'agent et la décision qu'il prend déterminent conjointement la distribution de probabilités des prochains états. Nous n'utiliserons pas ce type de modèle dans notre application car notre problème ne se prête pas bien à ce genre de modélisation de l'incertitude. Comme une explication, même minimale, des processus de décision markoviens serait relativement longue, nous vous référons à un article de Boutilier, Dean et Hanks [BDH99] faisant un bon survol de l'utilisation de ces processus dans la planification.

1.4 Planification et théorie de la décision

L'utilisation d'éléments de la théorie de la décision dans une optique de planification a pour objectif de nuancer l'évaluation des plans. Comme nous le disions plus tôt, la planification logique évaluait les plans sur une base binaire suivant qu'ils permettent ou non d'atteindre les objectifs fixés. Haddawy et Hanks [HH98] soutiennent que les modèles de planification basés sur la satisfaction d'un objectif binaire ou catégoriel sont limités parce que :

– ils définissent le succès du plan seulement en fonction de l'état du monde à la fin de

l'exécution ;

- ils n'accordent aucune valeur aux objectifs partiellement atteints ;
- ils considèrent deux plans permettant d'atteindre le même but comme équivalents, sans tenir compte de leur efficacité respective, tant sur le plan temporel qu'au niveau de la consommation de ressources ;

Une alternative basée sur la théorie de la décision pour la résolution de problème[‖] permettrait d'évaluer les plans en fonction des préférences de l'agent et des effets de la mise en application des plans. Pour ce faire, nous devons avoir une représentation des préférences permettant de comparer facilement les plans entre eux et nous devons établir une manière d'intégrer une évaluation continue des plans aux algorithmes de planification existant.

1.4.1 Théorie microéconomique et évaluation des plans

Il convient de faire une brève introduction à la théorie de l'utilité avant d'expliquer comment elle peut servir dans l'évaluation des plans. Nous discuterons essentiellement de la façon dont les économistes traduisent les préférences d'un individu en une fonction monotone et continue permettant de comparer facilement des paniers de biens difficilement comparables autrement.

Les préférences ont pour objectif de permettre un classement entre différents biens. Soit X un ensemble de biens et $x \in X, y \in X$ deux éléments de cet ensemble. Des préférences sur X pourraient permettre de dire, par exemple, que x est au moins aussi désirable que y ($x \succeq y$), que x est strictement préféré à y ($x \succ y$) ou que l'agent économique auquel appartiennent ces préférences est indifférent entre x et y ($x \sim y$). Afin de pouvoir exprimer les préférences sous la forme d'une *fonction d'utilité*, nous devons poser les trois axiomes suivants sur les préférences :

1. La complétude : toutes les paires d'éléments de X peuvent être comparées.
2. La réflexivité : $\forall x \in X, x \succeq x$.
3. La transitivité : $\forall x, y, z \in X, (x \succeq y) \wedge (y \succeq z) \rightarrow (x \succeq z)$.

[‖] decision theoretic for problem solving

En posant ces trois axiomes de même qu'un quatrième plus subtil concernant la continuité des préférences, il est toujours possible d'obtenir une *fonction d'utilité* $U : X \rightarrow \mathcal{R}$ représentant les préférences de telle sorte que $\forall x, y \in X, x \succeq y \iff U(x) \geq U(y)$. Cette fonction d'utilité traduit les préférences d'un agent économique sur l'ensemble de biens X et permet de comparer des paniers de ces biens en donnant à chacun une valeur dans l'ensemble des nombres réels. Nous sommes conscients que cette introduction est très brève et incomplète, aussi nous vous référons à l'ouvrage sur la théorie microéconomique de Mas-Colell, Whinston et Green [MCWG95] si vous souhaitez en savoir plus.

En planification, les plans produits peuvent être considérés comme des paniers de biens. Attention, ce ne sont pas les opérateurs du plan qui constituent les biens en question mais plutôt l'ensemble des effets produits par l'application du plan. Ainsi, une fonction d'utilité permettant d'évaluer correctement un plan devrait traduire les préférences de l'agent sur l'ensemble des conséquences possibles de la mise en oeuvre du plan. Par exemple, dans le monde des blocs, la fonction d'utilité ne devrait pas représenter des préférences sur le fait de déplacer un bloc plutôt qu'un autre. Ce sont les préférences sur l'ensemble des états du monde produits par le plan qui sont importantes. Ainsi, un agent qui aimerait beaucoup que le bloc A ne soit jamais posé sur la table ne devrait pas représenter cette préférence comme une aversion à utiliser l'opérateur *MOVE(A,X,table)* mais plutôt comme un désir d'éviter de passer par un état du monde dans lequel la proposition $on(table, A)$ serait vraie.

Définir les préférences sur les états plutôt que sur les opérateurs n'empêche pas d'avoir des préférences quant à la consommation de ressources. Les ressources en question doivent simplement faire partie des variables définissant les états du monde et leur consommation doit être l'un des effets de l'application des opérateurs Par exemple, si le déplacement d'un bloc occasionne une consommation d'énergie, alors les états du monde devraient inclure le niveau d'énergie du robot en plus de la position des blocs. L'application d'un opérateur aurait pour effet de passer à un état du monde dans lequel le robot aurait moins d'énergie et où un bloc aurait changé de position.

1.4.2 Intégration d'une évaluation continue des plans dans les algorithmes de planification

La recherche d'un plan revient à comparer plusieurs plans alternatifs et à choisir le meilleur. Au pire, ceci implique de calculer l'utilité pour tous les plans, ce qui nécessite le calcul de tous les états produits par chaque plan. Évidemment, tous ces calculs prennent beaucoup de temps et il est clair qu'une stratégie permettant de faire une comparaison efficace entre des plans doit être utilisée.

La stratégie de Haddawy et Hanks [HH98] consiste à comparer des plans partiels en cherchant à déterminer si un plan en domine un autre, c'est-à-dire que l'un aura nécessairement une utilité espérée supérieure à l'autre lorsqu'ils seront complets. Ainsi, à partir du moment où le plan $\mathcal{P}_1$ domine $\mathcal{P}_2$, le planificateur pourra définitivement exclure $\mathcal{P}_2$ même si aucun des deux plans n'est encore complet. Cette façon de faire réduit le nombre de plans à comparer et réduit aussi l'espace de recherche. Les travaux de comparaison et de recherche de plans en sont donc simplifiés. On considère que $\mathcal{P}_1$ domine $\mathcal{P}_2$ lorsque la borne minimale de $U(\mathcal{P}_1)$ est supérieure à la borne maximale de $U(\mathcal{P}_2)$.

1.5 Résumé

Ancêtres des planificateurs modernes, les planificateurs logiques ont légué une structure et un formalisme encore très utilisés aujourd'hui. La représentation *STRIPS* et le principe de raffinement itératif des plans partiels demeurent à la base de plusieurs planificateurs. Les difficultés rencontrées par la planification logique face à des applications réalistes ont motivé le développement d'extensions permettant notamment de planifier dans un environnement incertain. La planification conditionnelle et probabiliste a ouvert la voie à de nouvelles représentations des opérateurs et des plans qui pourraient être utiles dans plusieurs applications. L'évaluation des plans sur la base de fonctions d'utilité permet d'enrichir énormément les modèles de planification et surtout, donne un puissant outil heuristique pour diriger la planification. Avec la tendance actuelle à délaisser les planificateurs génériques au profit de solutions à des problèmes de plus en plus pointus, cette approche économique de la planification

deviendra probablement de plus en plus populaire.

Dans les chapitres suivants, nous proposerons un modèle simple pour la problématique de la planification de voyages dans un environnement dynamique et incertain. Ce modèle utilisera la représentation *STRIPS* décrite dans ce chapitre. Nous montrerons ensuite comment les préférences du voyageur peuvent servir d'heuristique de recherche dans une planification A^*.

Chapitre 2

Modélisation du problème de planification de voyages

L'objectif de notre travail n'est pas seulement de développer un algorithme de planification avancée. Nous souhaitons aussi proposer une solution à un problème appliqué bien précis nécessitant la gestion des préférences de l'agent et d'un environnement incertain : la planification de voyages.

Ce chapitre présente la modélisation détaillée de notre problème. Nous y définissons l'environnement dans lequel le planificateur devra évoluer, le modèle d'incertitude et le modèle des préférences des voyageurs. Nous commençons par une brève mise en situation de notre travail.

2.1 Mise en situation

Notre projet s'inscrit en complément du projet NADIM* du CIRANO† qui a pour objectif de réaliser un système d'assistance à l'achat de voyages [BAVGL+02]. Le projet est décomposé en quatre grands thèmes de recherche : la cueillette d'information, la représentation des préférences des voyageurs, la planification d'un voyage ou circuit touristique et la mise en

*Negotiating Agents on DIstributed Markets

†Centre Inter-universitaire de Recherche en ANalyse des Organisations

application d'un plan d'achats.

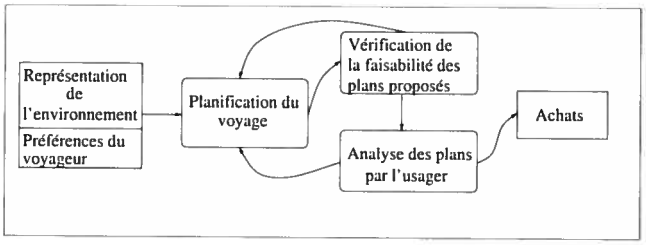

FIG. 2.1 – Système automatisé d'aide à la planification de voyages

La cueillette d'information est faite par des agents consultant les sites Internet des fournisseurs (avions, hôtels) afin d'en extraire l'information pertinente à l'organisation d'un voyage. Ces agents sont spécialisés dans la traduction d'un site particulier, c'est donc dire qu'il y a autant d'agents différents qu'il y a de sites Internet offrant de l'information pertinente. Ces informations constituent l'état du monde pour l'agent planificateur qui devra produire un plan en fonction des informations qu'il détient sur les prix, disponibilités, horaires, etc. et aussi en fonction des préférences des voyageurs.

Nous allons tenter de faire un planificateur trouvant toujours *la meilleure* solution. Mais, en raison de l'incertitude de l'environnement, il se peut que cette meilleure solution s'avère insatisfaisante et qu'elle soit rejetée par le voyageur. Nos données n'étant pas systématiquement à jour, l'algorithme ne peut que trouver la solution ayant les meilleures chances de satisfaire ses objectifs. Avant d'être exécutée, cette solution devra être vérifiée afin d'en éliminer l'incertitude. C'est après cette vérification qu'elle pourrait être rejetée. Il se peut donc que le planificateur soit appelé à refaire plusieurs fois le travail de planification à la lumière des nouvelles informations recueillies lors de la vérification. Afin d'éviter cela, il serait préférable de produire une suite de plans ordonnés selon leur espérance d'utilité de telle sorte que le premier soit celui ayant le plus de chances d'être le meilleur.

Une fois un plan accepté, les achats nécessaires devront être faits, possiblement selon un plan d'achat précis destiné à minimiser les probabilités d'avoir à débourser pour la mise en oeuvre de transactions compensatoires[‡]

[‡]Une action compensatoire consiste à annuler une transaction d'achat.

Notre contribution au projet se situe au niveau du module de planification. Nous estimons que la problématique du voyage, de par le très grand dynamisme de son environnement, la complexité des préférences des voyageurs et l'incertitude reliée aux informations disponibles, constitue un excellent banc d'essai pour des algorithmes de planification avancée.

2.2 Définition de l'environnement

Le projet NADIM en étant encore à ses balbutiements, nous n'avons pu utiliser de véritables agents pour faire la cueillette d'information. Aussi avons-nous dû simuler un environnement ainsi que l'incertitude qui s'y attache. Cette section décrit les paramètres de simulation de notre environnement et la section suivante discute du modèle d'incertitude que nous avons développé. À ce stade, nous avons cherché à modéliser un environnement ayant un ordre de complexité égal à celui du monde réel sans nous attarder à développer un modèle parfaitement conforme à la réalité.

2.2.1 Modèle géographique

Qui dit voyage, dit itinéraires, villes, logements, déplacements, etc. Ce sont ces éléments que nous avons simulés dans notre environnement. Notre monde modélisé compte 4 pays, chacun ayant trois villes dont une capitale. Chaque pays a sa compagnie aérienne qui fait la liaison entre ses trois villes et entre sa capitale et les 3 autres capitales. Toutes les villes comptent un certain nombre d'hôtels, les capitales ayant une capacité de logement supérieure aux autres villes. Formellement, le monde compte :

- 12 villes : $v_1, v_2, ..., v_{12}$
- 4 capitales : v_3, v_4, v_9, v_{12}
- 4 compagnies aériennes : a_1, a_2, a_3, a_4
- 3 types d'hôtel : h_1, h_2, h_3

2.2.2 Modèle du temps

Avant de poursuivre avec les détails concernant les compagnies aériennes et les hôtels, il est important d'exposer notre modélisation du temps. Celle-ci décompose le temps en unités discrètes correspondant à une heure. Ainsi, une journée est représentée par l'intervalle $[t_a, t_b]$ où $b - a = 24$, une semaine par l'intervalle $[t_a, t_b]$ où $b - a = 168$ et une année par l'intervalle $[t_a, t_b]$ où $b - a = 8736$.

Ce modèle du temps simplifie la gestion des aspects temporels du problème. Une discrétisation à l'heure près ne limite pas réellement l'expressivité du modèle puisqu'il serait possible d'utiliser une discrétisation plus fine (à la minute près par exemple). L'abscence de différenciation entre les heures constitue la plus grande lacune de notre modèle du temps. Il n'y a pas de distinction entre le jour et la nuit, entre la fin de semaine et la semaine, entre les vacances de Noël et le reste de l'année etc. De plus, ceci fait en sorte que les chambres d'hôtels puissent être louées à l'heure près, ce qui ne correspond pas à la réalité. Nous ne croyons toutefois pas que ces limites nuisent à la démonstration que nous souhaitons faire car un modèle de temps plus réaliste ne serait pas incompatible avec l'algorithme de planification que nous proposons et qui constitue l'élément principal de notre travail. Il faut bien comprendre que le développement d'un modèle réaliste pour une réalité aussi complexe que l'organisation de voyages est une très lourde tâche en soit ; aussi avons-nous décidé de limiter notre effort de modélisation afin de nous concentrer sur l'algorithme de planification.

2.2.3 Modélisation de la qualité

Comme vous le verrez dans les sous-sections qui suivent, notre environnement est composé de billets d'avions et de chambres d'hôtels. Nous devrons comparer ces produits selon leurs prix, contraintes temporelles et qualité. Le prix et les contraintes de temps sont assez naturelles puisqu'il s'agit de concepts que nous avons l'habitude d'utiliser. Par contre, la qualité d'un produit est plus difficilement mesurable. Pour les avions, nous utilisons habituellement le système de classes pour définir la qualité d'un billet : un billet en première classe devrait être de meilleure qualité qu'un billet en classe économique. Pour les hôtels, nous utilisons plutôt

un système d'étoiles dans lequel un nombre élevé d'étoiles est normalement un gage de grande qualité.

Afin de pouvoir comparer facilement les avions avec les hôtels et aussi afin d'avoir une mesure continue de la qualité, nous avons établi notre propre métrique. Plutôt que de parler de classes ou d'étoiles, nous parlerons de *points de qualité*. Ces points représenteront une note entre 0 et 100 correspondant à la qualité d'un avion ou d'un hôtel. De la même manière que la monnaie permet de comparer la valeur de deux produits fondamentalement différents, les points de qualité nous permettront de comparer la qualité de deux produits différents.

2.2.4 Les compagnies aériennes

Les compagnies aériennes vendent des places sur leurs différents vols. Un billet correspondant à l'une de ces places est défini par les attributs suivants : compagnie aérienne, ville de départ, ville d'arrivée, moment de départ, moment d'arrivée et prix. Chaque transporteur a ses propres caractéristiques de vitesse, de qualité et de prix. La vitesse est donnée en unité de distance par heure (udd/h), la qualité est mesurée en points de qualité et le prix est proportionnel à la distance d à parcourir et à la qualité offerte.

Transporteur	Vitesse	Pts de qté	Prix ($/udd)
a_1	4udd/h	100	$N(100d, 10d)$
a_2	4udd/h	50	$N(50d, 5d)$
a_3	2udd/h	90	$N(50d, 5d)$
a_4	2udd/h	30	$N(25d, 5d)$

TAB. 2.1 – Caractéristiques des compagnies aériennes

Le tableau 2.1 résume les caractéristiques des compagnies aériennes. Les prix y sont exprimés sous la forme de distributions normales, la notation $N(a, b)$ se lisant "distribution normale avec une espérance de a et une variance de b". Ces distributions sont utilisées pour la simulation de l'environnement afin que tous les avions d'une même compagnie aérienne desservant une même liaison n'affichent pas nécessairement le même prix. Elles n'ont donc rien à

voir avec notre modèle d'incertitude que nous présentons d'ailleurs un peu plus loin.

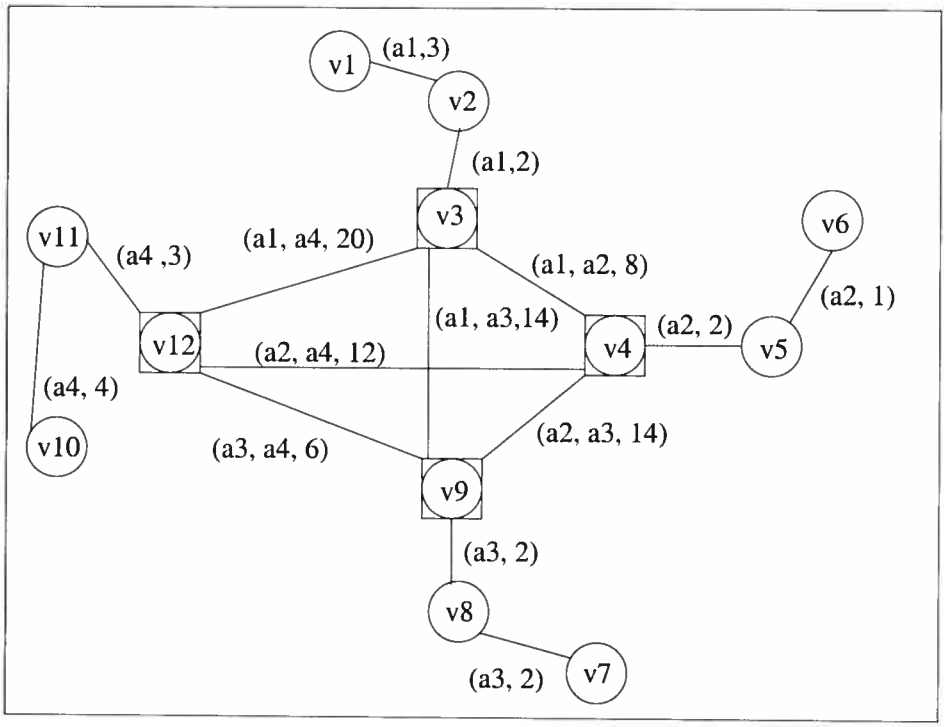

FIG. 2.2 – Graphe des liaisons aériennes entre les villes

La figure 2.2 illustre les liaisons aériennes entre les villes. Sur ce graphe, les arcs correspondent à des liens aériens et leurs étiquettes indiquent les compagnies desservant le lien et la distance entre les deux villes. Remarquez que toutes les distances vérifient l'inégalité triangulaire voulant que la longueur d'un côté soit toujours inférieure ou égale à la somme de la longueur des deux autres. Les villes encadrées représentent les capitales des quatre pays. Les liaisons offertes par les transporteurs dépendent de leur pays d'appartenance. Dans tous les cas, une compagnie aérienne dessert toutes les villes de son pays et relie sa capitale aux autres capitales. La fréquence des vols est toujours plus grande pour les liens locaux que pour les liaisons internationales.

2.2.5 Les hôtels

Les hôtels offrent des chambres, permettant au voyageur de se loger lors de son séjour dans une ville. Une chambre d'hôtel est définie par : le type d'hôtel, la ville et le prix pour un intervalle de temps précis. Le monde comprend 3 différents types d'hôtels : $\{h_1, h_2, h_3\}$ chacun ayant des caractéristiques précises quant à sa qualité (en points de qualité) et sa distribution de prix.

La disponibilité des chambres varie selon les hôtels. Nous simulons ceci en distribuant dans le temps des périodes durant lesquelles un hôtel est complet. Ainsi, tout l'horizon temporel d'un hôtel est alloué, en alternance, soit à une période durant laquelle des chambres sont disponibles, soit à une période durant laquelle l'hôtel est complet. La durée de ces périodes n'est pas déterministe et suit une distribution différente selon les types d'hôtels. Dans le tableau 2.2, F_{disp} représente la fonction de distribution de la durée (en heures) des périodes durant lesquelles une chambre est disponible et F_{nd} représente la durée des périodes durant lesquelles l'hôtel est complet. Comme pour les avions, toutes les distributions normales définies ici n'ont rien à voir avec notre modèle d'incertitude. Il s'agit seulement des paramètres utilisés pour la simulation de notre environnement afin de créer de la diversité parmi les hôtels.

Type	Pts de qté	Prix ($/jour)	F_{disp}	F_{nd}
h_1	10	$N(50, 5)$	$N(288, 96)$	$N(360, 120)$
h_2	60	$N(125, 12.5)$	$N(480, 92)$	$N(432, 72)$
h_3	100	$N(300, 30)$	$N(480, 144)$	$N(240, 72)$

TAB. 2.2 – Caractéristiques des types d'hôtels

Les hôtels sont répartis de manière uniforme entre les villes, suivant qu'il s'agisse d'une capitale ou d'une ville de province. Les capitales comptent 35 hôtels : 10 de type h_1, 15 de type h_2 et de type 10 h_3. Les villes de province comptent 30 hôtels : 15 de type h_1, 10 de type h_2 et de type 5 h_3. Comme notre monde compte 12 villes dont 4 capitales, il y a en moyenne 12 hôtels h_1 par ville, 13 hôtels h_2 par ville et 8 hôtels h_3 par ville.

Remarquez que la disponibilité des hôtels dépend de la durée du séjour. Il est tout à fait

possible qu'un hôtel soit disponible pour 3 jours mais pas pour 4 jours. Nous avons fait une simulation de *Monte Carlo* afin d'avoir une idée de la probabilité qu'un hôtel quelconque soit disponible pour une semaine (168 heures) dans une ville quelconque. Cette simulation utilisait les distributions du tableau 2.2 pour la durée des périodes de disponibilité et de non disponibilité et la répartition des hôtels présentées plus tôt. La simulation nous apprend que la disponibilité moyenne d'un hôtel quelconque dans une ville de notre monde pour un séjour d'une durée de 168 heures est de 30%.

2.3 Modélisation de l'incertitude

2.3.1 Modèle général

L'incertitude dans notre environnement est causée par le décalage entre le moment où l'agent a obtenu une information et le moment où il réalise son travail de planification. Notre modèle veut que le planificateur ait accès à une base de données contenant des informations relatives aux billets d'avion et aux chambres d'hôtels. Comme nous le disions plus tôt, cette base de données devrait idéalement être bâtie par des agents fureteurs recueillant de l'information sur Internet. Or, nous supposons que la quantité d'information concernant les voyages est trop grande et trop dynamique pour que des agents puissent maintenir l'ensemble de la base de données à jour. Nous associons donc une date à chaque élément d'information (billet d'avion, hôtel) et nous définissons l'incertitude associée à chacun comme une fonction monotone croissante de l'âge de l'information. Ainsi, plus une information est vieille, plus elle est incertaine.

Nous avons choisi de modéliser toutes les variables incertaines par des distributions normales centrées sur la valeur obtenue par l'agent fureteur et dont la variance croît avec l'âge de l'information. Nous avons opté pour des lois normales en raison de leurs propriétés de combinaison linéaire qui nous sera fort utile au moment de calculer la distribution d'un plan composé de plusieurs opérateurs et où chaque opérateur sera composé de valeurs normalement distribuées.

La combinaison linéaire de distributions normales se fait comme suit : soit $x \sim N(\mu_x, \sigma_x^2)$ et $y \sim N(\mu_y, \sigma_y^2)$, alors la distribution d'une variable $z = ax + y$ sera :

$$z \sim N(a\mu_x + \mu_y, a^2\sigma_x^2 + \sigma_y^2) \tag{2.1}$$

Pour plus d'informations sur les distributions normales, nous vous référons au chapitre 3 du livre d'économétrie de Green [Gre00].

2.3.2 Incertitude sur les prix

L'environnement n'est incertain qu'au niveau des prix associés aux opérateurs. Cette incertitude est modélisée de la même façon pour les avions que pour les hôtels : l'agent connaît le prix p_i valable au temps i et doit en estimer la valeur au temps $t \geq i$ où il fait la planification. La valeur de p au temps i est donnée par :

$$p_i \sim N(p_i, 0) \tag{2.2}$$

Autrement dit, la valeur de p au temps i est certaine puisque la variance de sa distribution est nulle. Soit $\Delta t = t - i$, $(t \geq i)$, la différence entre un moment quelconque t et le temps i où p est obtenu. Plus Δt est grand, plus la variance de la distribution de p_t est grande. Sa moyenne demeure toutefois toujours la même, c'est-à-dire p_i. Ainsi, p_t est donnée par :

$$p_t \sim N(p_i, \sigma_{\Delta t}^2) \tag{2.3}$$

La variance de la distribution croît logarithmiquement avec Δt suivant un taux de croissance proportionnel à sa moyenne, p_i. Formellement,

$$\sigma_{\Delta t}^2 = \ln[\max(1, \Delta t)]0.05p_i \tag{2.4}$$

Le choix d'une fonction logarithmique pour représenter la croissance de l'incertitude est

purement arbitraire. Nous pourrions utiliser une autre fonction et même une autre loi que la loi normale. L'important pour nous est d'avoir un modèle quelconque pour représenter l'incertitude. Celui-ci nous paraît raisonnable, mais rien n'empêcherait d'en utiliser un autre, dans la mesure où la distribution est connue du planificateur et qu'il peut retrouver la fonction de distribution à partir de l'âge de l'information ou d'un autre signal quel qu'il soit.

Notre objectif ici est de fournir un modèle raisonnable de l'incertitude propre aux données électroniques sur les produits de voyage. Toutefois, comme vous le verrez au chapitre 3, l'incertitude aura très peu d'impact sur l'algorithme de planification. En fait, nous nous contenterons de travailler sur les moyennes des distributions de prix. L'algorithme serait donc le même s'il n'y avait pas d'incertitude.

2.4 Description formelle d'un plan et de ses opérateurs

2.4.1 Les opérateurs

Avant de discuter du plan lui-même, nous devons décrire les opérateurs qui seront utilisés dans sa séquence d'actions. Notre agent planificateur dispose de deux opérateurs :

- $avion(v_a, v_b, t_a, t_b, p, ca)$: permettant au voyageur de se déplacer de la ville v_a à la ville v_b sur les ailes de la compagnie ca pour un prix p. Le départ a lieu en t_a et l'arrivée en t_b ;
- $hotel(v, t_a, t_b, p, h)$: permettant au voyageur de se loger dans la ville v durant l'intervalle $[t_a, t_b]$ dans un hôtel de type h pour un prix p ;

Les deux opérateurs ont un certain nombre de pré-conditions et d'effets représentés en terme des prédicats $etre(v, [t_a, t_b])$ (signifiant que le voyageur se trouve dans la ville v pendant tout l'intervalle $[t_a, t_b]$) et $loger(v, [t_a, t_b])$ (signifiant que le voyageur loge dans la ville v durant l'intervalle $[t_a, t_b]$). Le prédicat $etre(\cdot)$ indique où se trouve le voyageur pendant une période de temps et le prédicat $loger(\cdot)$ précise si le voyageur a accès à une chambre d'hôtel durant son séjour dans une ville autre que sa ville de départ.

Opérateur	Pré-conditions	Effets
$avion(v_a, v_b, t_a, t_b, p, ca)$	$etre(v_a, [t_a, t_a])$	$etre(v_b, [t_b, t_\eta[), \neg etre(v_a, [t_a, t_\eta[)$
$hotel(v, t_a, t_b, p, h)$	$etre(v, [t_a, t_b])$	$loger(v, [t_a, t_b])$

TAB. 2.3 – Pré-conditions et effets des opérateurs

Dans le tableau 2.3, t_η représente l'horizon connu du planificateur, c'est-à-dire que la base de données ne contient aucun opérateur dont la date d'application est au-delà de t_η. La conséquence d'un déplacement (aller simple) est donc de se retrouver dans la ville de destination pour toujours, toujours étant limité à l'horizon connu du planificateur. Évidemment, rien n'empêche un voyageur de se procurer un billet de retour, auquel cas t_η serait remplacé par la valeur t_a du billet de retour.

2.4.2 Les objectifs

Avant de parler du plan proprement dit, nous devons décrire le problème à résoudre, c'est-à-dire le voyage à planifier. Un voyage est décrit par un ensemble de contraintes temporelles et géographiques contenant la liste des villes devant être visitées et des heures de départ et d'arrivée souhaitées pour chacune de ces villes. Le tableau 2.4 décrit un voyage constitué de trois contraintes. Le voyageur habite v_7 et souhaite quitter au temps 2200 pour faire un séjour d'une semaine en v_9 avant de revenir chez-lui. Voyons comment exprimer ce voyage simple en terme des prédicats $etre$ que nous avons définis plus tôt. Au début de l'horizon connu du planificateur (le temps 0), le voyageur est chez-lui (à v_7) et il souhaite partir au temps 2200. Nous pouvons en déduire une première contrainte : $etre(v_7, [0, 2200])$. Le voyageur souhaite ensuite passer une semaine (168 heures) dans la ville v_9. Comme nous considérons que le voyageur ne retire aucun plaisir à se déplacer entre deux villes, nous supposons qu'il aimerait arriver à v_9 au même moment qu'il quitte v_7, c'est-à-dire au temps 2200 (il rêve d'un déplacement instantané). Ce séjour d'une semaine à v_9 constitue une seconde contrainte que nous pouvons écrire : $etre(v_9, [2200, 2368])$. Finalement, le voyageur souhaite rentrer chez-lui instantanément et y rester jusqu'à la fin de l'horizon connu du planificateur (une année) donc : $etre(v_7, [2368, 8736])$. Afin de simplifier l'écriture, nous écrirons toujours les prédicats $etre$

d'un voyage sous la forme d'un tableau semblable au tableau 2.4.

Arrivée	Départ	Ville
0	2 200	v_7
2 200	2 368	v_9
2 368	*8 736*	v_7

TAB. 2.4 – Description du voyage à planifier

Remarquez qu'il n'y a rien dans la description du voyage concernant les hôtels. Le planificateur comprend implicitement que tout séjour de plus de 24 heures dans une ville autre que la ville de départ nécessite la réservation d'une chambre d'hôtel. Le voyageur n'a donc pas à le préciser explicitement.

2.4.3 Le plan

Un plan $\mathcal{P}_{t_l,t_n}$ débutant en t_l et se terminant en t_n est défini par :

- une liste d'étapes ou d'opérateurs,
- une liste des effets produits par l'application du plan,
- une liste des effets potentiels du plan,
- une liste de conditions à satisfaire.

La figure 2.3 montre un plan partiel (en cours de formation) pour un voyage simple dans lequel le voyageur habitant une ville de province souhaite passer une semaine dans la capitale de son pays (tableau 2.4). Nous utiliserons cet exemple pour expliquer chacune des listes qui composent le plan. Notre objectif n'est donc pas d'expliquer la façon dont un plan est construit (nous y reviendrons au chapitre 3), mais seulement de présenter les structures entrant dans sa composition. Nous avons choisi un plan partiel parce qu'il permet de justifier plus facilement la présence de certaines structures.

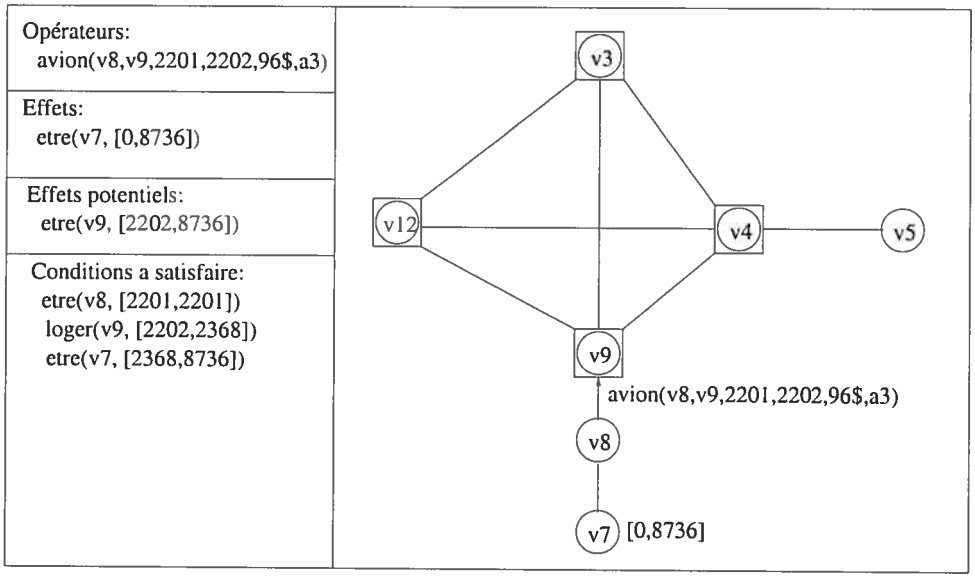

FIG. 2.3 – Exemple d'un plan partiel et ses effets sur l'état du monde

La liste des opérateurs

Comme dans tous les plans utilisant des opérateurs *STRIPS*, les plans produits par notre algorithme comportent une liste des opérateurs représentant la séquence d'étapes du plan. Dans l'exemple de la figure 2.3, le plan partiel ne comporte qu'un seul opérateur, un billet pour un avion de la compagnie a_3 partant de v_8 au temps 2201 et arrivant à v_9 une heure plus tard. Ce plan est bien sûr incomplet, d'autres opérateurs devront s'y ajouter pour qu'il puisse éventuellement être mis en oeuvre.

La liste des effets *potentiels*

Avant d'être considéré comme un effet réel d'un plan, chaque proposition doit d'abord faire un séjour dans l'ensemble des *effets potentiels*, ensemble que l'on retrouve dans chaque instance de plan. Cette étape qui peut sembler superflue est en fait essentielle. Même si un prédicat fait partie des effets d'un opérateur présent dans le plan, rien ne garantit que cet effet soit réellement *accessible*. Il se peut fort bien que l'opérateur engendrant l'effet en question soit lui-même impossible à appliquer lors de la mise en oeuvre du plan si toutes ses pré-conditions

ne sont pas encore satisfaites. Ainsi, tous les effets d'un nouvel opérateur sont d'abord placés dans cet ensemble d'effets potentiels pour ensuite être transférés vers l'ensemble des effets réels du plan lorsque toutes les pré-conditions de *toute la chaîne des opérateurs responsables de ces effets* seront satisfaites.

Dans le plan de la figure 2.3, les effets du seul opérateur du plan ont été placés dans la liste des effets potentiels justement parce que toutes les pré-conditions de cet opérateur ne sont pas satisfaites. Si ce plan était appliqué tel quel, le voyageur qui se trouve en v_7 au temps 2201 ne pourrait pas prendre un avion en partance de v_8. Les effets de cet opérateur demeureront donc des effets potentiels jusqu'à ce que ses pré-conditions soient satisfaites, c'est-à-dire jusqu'à ce que la proposition $etre(v_8, [2201, 2201])$ soit vraie. Si le plan était complet, cette liste serait vide.

La liste des effets

La liste des effets est une liste exhaustive des impacts qu'aurait la mise en oeuvre *immédiate* du plan. Il ne s'agit pas simplement de l'union des effets *accessibles* de tous les opérateurs du plan car il se peut que l'ajout d'un opérateur modifie les effets engendrés précédemment par un autre opérateur, notamment au niveau du temps. Dans notre exemple, l'unique effet est une conséquence de l'état du monde initial voulant que le voyageur se trouve en v_7. Mais, si le voyageur quittait cette ville au temps 2200 par exemple, alors la borne temporelle supérieure de $etre(v_7, [0, 8736])$ serait modifiée et le prédicat se lirait $etre(v_7, [0, 2200])$ signifiant que le voyageur n'est plus chez-lui du temps 0 au temps 8736, mais bien du temps 0 au temps 2200.

La liste des conditions à satisfaire

Une liste des conditions à satisfaire est maintenue durant la planification pour chaque instance de plan en formation. Elle est initialisée avec les objectifs du voyageur, comme c'est le cas pour les planificateurs logiques. Au fur et à mesure que des étapes sont ajoutées au plan, les pré-conditions de ces nouveaux opérateurs sont ajoutées à la liste. Les pré-conditions satisfaites par l'ajout d'un nouvel opérateur sont, quant à elles, retirées de la liste. Dans le plan partiel de

la figure 2.3, la proposition $etre(v_9, [2200, 2368])$ qui devait initialement faire partie des conditions à satisfaire, a été retirée après l'ajout de l'opérateur $avion(v_8, v_9, 2201, 2202, 96\$, a_3)$. Par contre, cet opérateur a aussi occasionné l'ajout de la condition $(etre(v_8, [2201, 2201]))$ nécessaire à l'application de l'opérateur. Comme pour la liste des effets potentiels, la liste des conditions à satisfaire est vide lorsque le plan est complet.

Remarques

Remarquez l'absence des ensembles classiques normalement associés à un plan, à savoir une liste d'ordonnancements des étapes du plan et un ensemble de liens causaux. Étant donné le caractère temporel du problème de planification de voyages, la liste d'ordonnancements est superflue. Les étapes du plan seront ordonnées naturellement selon les dates auxquelles elles s'appliquent. Quant à la liste de liens causaux, elle sert habituellement à détecter les conflits entre opérateurs afin de permettre une résolution par promotions et démotions. Encore une fois, le caractère temporel du problème nous oblige à exclure d'emblée la technique des promotions et démotions. Nous avons plutôt décidé de procéder à une construction chronologique du plan. Ainsi, nous pouvons ajuster les contraintes temporelles au fur et à mesure que le plan est construit, ce qui permet d'éviter les conflits plutôt que d'avoir à les résoudre. Nous reviendrons en détails sur la manière dont les plans sont construits au chapitre suivant.

En plus des ensembles évoqués ci-haut, un plan a toujours un coût monétaire et un coût de qualité. Le coût monétaire correspond à la somme totale à débourser pour obtenir l'ensemble des opérateurs qui le compose :

$$c_{\mathcal{P}} \sim N\left(\sum_{s \in S} \mu_{c_s}, \sum_{s \in S} \sigma^2_{c_s}\right) \tag{2.5}$$

où c_s correspond au coût monétaire d'un opérateur s.

Le coût de qualité représente la perte de qualité subie par l'ajout d'un opérateur par rapport à la qualité maximale qu'un tel type d'opérateur peut fournir. Par exemple, si la chambre d'hôtel parfaite a un niveau de qualité égal à 100, alors nous dirons d'une chambre qui a une qualité de 80 que son coût de qualité est de 20. Cette façon de mesurer le niveau de qualité peut sembler contre intuitive, mais elle se justifie par la nécessité de travailler uniquement sur des pertes

subies par le voyageur. Un coût de qualité permet de traiter la qualité de la même manière que nous traitons le coût monétaire, c'est-à-dire comme une perte que le voyageur doit assumer. Comme aucune incertitude n'est associée à la qualité, nous le représentons simplement comme la perte de qualité moyenne des opérateurs d'un plan :

$$cq_{\mathcal{P}} = \frac{\sum_{s \in S} cq_s}{|S|} \tag{2.6}$$

où cq_s correspond au coût en qualité d'un opérateur s.

2.5 Modélisation de la fonction d'utilité d'un voyageur

Afin de pouvoir comparer les plans entre eux, nous devons être en mesure d'attribuer une valeur réelle à chaque plan selon les préférences des voyageurs. Pour ce faire nous devons traduire ces préférences en une fonction $u(\mathcal{P}) : \mathcal{R}$ appelée fonction d'utilité. La fonction d'utilité du voyageur que nous allons présenter est sujette au coût du voyage, au respect des contraintes temporelles et au niveau de confort des avions et des chambres d'hôtel.

2.5.1 La contrainte de budget

Il s'agit ici de déterminer la perte d'utilité associée au coût du voyage. Nous proposons de modéliser cette utilité sous la forme d'une fonction linéaire décroissante valant zéro lorsque la contrainte de budget est atteinte. Ainsi, pour un plan $\mathcal{P}$ dont le coût total serait $c_{\mathcal{P}}$, l'utilité espérée associée au prix est donnée par :

$$\mu_{u(c_{\mathcal{P}})} = u_{max}(1 - \frac{c_{\mathcal{P}}}{budget}) \tag{2.7}$$

où u_{max} représente l'utilité maximale (si le voyage était gratuit) et $budget$ représente le montant maximal que le voyageur est prêt à payer. Il est important de remarquer que ce budget maximal n'est pas une contrainte ferme, c'est-à-dire que son dépassement n'occasionne pas nécessairement une utilité nulle pour le plan. Seule la partie de l'utilité associée au coût monétaire sera nulle, voir même négative. L'importance relative de cette utilité sera introduite

plus loin. Notez finalement que le choix d'une fonction linéaire pour représenter la relation entre le coût monétaire et son utilité est purement arbitraire. Toute autre fonction monotone décroissante pourrait être utilisée.

Comme les prix varient selon une fonction de distribution normale, $u(c_{\mathcal{P}})$ suivra aussi une loi normale :

$$u(c_{\mathcal{P}}) \sim N(\mu_{u(c_{\mathcal{P}})}, m^2\sigma^2_{\Delta t}) \tag{2.8}$$

2.5.2 Les contraintes de temps

Typiquement, le voyageur devrait demander un plan lui permettant de se trouver dans une ville v entre un temps t et un temps r. Exprimées comme ça, les contraintes de temps sont fermes, c'est-à-dire qu'aucune utilité n'est accordée au passage dans la ville v s'il n'a pas lieu entre les dates précisées. Mais le voyageur qui a un horaire plus souple pourrait spécifier sa contrainte comme suit : "je veux être dans la ville v entre $(t \pm \gamma)$ et $(r \pm \lambda)$" où γ et λ représentent l'écart maximal acceptable par rapport aux dates demandées. Ceci permet d'exprimer le fait de vouloir arriver à v autour de t pour repartir à peu près en r. L'utilité associée aux contraintes de temps devrait donc diminuer proportionnellement au respect des bornes temporelles. Nous proposons une décroissance linéaire dans les valeurs de ces deux variables telle que l'utilité associée au respect des contraintes temporelles soit nulle ou négative lorsqu'elles ne sont pas respectées. Encore une fois, le choix d'une fonction linéaire est arbitraire, toute autre fonction monotone décroissante pouvant tout aussi bien faire l'affaire.

Soit $C = \{..., (t_i \pm \gamma_i, r_i \pm \lambda_i), ...\}$ l'ensemble des contraintes de temps d'un voyageur. L'utilité espérée $\mu_{u(t_{\mathcal{P}})}$ associée à ces contraintes pour un plan $\mathcal{P}$ sera donnée par :

$$\mu_{u(t_{\mathcal{P}})} = \sum_{i=0}^{|C|} \left[\frac{u_{max}}{|C|} - m_{\gamma_i} |t_i^* - t_i| \right] + \sum_{i=0}^{|C|} \left[\frac{u_{max}}{|C|} - m_{\lambda_i} |r_i^* - r_i| \right] \tag{2.9}$$

où t_i^* et r_i^* représentent les valeurs réalisées correspondant aux contraintes t_i et r_i. L'équation 2.9 est composée de deux sommations : la première sur toutes les contraintes de temps corres-

pondant à une arrivée dans une ville (les t_i) et la seconde sur toutes les contraintes correspondant à un départ (les r_i). Expliquons la première sommation, l'explication étant identique pour la seconde. Dans cette sommation, $|t_i^* - t_i|$ représente l'écart entre le moment d'arrivée désiré t_i et le temps d'arrivée réalisé t_i^*. Plus cet écart est grand, moins l'utilité associée à la contrainte t_i sera élevée. Si l'écart était nul, l'utilité associée à t_i serait alors maximale : $\frac{u_{max}}{|C|}$. Comme il y a $|C|$ contraintes de temps, l'utilité espérée serait $|C| \times \frac{u_{max}}{|C|} = u_{max}$ si toutes les contraintes étaient parfaitement respectées. Les taux de décroissance m_{γ_i} et m_{λ_i} sont donnés par :

$$m_{\gamma_i} = \frac{u_{max}/|C|}{\gamma_i} \tag{2.10}$$

$$m_{\lambda_i} = \frac{u_{max}/|C|}{\lambda_i} \tag{2.11}$$

Comme λ_i et γ_i correspondent à l'écart à l'horaire maximal accepté par le voyageur, il est raisonnable que m_{λ_i} et m_{γ_i} leur soient inversement proportionnels.

Encore une fois, le non-respect des contraintes temporelles n'occasionnera pas directement la nullité de l'utilité globale du plan. Seule l'utilité liée aux contraintes de temps pourra être nulle ou même négative. Cette utilité suivra la distribution :

$$u(t_\mathcal{P}) \sim N(\mu_{u(t_\mathcal{P})}, 0) \tag{2.12}$$

2.5.3 Les contraintes de qualité

Nous supposerons ici que tous les voyageurs préfèrent plus de qualité. Ainsi, plus la qualité sera grande, plus l'utilité sera élevée. Seule l'importance relative de la qualité dans la valeur totale de l'utilité distinguera le voyageur soucieux de son confort des autres qui ne le sont pas. L'utilité espérée associée à la qualité d'un plan est donnée par :

$$\mu_{u(q_\mathcal{P})} = u_{max} - \left(\frac{u_{max}}{q_{max}}\right) c q_\mathcal{P} \tag{2.13}$$

Toute autre fonction monotone décroissante dans le coût de qualité pourrait remplacer cette fonction linéaire. Notez que nous multiplions $cq_{\mathcal{P}}$ par $\left(\frac{u_{max}}{q_{max}}\right)$ afin de ramener le coût de qualité exprimé comme une fraction de q_{max} en une fraction de u_{max}. La distribution de $u(q_{\mathcal{P}})$ est :

$$u(q_{\mathcal{P}}) \sim N(\mu_{u(q_{\mathcal{P}})}, 0) \tag{2.14}$$

2.5.4 L'espérance d'utilité d'un plan

En faisant l'hypothèse que toutes les espérances d'utilités décrites précédemment sont indépendantes, nous pouvons les additionner pour obtenir l'espérance d'utilité du plan. Nous pondérons cette somme par des coefficients k_i nous permettant d'ajuster l'importance relative des différentes utilités selon les préférences du voyageur. L'espérance d'utilité d'un plan est donc :

$$\mu_{u(\mathcal{P})} = k_1 \cdot \mu_{u(c_{\mathcal{P}})} + k_2 \cdot \mu_{u(t_{\mathcal{P}})} + k_3 \cdot \mu_{u(q_{\mathcal{P}})} \tag{2.15}$$

où $\sum_{i=1}^{3} k_i = 1$.

2.5.5 La fonction de distribution de l'utilité d'un plan

Puisque les prix entrant dans la composition de la fonction d'utilité sont incertains, il est impossible de mesurer cette utilité avec certitude tant que les réalisations de ces variables aléatoires ne sont pas connues. Toutefois, comme l'incertitude sur toutes les variables est modélisée par des lois normales, il est relativement simple de dériver la distribution de l'utilité à partir des distributions des variables qui la composent. Ainsi, grâce aux propriétés des lois normales nous pouvons écrire l'utilité d'un plan comme la somme des fonctions de distribution des variables qui la composent. L'utilité d'un plan suit la distribution :

$$u(\mathcal{P}) \sim N(\mu_{u(\mathcal{P})}, \sigma^2_{u(\mathcal{P})}) \tag{2.16}$$

où

$$\sigma^2_{u(\mathcal{P})} = k_1^2 \sigma^2_{c_\mathcal{P}} \tag{2.17}$$

Un "plan parfait" en environnement certain respectant toutes les contraintes de temps, fournissant un maximum de qualité et ne coûtant rien, aurait une utilité $u_\mathcal{P} \sim N(u_{max}, 0)$.

2.5.6 Exemple d'évaluation de la distribution d'utilité d'un plan

Afin que la notion d'évaluation de la distribution d'utilité soit bien claire, voici un exemple du calcul de la distribution d'utilité d'un plan complet. Le tableau 2.5 reprend l'exemple du tableau 2.4 en y ajoutant des bornes pour les contraintes temporelles du voyageur.

Arrivée	Départ	Ville
0	2200 ± 48	v_7
2200 ± 48	2368 ± 48	v_9
2368 ± 48	*8736*	v_7

TAB. 2.5 – Exemple de description de voyage avec contraintes temporelles

Supposons que ce voyageur ait un budget de 2500$ et que la qualité maximale (q_{max}) des hôtels et des avions soit de 100. Si le voyageur est très soucieux de son budget ($k_1 = 0.6$), moyennement soucieux du respect de ses contraintes temporelles ($k_2 = 0.3$) et peu soucieux de la qualité de son voyage ($k_3 = 0.1$), quelle serait la distribution d'utilité du plan illustré par le tableau 2.6 correspondant à une solution à ce problème de planification ?

Opérateurs :	Effets :
$avion(v_7, v_9, 2214, 2217, 246\$, a_3)$	$etre(v_7, [0, 2214])$
$avion(v_9, v_7, 2375, 2378, 327\$, a_3)$	$etre(v_9, [2217, 2375])$
$hotel(v_9, 2217, 2375, 47\$/24hrs, h_1)$	$etre(v_7, [2378, 8736])$
	$loger(v_9, [2217, 2375])$

TAB. 2.6 – Plan complet pour le problème du tableau 2.5

Le plan décrit par le tableau 2.6 propose que le voyageur quitte v_4 en $t = 2214$ pour arriver à v_9 trois heures plus tard. Après 6.5 jours, le voyageur retourne à v_7. Le coût monétaire total de ce voyage est de $246+327+\frac{2375-2217}{24} = 882\$$. L'écart par rapport aux contraintes temporelles du voyageur est de $|2200 - 2214| + |2200 - 2217| + |2368 - 2375| + |2368 - 2378| = 48$ heures. Le coût de qualité moyen de ce plan sera calculé en prenant la moyenne du coût de qualité du premier avion ($100 - 90 = 10$), du second avion ($100 - 90 = 10$) et de l'hôtel ($100 - 10 = 90$), c'est-à-dire $\frac{10+10+90}{3} = 37$. Notez que les avions de la compagnie a_3 ont une qualité de 90 et que les hôtels de type h_1 ont une qualité de 10. Rappelons aussi que la qualité maximale q_{max} est de 100.

Fixons u_{max} à 1000 (toute autre valeur serait acceptable). L'espérance d'utilité associée au coût de ce plan (882$) est calculée par l'équation 2.7 qui nous donne $\mu_{u(c_\mathcal{P})} = 1000(1 - \frac{882}{2500}) = 647$. L'espérance d'utilité associée au respect des contraintes temporelles ($\mu_{u(t_\mathcal{P})} = 833$) est obtenue de l'équation 2.9 (ici le calcul est long, nous vous laissons appliquer la formule vous-même). Finalement, l'espérance d'utilité associée à la qualité ($\mu_{u(q_\mathcal{P})} = 1000 - 10 \times 37 = 630$) est obtenue de l'équation 2.13. La somme pondérée de ces trois utilités nous donne l'espérance d'utilité du plan : $\mu_{u(\mathcal{P})} = 0.6 \times 647 + 0.3 \times 833 + 0.1 \times 630 = 701$.

Pour calculer la variance de l'utilité du plan, il faut connaître la date à laquelle chacun de ses opérateurs a été entré dans la base de données. Cette date nous permet de calculer la variance associée au prix de chaque opérateur par l'équation 2.4. Nous appliquons ensuite l'équation 2.5 afin de connaître la distribution du coût monétaire du plan. Dans notre exemple, cette distribution suit une normale $N(882, 290)$. Ensuite, l'équation 2.8 nous donne la distribution de l'utilité liée au coût : $N(647, 46)$. Finalement, par l'équation 2.17, nous trouvons la variance de l'utilité du plan qui nous permet d'obtenir la distribution d'utilité : $N(701, 15)$.

2.6 Discussion sur le réalisme de l'environnement simulé et des modèles utilisés

Nos modèles correspondent-ils à la réalité ? Non, nous ne prétendons pas avoir simulé un environnement réaliste et nous ne prétendons pas non plus avoir fait une modélisation réaliste

de l'incertitude ou des fonctions d'utilité des voyageurs. Il apparaît évident que l'incertitude ne devrait pas se limiter aux prix. Une modélisation plus réaliste inclurait de l'incertitude sur la disponibilité de tous les opérateurs. Quant aux fonctions d'utilité, elles devraient inclure beaucoup plus de paramètres afin d'exprimer plus adéquatement les préférences des voyageurs. Ce raffinement des fonctions d'utilité va évidemment de pair avec un raffinement des opérateurs puisque chaque caractéristique d'un opérateur peut éventuellement devenir un paramètre de la fonction d'utilité.

Notre modèle fait l'hypothèse que le planificateur connaît les fonctions de distribution de l'incertitude dans l'environnement. Cette supposition est forte puisqu'il est difficile d'avoir une telle connaissance dans la réalité. Néanmoins, tous les planificateurs probabilistes que nous avons vus font aussi cette hypothèse [KHW95]. Nous n'avons donc pas de scrupule à faire de même. De plus, nous croyons qu'il serait possible de faire une estimation réaliste des fonctions de distribution par une analyse des données passées relatives aux variables pour lesquelles nous souhaitons modéliser de l'incertitude.

Nous croyons que cet exercice fournira un banc d'essai intéressant pour notre planificateur puisqu'il contient tout de même les éléments de base de l'environnement réel dans lequel nous voudrions l'utiliser, c'est-à-dire l'incertitude et une variation dans les préférences des voyageurs. De plus, l'utilisation d'un modèle d'incertitude réaliste et de fonctions d'utilité complexes pourrait se faire sans avoir à modifier le planificateur. C'est là l'intérêt de notre travail : nous prétendons fournir une façon de planifier dans un environnement incertain en fonction des préférences de l'agent, peu importe que l'incertitude ou les préférences soient parfaitement réalistes ou pas. Notre objectif est de montrer comment utiliser ce genre de modèle dans la planification et non de développer et d'évaluer un modèle. Nous estimons toutefois que si un tel travail était fait et qu'un modèle complexe était développé, il serait relativement simple d'utiliser notre stratégie de planification avec ce modèle.

Chapitre 3

L'algorithme de planification

Ce chapitre présente l'algorithme de planification que nous avons développé et testé dans le but de produire des plans de voyages évalués à l'aide d'une fonction d'utilité dans un environnement incertain. Comme nous le mentionnions plus tôt, un algorithme de planification est en fait un algorithme de recherche dirigée dans l'espace des plans. L'algorithme que nous proposons ici est basé sur les principes généraux de la planification en intelligence artificielle. La représentation du problème est faite avec le langage *STRIPS* et l'algorithme respecte la structure générale des planificateurs classiques tels que *POP* ou *SNLP*.

3.1 Modèle général de l'algorithme

Avant de donner les détails spécifiques à notre implantation, nous présenterons le modèle général de l'algorithme que nous avons utilisé. Nous exposerons les objectifs de notre planificateur, nous ferons quelques remarques sur la base de données utilisée et nous expliquerons finalement le fonctionnement général de notre algorithme.

3.1.1 Les objectifs

Le processus de planification a pour objectif de créer une ou plusieurs solutions réalisables à partir d'un plan initial minimal. La quasi-totalité des techniques et algorithmes présentés dans la revue de l'état de l'art pouvaient produire une seule solution à un problème, cette solution étant normalement "la meilleure solution possible". Dans une situation incertaine, il est tentant de vouloir simplement produire le plan ayant les meilleures chances de réussir ou, lorsque les plans sont évalués par une fonction d'utilité, de produire le plan permettant de maximiser l'espérance d'utilité. Or, nous ne pouvons jamais être certain que le plan ayant la moyenne d'utilité la plus élevée sera bel et bien celui offrant le meilleur niveau d'utilité. C'est seulement lors de la réalisation du plan, ou lors d'une étape d'observation préalable, que l'agent pourra évaluer l'utilité *certaine* d'un plan. Il se peut donc que le plan qu'on croyait être le meilleur s'avère très mauvais lorsque son utilité est connue avec certitude. Afin d'éviter qu'une telle situation n'oblige l'agent à faire une seconde planification, notre algorithme offre la possibilité de produire plusieurs plans alternatifs. Un usager peut toujours demander qu'on lui présente un plan supplémentaire afin qu'il puisse ensuite choisir celui qu'il préfère. L'objectif de notre planificateur n'est donc pas de produire *le* meilleur plan mais bien de produire *les* meilleurs plans correspondants au problème à résoudre.

Notre objectif principal est de produire des plans réalisables dans un délai raisonnable. Pour ce faire, nous devons diriger judicieusement la recherche dans l'espace des plans au moyen d'heuristiques basées sur les préférences de l'utilisateur.

3.1.2 La base de données

Le planificateur ne peut pas travailler sans avoir une connaissance minimale de l'environnement dans lequel il évolue. Nous faisons l'hypothèse que notre planificateur dispose d'une base de données contenant tous les opérateurs disponibles et qu'il connaît la distribution de probabilités associée à chacun. Dans le contexte de la planification de voyage, cela signifie que le planificateur ait accès à une liste de tous les billets d'avions et de toutes les chambres d'hôtel disponibles dans le monde.

3.1.3 Fonctionnement général de l'algorithme

Notre planificateur est un algorithme itératif faisant une recherche dans l'espace des plans. Son fonctionnement est très similaire aux algorithmes de planification logiques classiques. Il s'agit d'une suite d'itérations permettant de passer d'un plan minimal à un plan complet en raffinant *des* solutions partielles. Nous mettons l'accent sur la multiplicité des solutions partielles parce qu'il s'agit d'une différence importante avec les algorithmes logiques qui raffinent toujours une seule solution. Comme nous le disions au chapitre 1, ils peuvent procéder ainsi parce qu'ils travaillent avec des opérateurs partiellement instanciés. L'instanciation progressive de ces opérateurs leur permet de ne maintenir qu'une seule solution en mémoire.

Notre problème est particulier car nous n'avons pas accès à des opérateurs pouvant être partiellement instanciés. Les valeurs que peuvent prendre les variables des opérateurs ($avion(\cdot)$ et $hotel(\cdot)$) sont limitées par les billets d'avion et les chambres d'hôtel disponibles. Nous ne pouvons donc pas prendre pour acquis que tous les vols d'avion imaginables existent, ni que tous les hôtels imaginables soient disponibles. Ceci diffère grandement du monde des blocs où il n'y a aucune contrainte sur la façon dont les opérateurs peuvent être instanciés. Cette caractéristique de notre problème nous oblige à faire une recherche explicite parmi tous les opérateurs possibles et à garder en mémoire plusieurs solutions partielles. Regardons le squelette de notre algorithme :

1. Vérifier si le plan est complet avant de poursuivre.
2. Choisir le plan à raffiner (le meilleur plan partiel).
3. Choisir l'objectif à atteindre parmi les conditions à satisfaire de ce meilleur plan.
4. Vérifier s'il est possible de satisfaire l'objectif à partir des effets déjà présents dans le plan. Si oui, faire le lien avec cet effet et passer à l'étape 7.
5. Choisir, selon la fonction d'utilité, le meilleur opérateur permettant de satisfaire l'objectif.
6. Créer une nouvelle instance de plan partiel en copiant le plan à raffiner et en ajoutant l'opérateur choisi en 5 à cette copie.
7. Vérifier si le plan est complet. Si c'est le cas, le transférer de la liste des plans partiels vers

l'ensemble des plans complets. Sinon, l'insérer dans la liste des plans partiels. Retourner à l'étape 1.

Le choix du meilleur plan partiel (étape 2)

Choisir le meilleur plan partiel parmi tous ceux en mémoire est une étape très importante car c'est là que la branche à explorer est choisie. Il est toujours préférable d'explorer dans une direction prometteuse afin d'éviter de gaspiller des ressources sur une solution que nous savons être potentiellement mauvaise. Le problème est de déterminer la métrique qui nous permettra de décider quelle solution partielle est la meilleure et devrait donc être explorée.

C'est ici que la fonction d'utilité du voyageur entre en jeu. De par sa nature, elle nous permet de comparer facilement deux solutions alternatives. Toutefois, comme nous travaillons dans un environnement incertain, nous ne pouvons pas calculer une valeur d'utilité certaine pour les plans partiels. Nous avons plutôt accès à une distribution d'utilité. Dans notre modèle, toutes ces distributions suivent une loi normale, ce qui simplifie la tâche de comparaison. Nous pouvons simplement appliquer les règles de la dominance stochastique afin de déterminer si un plan partiel en domine un autre. Nous faisons d'abord une comparaison basée sur la dominance stochastique de premier ordre (comparaison des moyennes des distributions) et, en cas d'égalité*, nous appliquons la dominance stochastique de second ordre (comparaison des variances des distributions). Pour une révision des notions de dominance stochastique, nous vous référons au chapitre 6 de Mas-Colell, Whinston et Green [MCWG95].

Encore faut-il pouvoir évaluer les distributions d'utilité d'un plan partiel. Au chapitre 2, nous donnions un exemple de calcul de la distribution d'utilité d'un plan complet, mais qu'en est-il d'un plan partiel ? Il s'agit là d'un problème important dont la solution mérite une explication détaillée. Pour le moment, prenons pour acquis que nous savons faire ce calcul. Nous donnerons plus tard les explications qui s'imposent.

*En réalité, nous devrions plutôt parler de quasi-égalité. Dans notre implantation, nous considérons deux espérances d'utilité égales lorsqu'elles ont une différence inférieure à 2%.

Choix de l'objectif à atteindre (étape 3)

Une fois que le plan à raffiner a été choisi, il faut déterminer lequel de ses objectifs devrait être satisfait. Deux heuristiques guident ce choix, la première ayant préséance sur la seconde :

1. Toutes les conditions $etre(\cdot)$ doivent être satisfaites avant qu'on tente de satisfaire les conditions $loger(\cdot)$.
2. Les conditions doivent toujours être satisfaites en ordre chronologique.

La première règle est justifiée dans la mesure où les avions sont plus rares que les hôtels. Si nous supposons qu'il y a de très fortes chances pour qu'il existe des chambres d'hôtel correspondant à n'importe quel ensemble de billets d'avion cohérent, alors il est nécessairement préférable de commencer par explorer les opérateurs $avion(\cdot)$ disponibles. Le contraire obligerait le planificateur à considérer un grand nombre de chambres d'hôtel dont la disponibilité ne correspondrait à aucun billet d'avion. Or, nos opérateurs ont été générés de telle sorte que la rareté soit effectivement plus grande pour les avions que pour les hôtels. Cette façon de faire s'apparente à la décomposition hiérarchique dont nous discutions au chapitre 1. Nous décomposons le problème de planifier un voyage composé de billets d'avion et de réservations d'hôtels en deux sous-problèmes : le premier consistant à trouver un ensemble de billets d'avion et le second consistant à trouver des chambres d'hôtel correspondant à ces billets. Malgré cette décomposition apparente, nous tenons à préciser que le plan est construit comme un tout et non par morceaux. Ceci est indispensable car notre objectif est de trouver le meilleur plan en fonction des préférences du voyageur et ce meilleur plan n'est pas nécessairement une union des meilleurs plans partiels. La possibilité d'une incompatibilité temporelle entre ceux-ci peut faire en sorte qu'une telle union soit impossible. Par exemple, le meilleur billet d'avion permettant de faire l'aller entre deux villes pourrait être disponible après le meilleur billet de retour entre ces deux mêmes villes !

La seconde règle simplifie la gestion des bornes temporelles des effets d'un plan partiel. Une fois qu'un premier opérateur est trouvé, ses propres contraintes de temps limitent l'ensemble des prochains opérateurs pouvant être ajoutés. Ainsi, ajouter les opérateurs en ordre chronologique permet de profiter de cette restriction sur l'espace des possibilités.

La satisfaction d'une condition (étapes 4 et 5)

Il y a deux manières d'atteindre un objectif : en ajoutant un nouvel opérateur dont l'un des effets correspond à cet objectif ou en liant directement l'objectif à atteindre avec un effet déjà présent dans le plan. Afin d'éviter d'ajouter des opérateurs inutilement et possiblement de ne jamais arriver à compléter le plan, il est essentiel de toujours prioriser les effets déjà présents dans le plan et d'avoir recours à un nouvel opérateur seulement lorsqu'aucun effet du plan ne permet de satisfaire l'objectif.

La recherche d'un nouvel opérateur peut être un problème en soi lorsque leur nombre est très grand. Pour un objectif n'ayant jamais fait l'objet d'une recherche des opérateurs permettant de le satisfaire, nous faisons une recherche dans la base de données afin de trouver tous les opérateurs pertinents. Nous les classons ensuite en ordre croissant de "qualité". Lorsqu'un objectif a déjà fait l'objet d'une telle recherche, nous prenons simplement l'opérateur suivant dans la liste des opérateurs possibles.

Le classement des opérateurs selon leur "qualité" étant fortement lié au classement des plans dont nous discutions plus tôt, prenons simplement pour acquis que nous savons le faire. Nous reviendrons plus tard sur la manière de procéder.

Vérifier la complétude d'un plan (étape 7)

Chaque fois qu'une modification est faite à un plan partiel, le planificateur vérifie la complétude du plan. S'il est complet, il sera transféré de la liste des plans en cours de développement vers la liste des plans complets. Il n'y a qu'un seul critère permettant de déterminer si un plan est complet : toutes ses conditions doivent être satisfaites. La procédure que nous employons pour construire les plans garantit que *tout plan complet soit cohérent*. Conséquemment, à partir du moment où toutes les conditions d'un plan sont atteintes, nous pouvons considérer ce dernier comme une solution possible à la requête de planification.

3.2 Évaluation des plans partiels et des opérateurs

Dans la section précédente, l'évaluation des plans partiels et des opérateurs était prise pour acquise. Nous expliquerons maintenant comment nous nous servons de la fonction d'utilité du voyageur afin de faire ces évaluations.

3.2.1 La planification A^*

Au premier chapitre, nous avons discuté des avantages d'utiliser le principe d'une recherche A^* lorsque l'espace de recherche est très grand et qu'une décomposition hiérarchique ne permet pas de le réduire. Nous croyons que notre problème correspond parfaitement à cette situation. De plus, il est rare qu'un voyage comporte plus de quelques billets d'avion et quelques hôtels. Les solutions sont donc relativement petites ce qui limite la complexité de A^*. Nous croyons par conséquent que A^* représente un très bon choix d'algorithme de recherche pour les besoins de notre application. Cette hypothèse a d'ailleurs été confirmée par nos tests comme vous le verrez au chapitre 4.

Le principe de A^* consiste à utiliser une estimation du coût des solutions partielles afin de les comparer entre elles. Pour assurer que l'algorithme trouve toujours la meilleure solution, l'estimation ne doit jamais dépasser la valeur réelle de ce coût. C'est avec de telles estimations que nous avons comparé les plans partiels à l'étape 2 de l'algorithme, lorsque le meilleur plan partiel doit être sélectionné. Nous calculons une sous-estimation du coût espéré des plans partiels et choisissons celui dont le coût estimé est le plus faible. Afin que l'évaluation tienne compte des préférences du voyageur, nous utilisons le coût d'utilité qui, par définition, capture l'ensemble de ces préférences.

3.2.2 La sous-estimation du coût d'utilité espéré d'un plan partiel

Le coût d'utilité espéré correspond à la perte d'utilité totale d'un plan par rapport à l'utilité maximale pouvant être atteinte par un plan quelconque. Dans notre modèle (voir le chapitre 2), cette valeur prend la forme de la constante u_{max}. Comme l'ajout d'un opérateur implique

nécessairement un coût monétaire, une perte possible en qualité par rapport à la qualité maximale ainsi que la possibilité que certaines contraintes temporelles ne soient pas respectées, il est clair que tout nouvel opérateur représente toujours une perte en utilité. Le meilleur plan est par conséquent celui qui minimise le coût d'utilité espéré.

Pour reprendre la notation habituelle de A^*, nous dirons qu'un plan partiel a un coût d'utilité connu découlant des effets des opérateurs qui le compose $(g(n))$ et un coût d'utilité estimé $(h(n))$ correspondant au coût minimal nécessaire pour que le plan partiel devienne un plan complet. Cette sous-estimation devrait, au plus, être égale à la valeur du meilleur plan possible. Évidemment, s'il était facile de connaître le coût d'utilité du meilleur plan, il serait tout aussi facile de le trouver. Or, ce n'est absolument pas le cas. En fait, il est relativement facile de trouver le meilleur billet d'avion pour une liaison aérienne quelconque ou le meilleur hôtel dans une ville. La difficulté vient de la disponibilité temporelle restreinte des opérateurs. Par contre, nous pouvons assez facilement calculer une sous-estimation raisonnable du coût d'utilité espéré en excluant les contraintes temporelles de l'équation. Autrement dit, nous pouvons facilement trouver les meilleurs opérateurs en terme de prix et de qualité : il suffit de mesurer, pour une fonction d'utilité donnée, le coût d'utilité engendré par un opérateur en ne considérant que son prix et sa qualité. Les équations 2.5 et 2.6 nous permettent d'exprimer le coût d'utilité espéré occasionné par le prix comme :

$$\mu_{C(u_c)} = k_1 \left[prix \times \frac{u_{max}}{budget} \right] \tag{3.1}$$

et que le coût d'utilité découlant de la qualité comme :

$$\mu_{C(u_q)} = k_3 \left[q_{max} - qualite \right] \left[\frac{u_{max}}{q_{max}} \right] \tag{3.2}$$

La figure 3.1 reprend l'exemple de plan partiel du chapitre 2. Rappelons que le voyageur a un budget de 2500$, que la qualité maximale (q_{max}) des hôtels et des avions est de 100 et que les k_i sont : $(k_1 = 0.6)$, $(k_2 = 0.3)$ et $(k_3 = 0.1)$. Le tableau 3.1 rappelle les contraintes temporelles du voyageur.

Arrivée	Départ	Ville
0	2200 ± 48	v_7
2200 ± 48	2368 ± 48	v_9
2368 ± 48	*8736*	v_7

TAB. 3.1 – Contraintes temporelles du voyageur correspondant au plan de la figure 3.1

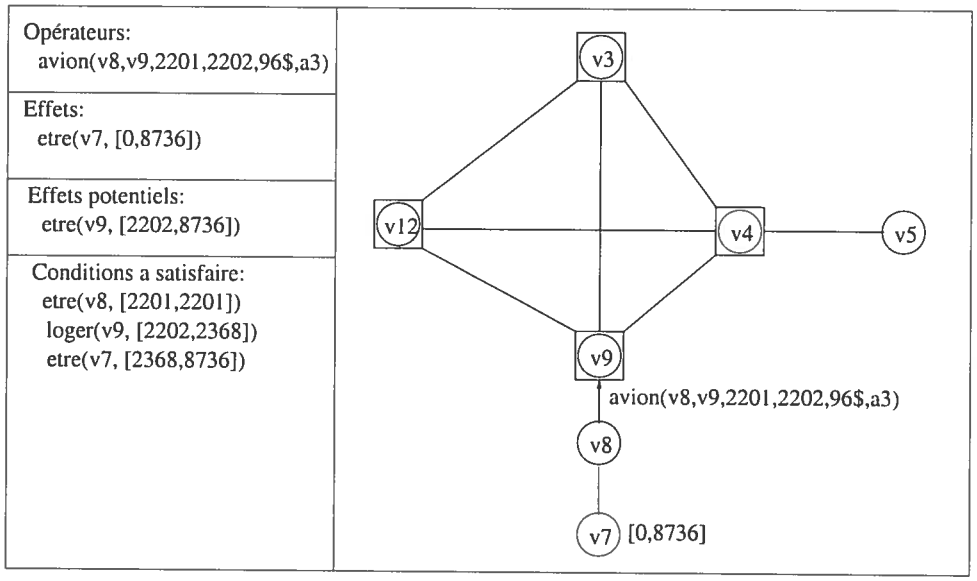

FIG. 3.1 – Plan partiel à évaluer

Quel serait le coût d'utilité estimé de ce plan ? La partie connue du coût ($g(n)$) correspond aux coûts engendrés par l'application de l'opérateur $avion(v_8, v_9, 2201, 2202, 96\$, a_3)$. Celui-ci occasionne une perte d'utilité de 38 en raison de son prix (96×0.4) et de 100 parce que sa qualité est de 10 points inférieure à 100 (q_{max}). Le fait qu'il permette d'arriver à v_9 avec 2 heures de retard par rapport au moment souhaité occasionne une perte d'utilité de 7. Au total, la partie connue du coût d'utilité est donc de 55.

Il reste maintenant à calculer la sous-estimation du coût d'utilité occasionné par les effets des opérateurs manquant à ce plan. La première question à se poser est donc précisément, que manque-t-il à ce plan ? Il manque des billets d'avion permettant de faire les liaisons v_7 à v_8 et v_9 à v_7 ainsi qu'une chambre d'hôtel pour le séjour en v_9.

L'estimation du coût d'utilité nécessaire au déplacement entre une ville v_a et une ville v_b doit avoir été faite lors d'une étape préalable à la planification. Par l'algorithme de Floyd [Flo62], nous construisons une matrice $n \times n$ (n, le nombre de villes) contenant le meilleur coût d'utilité possible pour voyager entre chaque paire de villes dans le monde. Ces coûts d'utilité minimaux sont calculés à partir de tous les billets d'avion disponibles durant la fenêtre de temps du voyageur, sans égard aux contraintes temporelles. Puisqu'ils ne tiennent pas compte des contraintes temporelles, il est absolument certain que tous les coûts d'utilité de la matrice sont sous-estimés. De plus, dans le cas d'un voyage comprenant plusieurs billets d'avions[†], il est fort probable que les avions utilisés dans le calcul du coût d'utilité ne soient même pas compatibles. Dans notre exemple, nous pouvons utiliser cette matrice des coûts d'utilité minimaux afin d'obtenir l'estimation de la perte d'utilité engendrée par les billets d'avion qui devront être ajoutés au plan. Notez que l'algorithme de Floyd produit cette matrice en temps d'ordre $O(n^3)$.

L'estimation du coût d'utilité est plus simple pour les hôtels : il suffit de trouver le meilleur hôtel dans chaque ville sans égard aux contraintes temporelles. Le planificateur mémorise donc le meilleur hôtel (en terme de prix et de qualité) disponible dans chaque ville durant la fenêtre de temps du voyage à planifier. Il est fort probable que le meilleur hôtel ne soit pas compatible avec les avions en raison des contraintes de temps, mais cela n'a pas d'importance puisque nous cherchons simplement à avoir une sous-estimation raisonnable du coût d'utilité espéré.

Bref, le coût d'utilité d'un plan partiel se divise en deux termes : $g(n)$ et $h(n)$. Le premier se calcule directement et le second est obtenu à partir de données calculées lors d'une étape préalable à la planification. Comme nos estimés ne surestiment jamais les vrais coûts, nous sommes certains de toujours trouver le meilleur plan répondant aux préférences du voyageur.

3.2.3 Le choix d'un opérateur

À l'étape 5 de l'algorithme de planification, nous devons choisir le meilleur opérateur permettant de satisfaire un objectif. Vous ne serez pas surpris d'apprendre que ce meilleur opérateur est celui occasionnant les effets dont les coûts d'utilité sont les plus faibles. La

[†]En fait, tous les voyages produits par notre application comportent au moins deux billets d'avion : l'aller et le retour.

procédure de comparaison des opérateurs est très simple : nous calculons le coût d'utilité marginal pour chaque opérateur possible avec les équations 3.1 et 3.2, sans égard aux contraintes temporelles. Le meilleur opérateur sera celui ayant le coût le plus faible.

3.3 Exemple complet de planification d'un voyage simple

Cette section décrit le cheminement complet de la planification d'un voyage simple. Ce voyage consiste à passer environ une semaine dans une capitale autre que la capitale de départ. Notre objectif sera de trouver le meilleur plan pour ce voyage en fonction des préférences du voyageur. Le plan devrait donc contenir deux billets d'avion (aller et retour) et une chambre d'hôtel. Le tableau 3.2 décrit précisément le voyage à planifier. L'arrivée dans la première ville au temps 2000 signifie que le voyageur se trouvera dans la ville de départ avant le début du voyage. La planification ayant lieu au temps 2000, l'endroit où se trouve le voyageur avant cette date est sans importance. Le départ de la dernière ville au temps 8736 signifie que le voyageur restera dans la ville finale jusqu'à la fin de l'horizon connu du planificateur (une année).

Arrivée	Départ	Ville
2000	2200 ± 48	v_4
2200 ± 48	2368 ± 48	v_9
2368 ± 48	*8736*	v_4

TAB. 3.2 – Description du voyage à planifier

Le voyageur dispose d'un budget 5000$ et accorde une importance égale au coût, à la qualité et au respect des contraintes temporelles. Les taux de substitution de la fonction d'utilité seront donc : $k_1 = 0.33$, $k_2 = 0.33$ et $k_3 = 0.33$. Notez que la qualité maximale d'un hôtel et d'un avion sont de 100 et que l'utilité maximale (utilité d'un plan parfait) est de 1000.

La construction d'un plan minimal est préalable au lancement de l'algorithme de planification. Le plan minimal est composé du prédicat *etre* représentant l'état initial (où se trouve le voyageur au début de la planification) et de l'ensemble des conditions à satisfaire pour avoir

l'état final. Le plan initial est donc un plan dont la séquence d'opérateurs permettant de passer de l'état initial à l'état final est vide. Le plan minimal pour notre problème ainsi qu'une illustration des effets de ce plan apparaissent sur la figure 3.2.

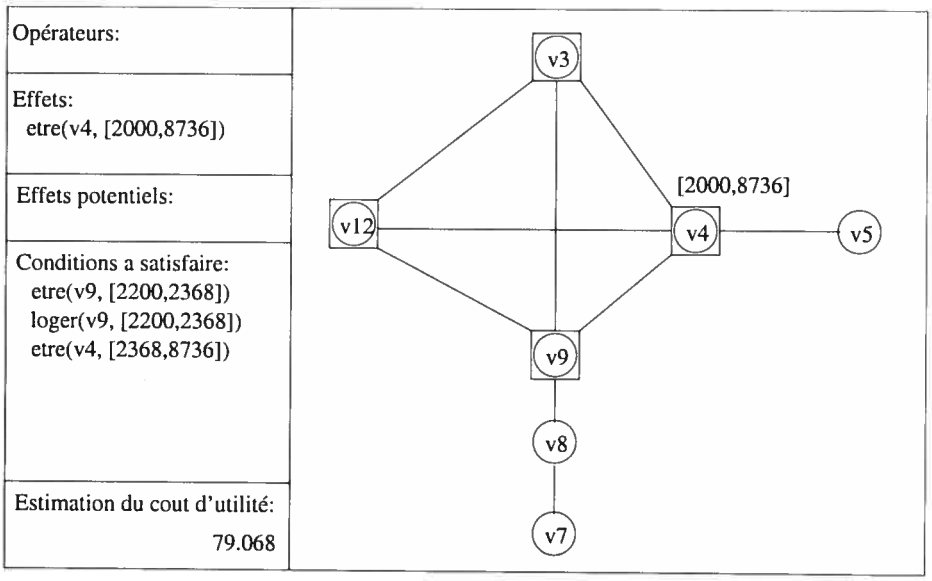

FIG. 3.2 – Plan initial et ses effets (plan #0)

Rappelons que l'estimation du coût d'utilité espéré est faite à partir de données calculées avant le début de la planification. Dans notre exemple, cet estimé représente la somme du meilleur avion assurant la liaison $v_4 - v_9$, du meilleur avion assurant la liaison $v_9 - v_4$ et du meilleur hôtel pour une semaine en v_9. Tous ces "meilleurs opérateurs" sont issus d'une comparaison entre tous les opérateurs dans l'intervalle $[2152, 2416]$ sans égard à leurs contraintes temporelles.

Itération 1

- Liste des plans partiels : plan#0
- Meilleur plan : plan#0
- Condition à remplir : $etre(v_9, [2200, 2368])$
- Opérateur ajouté : $avion(v_8, v_9, 2201, 2202, 96\$, a_3)$

Le choix de cet avion peut sembler bizarre compte tenu du fait qu'il part de v_8 et que le

voyageur se trouve en v_4. Mais l'objectif du planificateur est d'essayer *le meilleur opérateur compte tenu de la fonction d'utilité du voyageur* lui permettant d'arriver à v_9 à l'intérieur de ses contraintes temporelles. Or, il s'avère que parmi les 15 vols en provenance de quatre différentes villes atterrissant à v_9 dans l'intervalle $[2152, 2248]$, le meilleur décolle de v_8.

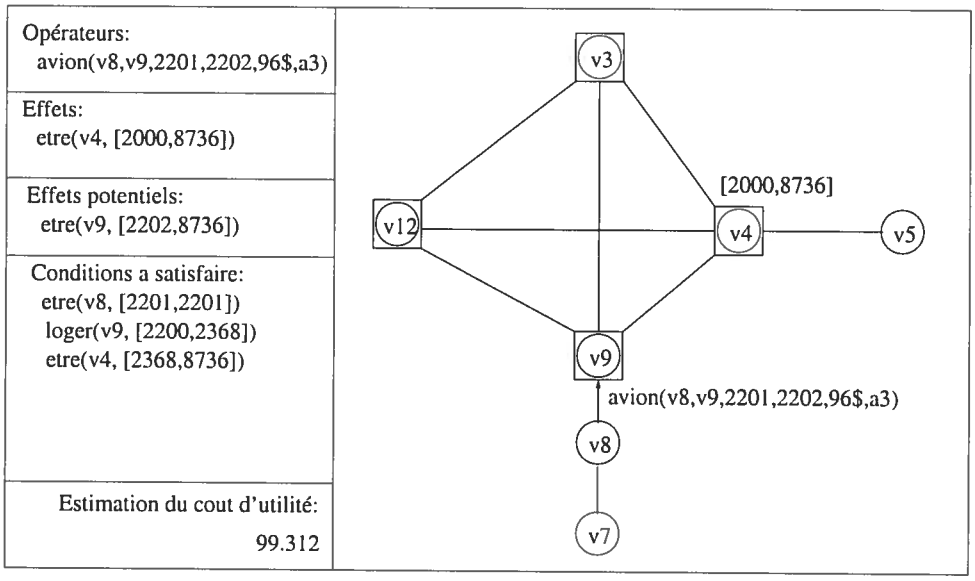

FIG. 3.3 – Plan #1 et ses effets sur l'état du monde

Remarquez que l'effet de l'opérateur qui a été ajouté s'est placé dans la liste des effets potentiels. Comme nous le disions au chapitre 2, tous les opérateurs qui ne sont pas applicables immédiatement en raison de pré-conditions qui ne sont pas remplies sont temporairement placés dans cette liste. Dans notre cas, le billet d'avion ne peut être appliqué parce que le voyageur doit d'abord se trouver en v_8. C'est aussi la raison pour laquelle l'intervalle de temps associé à v_4 sur la figure 3.3 est le même que sur la figure 3.2.

Itérations 2 à 16

Nos estimations de coût étant systématiquement inférieures aux coûts d'utilité réels, il n'est pas étonnant que le plan partiel #1 ait un coût d'utilité plus élevé que celui du plan minimal. En fait, tous les opérateurs permettant d'atteindre la première condition à remplir seront explorés parce que le plan minimal sera systématiquement meilleur que tous les plans partiels issus de

son raffinement. Dans notre exemple, comme il y a 15 avions qui permettent d'arriver à v_9 en temps voulu, 15 plans partiels seront créés et le meilleur de ces 15 plans sera ensuite raffiné. Les itérations 2 à 16 serviront à la construction de ces plans. Sautons donc à l'itération 17 où le meilleur des 15 plans partiels en mémoire sera raffiné.

Itération 17

- Liste des plans partiels : $[\#1, 99]$, $[\#2, 104]$, $[\#3, 108]$, $[\#9, 114]$, $[\#4, 118]$, ..., chaque paire $[a, b]$ représentant le numéro du plan (a) et son utilité estimée (b).
- Meilleur plan : plan#1
- Condition à remplir : $etre(v_8, [2201, 2201])$
- Opérateur ajouté : $avion(v_7, v_8, 2197, 2198, 94\$, a_3)$

Avant de discuter du raffinement du plan #1, regardons la liste des plans partiels. Comme nous ajoutons toujours le meilleur opérateur, il peut paraître surprenant de voir que, par exemple, le plan #9 est meilleur que le plan #4. Pourtant, l'explication est très simple : le fait d'ajouter le meilleur opérateur garantit que le coût d'utilité de cet opérateur $(g(n))$ sera minimal, mais rien n'empêche le coût d'utilité estimé du plan $(h(n))$ d'être très élevé.

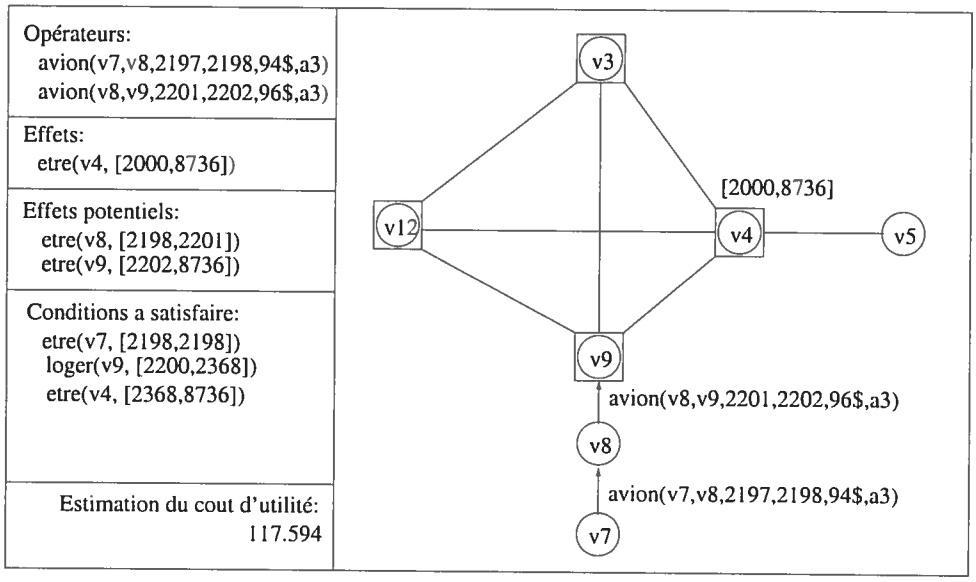

FIG. 3.4 – Plan #16 et ses effets sur l'état du monde

Le plan partiel produit par l'ajout d'un second avion est plutôt mauvais (figure 3.4), son coût d'utilité espéré estimé étant de 117.594 ce qui le place en cinquième position de la liste des plans partiels. Comme cet avion partant de v_7 était le seul moyen d'arriver à v_8 à temps pour prendre le second avion sans arriver trop tôt, l'exploration du plan #1 s'arrêtera ici, se soldant par un échec.

Itérations 18 à 25

Après l'échec de l'exploration du plan #1, la recherche se poursuit avec le plan #2 (devenu le meilleur) pour se solder encore une fois par un échec. En fait, c'est l'exploration du plan #9 qui finira par donner de bons résultats. Rappelons que ce plan est issu du raffinement du plan initial. La figure 3.5 illustre le plan #9 et ses effets.

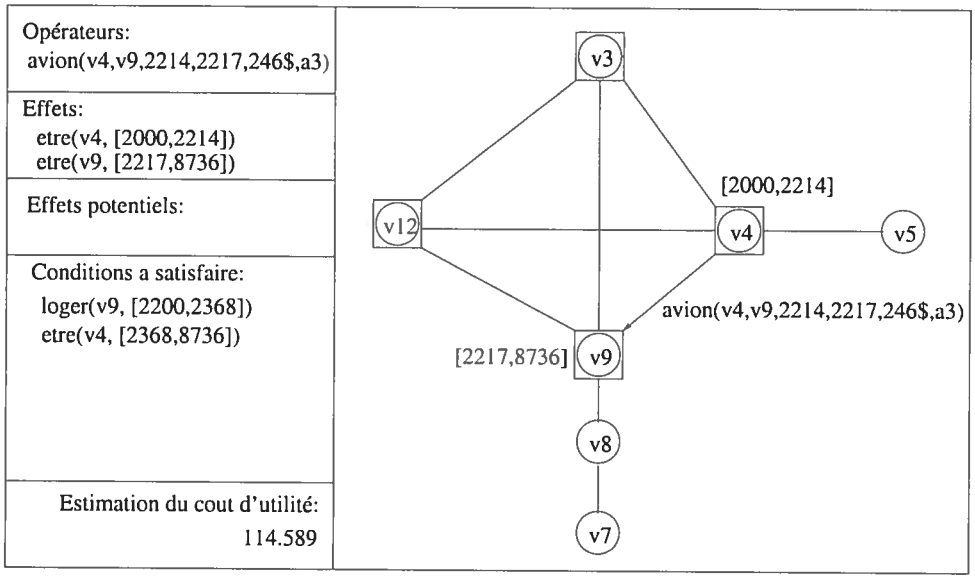

FIG. 3.5 – Plan #9 et ses effets sur l'état du monde

Remarquez que l'effet du billet d'avion s'est placé dans la liste des effets et non dans la liste des effets potentiels parce que, partant de v_4, il est applicable immédiatement. Il faut maintenant trouver les opérateurs permettant de revenir en v_4 environ une semaine plus tard. Rappelons que notre algorithme commence par trouver les opérateurs pour tous les déplacements avant de trouver les chambres d'hôtel. Cette recherche du billet de retour n'a pas trouvé la meilleure

solution du premier coup. Sur les 16 avions possibles permettant d'arriver à v_4 dans les temps, c'est la cinquième tentative qui aura produit le meilleur plan partiel. Sautons donc à l'itération 26 où ce nouveau plan a été trouvé.

Itération 26

- Liste des plans partiels : #9,#16, #4, #17, #5,... (18 en tout)
- Meilleur plan : plan#9
- Condition à remplir : $etre(v_4, [2368, 8736])$
- Opérateur ajouté : $avion(v_9, v_4, 2375, 2378, 327\$, a_3)$

Le nouveau plan est illustré par la figure 3.6.

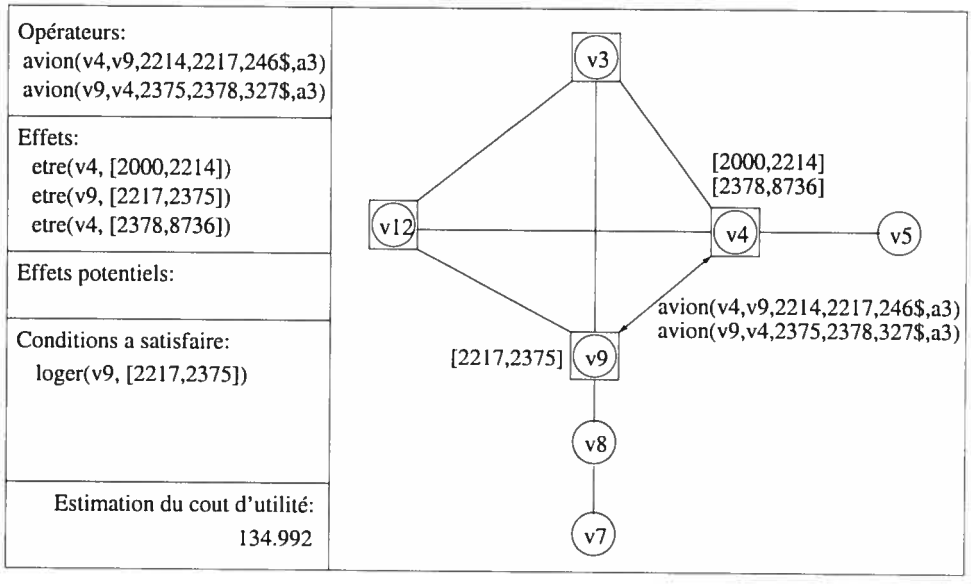

FIG. 3.6 – Plan #22 et ses effets sur l'état du monde

Itérations 27 à 45

Le plan #22 n'est pas ressorti comme étant le meilleur immédiatement. Il a été classé en septième position dans la liste des plans partiels. Toutefois, l'exploration des solutions potentiellement meilleures que le plan #22 s'est toujours soldée par un échec. Sautons à l'itération

46 où le plan #22 est finalement devenu la meilleure solution potentielle.

Itération 46

- Liste des plans partiels : #22,...
- Meilleur plan : plan#22
- Condition à remplir : $loger(v_9, [2217, 2375])$
- Opérateur ajouté : $hotel(v_9, 2217, 2375, 47\$/24hrs, h_1)$

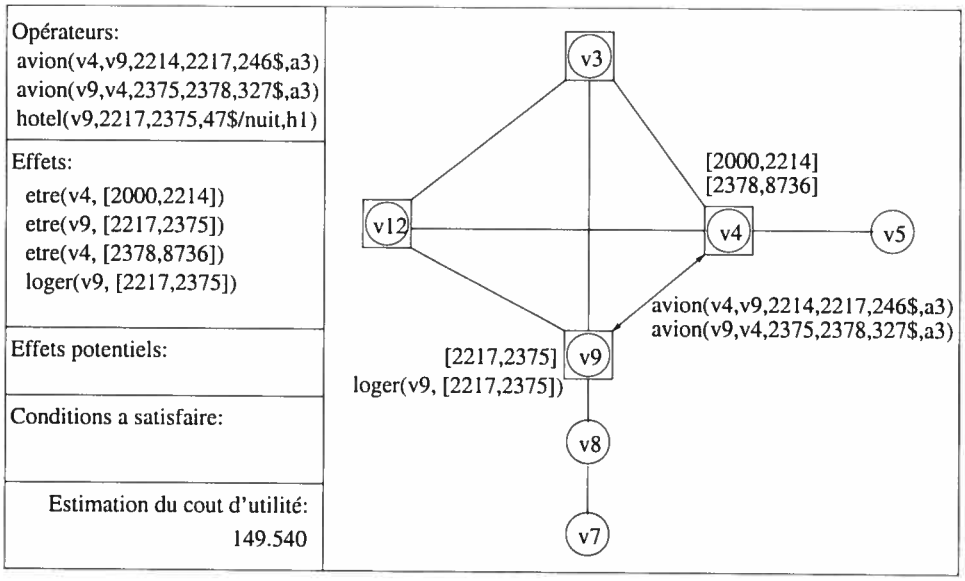

FIG. 3.7 – Plan #37 (complet et optimal) et ses effets sur l'état du monde

Trouver un plan complet *ne garantit pas* que ce plan soit optimal. Nous saurons qu'il est optimal seulement lorsqu'aucun autre plan partiel n'aura un estimé de coût d'utilité inférieur au coût d'utilité du plan complet. Dans notre exemple, 8 plans partiels sur les 28 en mémoire avaient un espoir d'être meilleurs que le plan #37. Toutes ces branches de solution ont donc été partiellement ou complètement explorées afin d'être certain qu'aucune ne permette d'atteindre un coût d'utilité espéré plus faible que celui du plan #37. C'est à l'itération 84 que le plan #37 a été déclaré meilleure solution.

Le fait d'avoir 38 itérations après qu'un plan complet ait été trouvé (en seulement 46 itérations) peut sembler long. Ces 38 itérations servent à s'assurer qu'un hôtel inaccessible avec

les billets d'avion du plan complet ne permettent de diminuer le coût d'utilité. En débutant la recherche avec les billets d'avion, nous contraignons les chambres d'hôtels disponibles et il faut s'assurer que cette limitation ne nous empêche pas de trouver le meilleur plan. Si, dans nos estimations du coût d'utilité, nous utilisions seulement les hôtels accessibles en fonction des avions choisis plutôt que le meilleur hôtel sans égard aux contraintes temporelles, nous pourrions éviter ces itérations supplémentaires. Cela aurait par contre un autre coût : il faudrait comparer tous les hôtels disponibles en fonction des avions choisis à chaque évaluation d'un plan partiel. Nous avons jugé ce coût plus élevé que les quelques itérations supplémentaires rendues nécessaires par une estimation moins juste du coût d'utilité des chambres d'hôtel.

3.4 Discussion

L'algorithme que nous avons décrit dans ce chapitre n'a pas été développé d'un seul coup. Quelques tentatives l'ont précédé, nous permettant chaque fois de tirer des leçons de nos erreurs, principalement au niveau de la stratégie de planification.

Dès le début, nous souhaitions travailler sur le modèle général d'algorithme présenté en 3.1. Nous n'utilisions toutefois pas la même approche de recherche. Plutôt que de faire une recherche A^*, nous utilisions une approche gloutonne ne tenant pas compte des estimés de coût des plans partiels. De plus, nous ne dirigions pas la recherche avec les préférences du voyageur. Nous avions plutôt des heuristiques communes à toutes les requêtes permettant de limiter l'espace de recherche. Les résultats nous semblaient satisfaisants, mais notre incapacité à savoir si nous trouvions les meilleures solutions rendait l'évaluation difficile. En fait, tout ce que nous pouvions faire était de demander au planificateur de trouver plusieurs milliers de plans afin de vérifier jusqu'à quel point il arrivait à trouver de meilleures solutions.

En comparant la recherche A^* dirigée par le coût d'utilité espéré, nous avons constaté que notre ancienne approche fonctionnait bien pour des problèmes très simples nécessitant peu d'opérateurs (deux avions et un hôtel par exemple). Par contre, dès que le problème devenait un peu plus complexe, le comportement de notre stratégie initiale s'avérait plutôt imprévisible. Dans certaines situations, il arrivait que le planificateur trouve le meilleur plan, mais dans

d'autre cas, la meilleure solution n'apparaissait même pas dans les 10000 premiers plans trouvés. Nous croyons par conséquent que l'utilisation d'estimés sur les plans partiels a fourni une valeur ajoutée importante à notre stratégie en lui permettant de choisir plus adéquatement les plans partiels à raffiner.

De plus, le choix d'une heuristique directement basée sur les préférences du voyageur permet de réellement trouver les meilleurs plans en fonction de notre critère de comparaison. Nous avons constaté qu'il est très difficile de construire des heuristiques communes à toutes les fonctions d'utilité sans exclure des solutions intéressantes. Il arrive que les plans qui s'avèrent être les meilleurs soient légèrement étranges et une heuristique générale, même si elle fonctionne la plupart du temps, a tendance à éliminer ces pistes de solution.

Chapitre 4

Analyse des résultats

Ce chapitre présente les tests que nous avons utilisés afin d'évaluer notre implantation de l'algorithme présenté au chapitre précédent. La première section définit les objectifs des tests et l'environnement utilisé, la seconde section discute de la base de données utilisée et la troisième section présente une analyse des résultats obtenus.

4.1 Les objectifs de l'évaluation

Notre planificateur étant conçu pour résoudre un type de problème très précis, nous ne pouvons comparer directement sa performance avec d'autres planificateurs génériques comme *Graphplan* [BF97] par exemple. Nous essaierons donc simplement de tracer un portrait le plus fidèle possible des forces et faiblesses de notre implantation.

Nous évaluerons d'abord la sensibilité des solutions à des variations dans la fonction d'utilité afin de mesurer la souplesse d'adaptation des plans à de petits changements des préférences. Ensuite, nous analyserons l'erreur d'estimation du coût d'utilité $(h(n) - h^*(n))$ en fonction de la taille de la solution afin d'avoir une intuition sur la complexité de notre algorithme. Puis, nous discuterons de la consommation de temps et de mémoire de notre implantation en fonction de la taille de la solution et du nombre de plans partiels créés. Nous évaluerons finalement le rapport entre la qualité de solutions supplémentaires (après avoir obtenu la meilleure) et le

temps nécessaire pour les trouver.

Nous ne ferons aucune analyse de l'utilité espérée des plans produits. Trois raisons motivent cette décision. D'abord, nous savons déjà que notre planificateur trouve toujours la meilleure solution, il est donc inutile de faire des tests relatifs à la qualité des plans ou à leur stabilité. De plus, rappelons que la qualité des solutions dépend de la qualité et de la diversité des opérateurs dans la base de données. Finalement, notez que les variations du niveau d'utilité atteint par différentes solutions dépendent directement des choix que nous avons faits lors de la modélisation des fonctions d'utilité. L'objectif de nos tests est d'évaluer la qualité de notre algorithme de planification et non la qualité de la simulation du monde ou de la modélisation des fonctions d'utilité.

Tous les tests ont été faits sur un Pentium III 733MHz disposant de 528Mo de mémoire vive. Le programme *Java* a été exécuté sur *Java HotSpot(TM) Client VM (build 1.4.1_02-b06, mixed mode)* dans l'environnement *Linux* de *RedHat 7.3*. La base de données était une base de données *PostgreSQL 7.0.3*.

4.2 La base de données

L'environnement simulé sur lequel nos tests ont été faits correspond au monde décrit au chapitre 2. La base de données a donc été générée à partir de variables aléatoires et il va sans dire que son contenu aura une influence directe sur les plans produits et la difficulté avec laquelle ils seront construits.

La base de données compte 14 144 entrées pour les avions et 4 571 entrées pour les hôtels. Comme notre monde compte 12 villes et que la simulation a été faite sur l'intervalle de temps [1 000, 8 736], il y a en moyenne 0.152 $\left(\frac{14\,144}{7\,736 \times 12}\right)$ départ à l'heure par ville. Soit x l'étendue de l'intervalle autour de l'horaire demandé par le voyageur dans lequel la recherche des billets d'avion sera faite. Alors, le facteur de branchement moyen pour les avions (b_a) sera de $0.152x$. Si $x = 48$ (on recherche 24 heures autour des moments spécifiés par le voyageur), alors $b_a = 7.3$. Nous pouvons faire le même exercice pour les hôtels. Nous avons 400 hôtels répartis dans 12 villes, donc en moyenne 33 hôtels par ville. La disponibilité des hôtels varie selon la durée

du séjour, en d'autres termes, il est possible qu'un hôtel soit disponible pour 3 jours mais pas pour 4. Afin de nous permettre de calculer le facteur de branchement moyen pour les hôtels, nous avons fixé tous les séjours de nos tests à une semaine (168 heures). Nous avons montré en 2.2.4 que la probabilité moyenne qu'un hôtel soit vacant pour au moins une semaine à partir d'une heure précise est de 30%. Comme il y a une moyenne de 33 hôtels par ville, le facteur de branchement moyen (b_h) pour un séjour de 168 heures dans une ville sera de $33 \times 30\% = 10$.

La planification d'un voyage simple nécessitant deux avions et un hôtel a donc un espace de recherche dont la taille est à peu près de 490 ($7.3^2 \times 10$). Cette taille augmente à 1.7×10^{10} pour un voyage plus complexe nécessitant 10 avions et 4 hôtels ($7.3^{10} \times (10 \times 4)$). Remarquez que nous multiplions b_h par le nombre d'hôtels dans la solution plutôt que de prendre 10^4. Notre algorithme ne fait jamais de retour arrière sur les hôtels, il prend simplement toujours le meilleur en fonction des contraintes temporelles imposées par les billets d'avion choisis. Ainsi, pour un ensemble de billets d'avion donné, il y a b_h hôtels possibles. Donc, dans le cas où 4 hôtels doivent être sélectionnés, le choix devra se faire parmis $4 \times b_h$ possibilités et ce choix est définitif puisqu'il est certain qu'aucun autre hôtel ne peut être meilleur que celui qui sera sélectionné.

Nous reprenons ci-dessous la figure 2.2 afin que vous puissiez vous y référer facilement pour comprendre la géographie du monde simulé.

4.3 Analyse des résultats

Voici maintenant les résultats de nos tests. Chacune des sous-sections qui suivent présente un test, sa méthodologie et les résultats obtenus.

Notez que dans tous nos tests, la planification se fait sur un horizon d'un an, c'est-à-dire que le temps peut varier entre $t = 0$ et $t = 8\,736$ heures. Le travail de planification se fait toujours en $t = 2\,000$ et les informations contenues dans la base de données sont datées aléatoirement entre $t = 1\,000$ et $t = 2\,000$ heures.

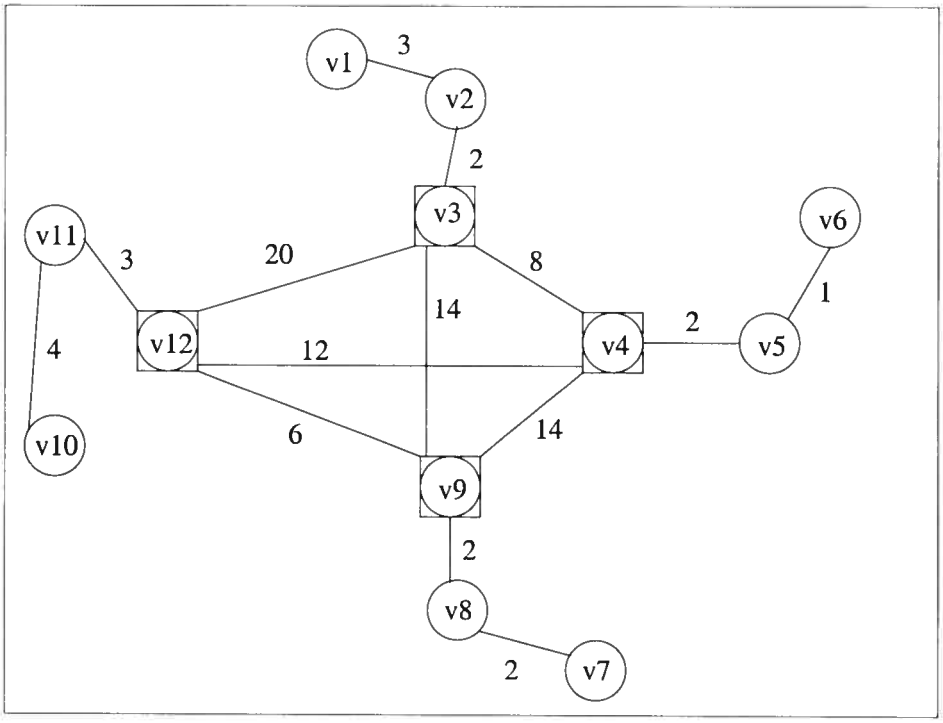

FIG. 4.1 – Graphe des distances entre les villes

4.3.1 Sensibilité aux préférences du voyageur

L'objectif de ce test est de mesurer à quel point les solutions s'adaptent aux préférences du voyageur au niveau du prix, de la qualité et de ses contraintes temporelles. Notre hypothèse veut que les solutions respectent la fonction d'utilité puisque celle-ci sert à construire les estimés qui dirigent la recherche.

Méthodologie du test

Nous avons fait plusieurs planifications pour un voyage représentatif avec des fonctions d'utilité différentes. Plus précisément, nous avons fait varier les taux de substitution entre le coût (k_1), la qualité (k_3) et le respect des contraintes temporelles (k_2) et nous avons mesuré l'influence de ces changements sur le coût monétaire, la qualité et le respect des contraintes temporelles dans les plans proposés par le planificateur. Le coût et la qualité d'un plan sont

relativement simples à mesurer, quant au respect des contraintes de temps, nous l'évaluons en mesurant la somme des *écarts à l'horaire*, c'est-à-dire la somme des différences entre les moments précisés par le voyageur et les moments réalisables dans une solution. Par exemple, si un voyageur avait demandé à être en v_1 de $t = 2\,100$ à $t = 2\,200$ et qu'un plan permette d'y être de $t = 2\,098$ à $t = 2\,187$, alors l'écart à l'horaire serait de 15.

La base de données utilisée a été générée à partir de nombres aléatoires, il est donc possible de tomber sur une section "aberrante" de notre jeu de données. Afin de s'assurer que ces cas soient détectés, chaque scénario de test est répété trois fois avec des villes différentes. Étant donnés les paramètres de génération de l'environnement, chaque pays a exactement les mêmes caractéristiques. Ainsi, faire un voyage entre deux villes de province d'un pays est *strictement équivalent** à faire un voyage entre deux villes de province d'un autre pays. Nous profiterons de cette symétrie de notre environnement pour s'assurer que les résultats ne sont pas biaisés par une situation bizarre.

Le voyage à planifier a une capitale pour point de départ. Il doit permettre de visiter une capitale d'un autre pays et ensuite, une ville de province de ce même pays. Le plan devrait donc contenir au moins 4 billets d'avion et au moins 2 hôtels.

Arrivée	Départ	Itinéraire 1	Itinéraire 2	Itinéraire 3
2 000	$2\,200 \pm 48$	v_3	v_4	v_9
$2\,200 \pm 48$	$2\,368 \pm 48$	v_4	v_9	v_{12}
$2\,368 \pm 48$	$2\,427 \pm 48$	v_5	v_8	v_{11}
$2\,427 \pm 48$	*8 736*	v_3	v_4	v_9

TAB. 4.1 – Description des itinéraires

Dans le tableau 4.1, le moment d'arrivée dans la ville de départ (2 000) signifie que le voyageur se trouvera dans cette ville au début de la planification. De la même manière, le moment de départ de la dernière ville du voyage (8 736) signifie que le voyageur restera dans cette ville pour tout l'horizon connu du planificateur, c'est-à-dire un an. Le voyageur dispose d'un budget de 4 000$, l'utilité maximale d'un plan est fixée à 1 000 et la qualité maximale

*La seule différence vient de l'asymétrie des compagnies aériennes desservant les liaisons locales, mais celle-ci n'empêche pas de détecter de possibles aberrations dans les jeux de données.

d'un avion et d'un hôtel est de 100.

Résultats obtenus

Nous avons fait trois séries de tests, faisant varier à tour de rôle k_1, k_2 et k_3 de 0 à 1 en gardant les deux autres facteurs égaux. Par exemple, lorsque nous faisons varier k_1 et que celui-ci prend la valeur 0.5, alors $k_2 = k_3 = 0.25$. Comme le montre le tableau 4.2, la caractéristique (coût, qualité, écart d'horaire) étudiée du plan s'ajuste de façon monotone aux variations du taux de substitution.

$k1$	$k2$	$k3$	Coût ($)	Écart à l'horaire (heures)	Perte de qualité moyenne (unités)
0.02	0.49	0.49	**4 218**	69	17
0.2	0.4	0.4	**2 287**	69	36
0.33	0.33	0.33	**1 337**	92	52
0.5	0.25	0.25	**1 240**	84	59
0.8	0.1	0.1	**1 240**	84	59
0.98	0.01	0.01	**1 185**	167	61
0.49	**0.02**	0.49	1 333	**96**	52
0.4	**0.2**	0.4	1 337	**92**	52
0.33	**0.33**	0.33	1 337	**92**	52
0.25	**0.5**	0.25	1 708	**69**	53
0.1	**0.8**	0.1	1 960	**51**	55
0.01	**0.98**	0.01	3 028	**41**	67
0.49	0.49	**0.02**	1 358	66	**66**
0.4	0.4	**0.2**	1 240	84	**59**
0.33	0.33	**0.33**	1 337	92	**52**
0.25	0.25	**0.5**	1 962	92	**36**
0.1	0.1	**0.8**	3 597	92	**19**
0.01	0.01	**0.98**	4 027	92	**17**

TAB. 4.2 – Influence des préférences sur le coût, la qualité et le respect des contraintes de temps des solutions proposées

Ainsi, le planificateur a su trouver un plan bien adapté aux préférences dans toutes les situations. Les figures 4.2 à 4.4 montrent d'ailleurs très clairement que les plans s'ajustent aux

fonctions d'utilité. La figure 4.3 montre qu'une croissance de l'importance relative du respect des contraintes de temps fait diminuer l'écart à l'horaire, mais en contrepartie, fait augmenter le coût monétaire. De même, la figure 4.4 nous montre qu'une croissance de l'importance relative de la qualité fait diminiuer la perte de qualité moyenne, mais fait augmenter le coût monétaire. Bien entendu, cette adaptation est possible dans la mesure où il y a suffisamment d'opérateurs disponibles pour que plusieurs solutions alternatives puissent être produites.

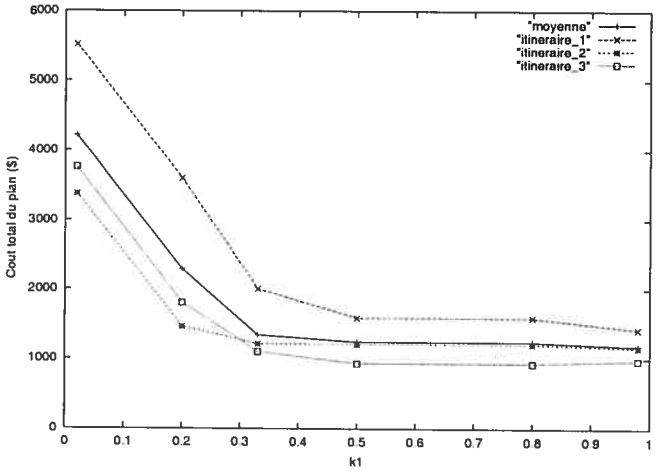

FIG. 4.2 – Effet du facteur k_1 (importance du coût ($) du plan) sur le coût monétaire du plan

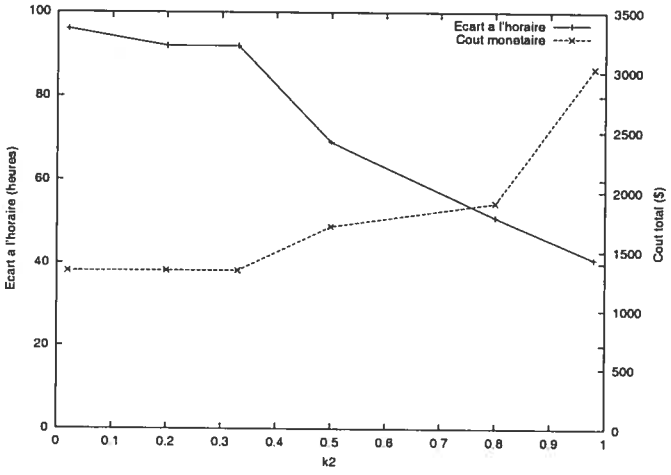

FIG. 4.3 – Effet du facteur k_2 (importance de l'écart à l'horaire) sur l'écart à l'horaire et le coût monétaire du plan

La figure 4.2 est différente des deux autres. Nous y avons tracé trois courbes correspondant

aux plans issus des trois scénarios frères testés et une pour la moyenne des trois. Nous souhaitons ainsi montrer la réelle symétrie entre les trois itinéraires pour lesquels nous avons fait la planification. Remarquez que l'itinéraire 1 a un coût systématiquement plus élevé que les deux autres. Il y a deux explications possibles à cette différence. D'abord, les distances à parcourir sont un peu plus grandes dans l'itinéraire 1, les billets d'avion sont donc plus chers. Ensuite, le hasard des disponibilités a pu jouer en défaveur de cet itinéraire. L'important est de voir que la forme des trois courbes est la même.

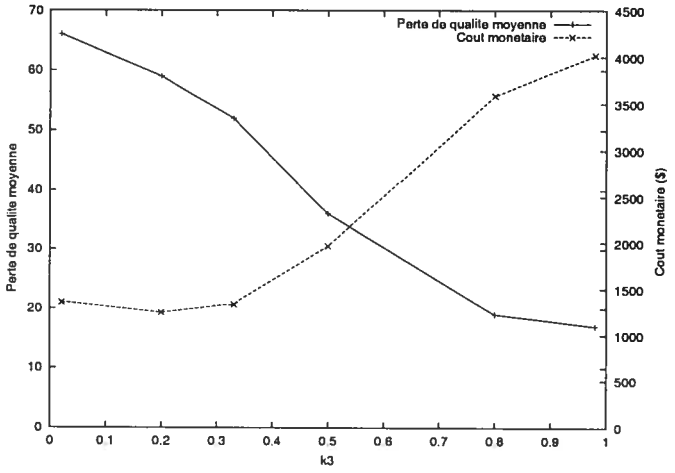

FIG. 4.4 – Effet du facteur k_3 (importance de la qualité) sur la perte de qualité moyenne et le coût monétaire du plan

Afin de vous donner un aperçu des différences entre les plans adaptés à différentes fonctions d'utilité, nous vous présentons trois plans où le poids d'une utilité est beaucoup plus fort que celui des deux autres. Ces plans correspondent à la solution proposée pour le scénario de test correspondant à l'itinéraire 2.

La figure 4.5 illustre le plan proposé à un voyageur accordant une forte importance relative au coût monétaire de son voyage. Le plan permet au voyageur de profiter d'un séjour à v_9 et à v_8. Pour chaque hôtel et chaque avion, nous indiquons l'intervalle de temps durant lequel l'opérateur est actif, le coût monétaire et la qualité de l'opérateur. Comme cette qualité correspond à une note sur 100, la perte de qualité moyenne du plan est obtenu en faisant la somme des différences entre 100 et la qualité de chaque opérateur, puis en divisant par le nombre d'opérateurs dans le plan (37 pts de qualité).

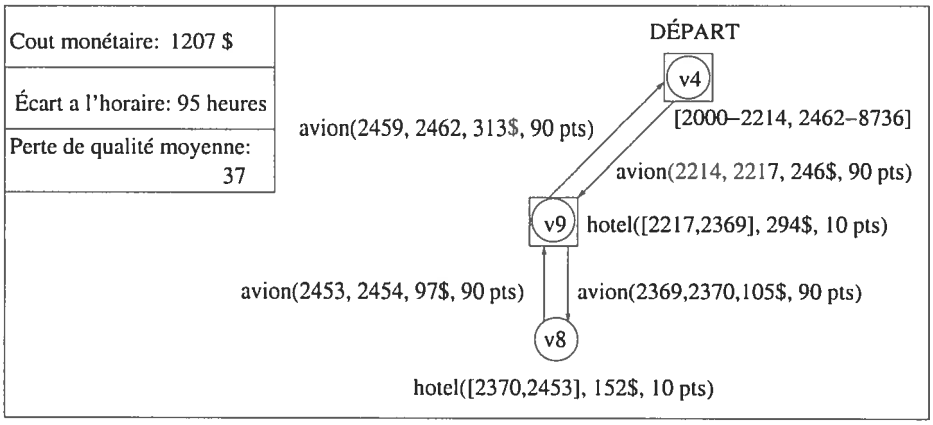

FIG. 4.5 – Plan proposé lorsque le coût monétaire prime ($k_1 = 0.8$, $k_2 = 0.1$, $k_3 = 0.1$)

Le coût monétaire s'obtient simplement en additionnant les prix de chaque opérateur (1 207$) et l'écart à l'horaire correspond à la somme des écarts pour chaque contrainte spécifiée dans le tableau 4.1. Ce tableau affiche quatre contraintes : v_4 : [2 000, 2 200], v_9 : [2 200, 2 368], v_8 : [2 368, 2 427] et v_4 : [2 427, 8 736]. Or, ce plan permet d'obtenir l'horaire : v_4 : [2 000, 2 214], v_9 : [2 217, 2 369], v_8 : [2 370, 2 453] et v_4 : [2 462, 8 736]. L'écart à l'horaire total sera donc de 95 heures.

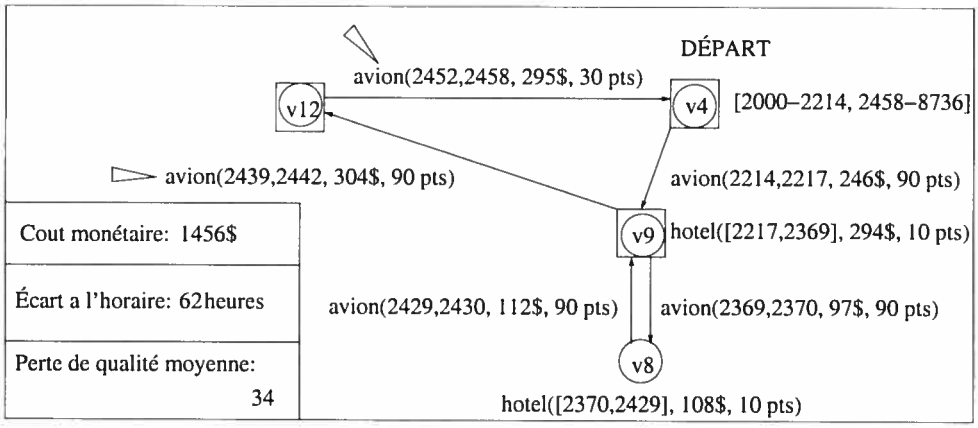

FIG. 4.6 – Plan proposé lorsque le temps prime ($k_1 = 0.1$, $k_2 = 0.8$, $k_3 = 0.1$)

On remarque que pour gagner sur le respect des contraintes de temps, le plan de la figure 4.6 prendra un avion de plus que le plan illustré en 4.5. Même si cela augmente considérablement le prix et diminue légèrement la qualité, le planificateur fait cette suggestion afin de se plier

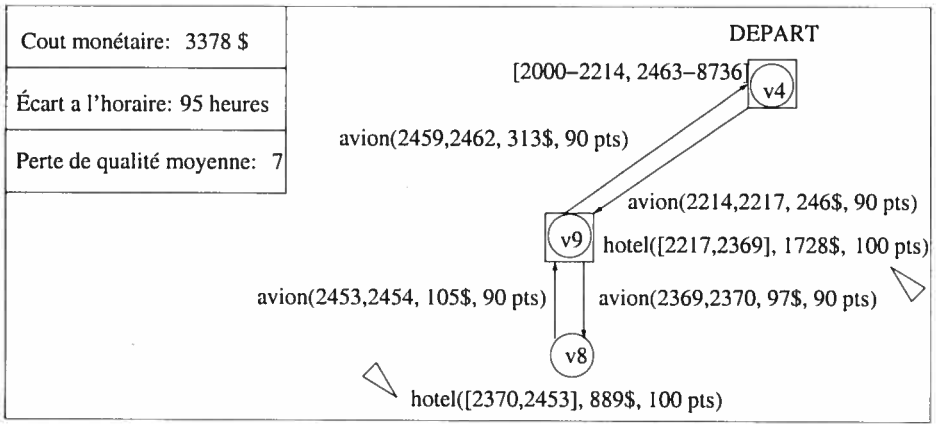

FIG. 4.7 – Plan proposé lorsque la qualité prime ($k_1 = 0.1$, $k_2 = 0.1$, $k_3 = 0.8$)

aux préférences du voyageur.

Remarquez aussi que le plan de la figure 4.7 est composé des mêmes avions que celui en 4.5. Par contre, en voulant maximiser la qualité, il propose des hôtels plus luxueux, même si cela coûte très cher.

4.3.2 Erreurs d'estimation de $h(n)$ et taille des solutions

Nous en discutions au chapitre 1, la stratégie de recherche A^* a, en pire cas, une complexité exponentielle dans la taille de la solution lorsque l'erreur d'estimation relative entre l'estimation $h(n)$ du coût d'utilité et le vrai coût $h^*(n)$ $\left(\frac{h(n)-h^*(n)}{h^*(n)}\right)$ est constante dans la taille de la solution. Toutefois, si l'erreur d'estimation absolue $(h(n) - h^*(n))$ est constante dans la taille de la solution, A^* aura une complexité linéaire en pire cas. Avant de discuter du temps d'exécution en 4.3.3, il nous semble important d'observer le comportement des erreurs d'estimation relative et absolue lorsque la taille de la solution augmente.

Méthodologie du test

Nous avons planifié des voyages pour toutes les permutations de 1, 2, 3 et 4 villes possibles en partance d'une capitale. En particulier, nous avons planifié tous les voyages possibles en partance de v_3 permettant de visiter 1, 2, 3 et 4 villes. Comme il y a 12 villes dans notre monde

et que nous avons évité de visiter deux fois la même ville dans un même voyage, 9 031 plans différents pouvaient potentiellement être générés. Toutefois, en raison d'une insuffisance de mémoire sur laquelle nous reviendrons en 4.3.3, nous avons plutôt obtenu 1 766 plans dont la taille varie entre 3 et 19 opérateurs[†]. Ces 1 766 plans optimaux et les informations recueillies lors de leur construction alimenteront les discussions de ce test et du test 4.3.3.

Notez que la fonction d'utilité utilisée accorde une importance égale au prix, à la qualité et au respect des contraintes de temps ($k_1 = k_2 = k_3 = 0.33$). Le budget est de 5 000$ et la qualité maximale d'un avion et d'un hôtel est de 100.

Résultats obtenus

La figure 4.8 illustre l'erreur d'estimation relative en fonction de la taille de la solution. L'erreur d'estimation relative est une fonction de n, le noeud à partir duquel on fait l'estimation $h(n)$ de la meilleure solution. Nos graphiques représentent la moyenne des erreurs relatives pour des solution de tailles variables. La relation décroissante entre l'erreur relative et le nombre d'opérateurs dans la solution indique que notre algorithme aura une complexité moindre qu'exponentielle en pire cas. Rappelons que si nous avions trouvé une relation constante entre l'erreur relative et la taille de la solution, la complexité en pire cas aurait été exponentielle.

La figure 4.9 illustre l'erreur d'estimation absolue en fonction de la taille de la solution. La relation croissante nous indique que la complexité en pire cas de notre algorithme sera supérieure à une croissance linéaire. Rappelons que si la figure 4.9 avait affiché une relation constante, nous aurions pu conclure à une complexité linéaire en pire cas.

La conclusion que nous pouvons tirer ici est que notre algorithme a une complexité en pire cas quelque part entre une croissance linéaire et une croissance exponentielle. L'éventail est très large, mais nous savons néanmoins que même dans la pire situation de recherche, la complexité devrait être moindre qu'exponentielle, ce qui est en soit un résultat important.

[†] Un plan compte toujours $\sim 20\%$ d'hôtels et $\sim 80\%$ d'avions.

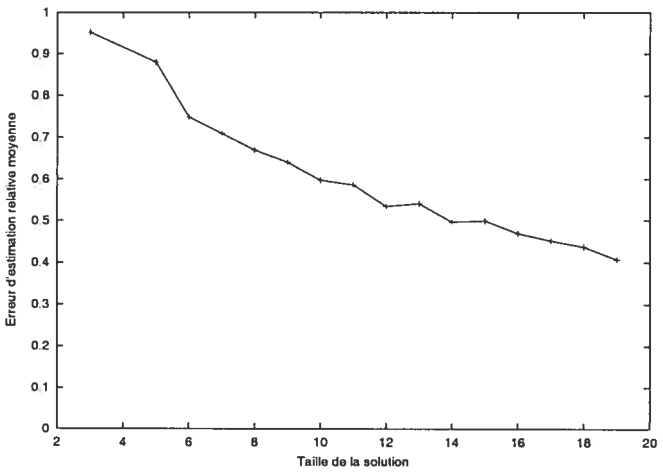

FIG. 4.8 – Erreur d'estimation relative moyenne en fonction de la taille de la solution

4.3.3 Temps d'exécution et consommation de mémoire

Typiquement, un algorithme de planification (ou de recherche) consomme deux types de ressources : du temps et de l'espace mémoire. Comme nous le soulignions plus tôt, le choix d'un algorithme de recherche revient souvent à faire un compromis entre la performance et la qualité des solutions. Nous avons choisi la qualité au détriment de la performance en optant pour l'algorithme de recherche A^* qui trouve toujours les meilleures solutions mais dont le temps de recherche peut être relativement élevé. En faisant ce choix, nous supposions que les solutions seraient suffisamment petites pour que la complexité de l'algorithme ne le rende pas inadéquat pour l'utilisation que nous voulons en faire.

Nous avons deux objectifs : d'abord, évaluer le temps nécessaire à la planification de différents types de voyage et ensuite, déterminer l'importance de la taille de la solution et du nombre de solutions partielles explorées sur le temps de calcul total. Les tests que nous avons réalisés pour satisfaire ces deux objectifs nous permettront de discuter de la consommation de mémoire de notre implantation.

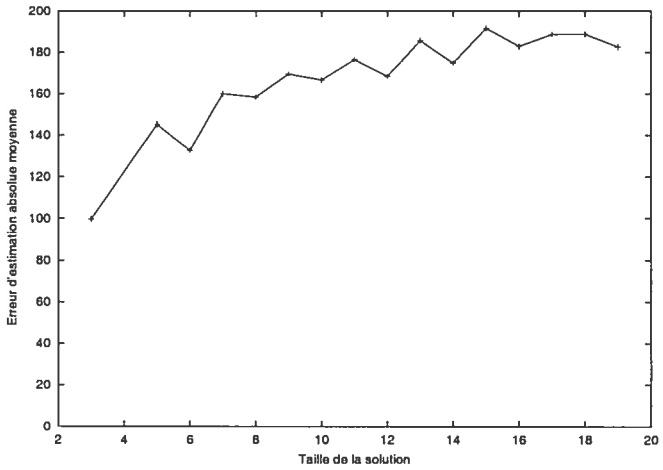

FIG. 4.9 – Erreur d'estimation absolue moyenne en fonction de la taille de la solution

Méthodologie du test

Ce test utilise exactement la même méthodologie que le test 4.3.2. Nous vous invitons donc à vous y référer pour un rappel des détails.

Résultats obtenus

Les figure 4.10 illustre le temps d'exécution en fonction de la taille de la solution. On y voit la croissance du temps total, du temps nécessaire aux calculs initiaux (matrices des estimations), du temps d'accès à la base de données durant la planification et du temps nécessaire à la planification elle-même.

Le temps nécessaire aux calculs initiaux est constant à environ 5 secondes. Cette constance ne devrait pas nous surprendre puisque la complexité de ces calculs dépend seulement du nombre de villes dans l'environnement. Le temps d'accès à la base de données est relativement important, mais croît tout de même moins vite que le temps nécessaire à la planification proprement dite. Globalement, moins de 10 secondes sont nécessaires à la production d'un plan optimal pour un problème de taille raisonnable (moins de 8 opérateurs). De ces 10 secondes, seulement 2 sont utilisées par l'algorithme de planification. Tout le reste du temps est passé dans la base de données ou sur des calculs initiaux. Chose certaine, la complexité en temps

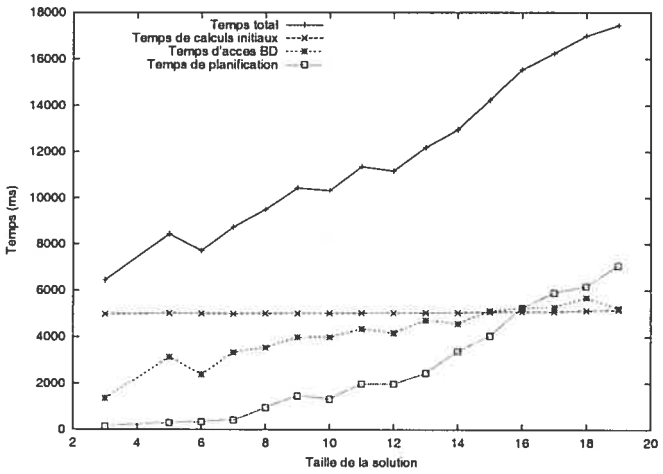

FIG. 4.10 – Temps d'exécution en fonction de la taille de la solution

de notre implantation sur ce jeu de tests est loin d'être exponentielle. Elle nous semble en fait pratiquement linéaire, mais avec un fort taux de croissance.

Il est aussi intéressant d'observer la relation entre le temps de calcul et le nombre de solutions partielles explorées. La figure 4.11 illustre la relation entre la taille des solutions et le nombre de plans partiels explorés. La relation est semblable à celle entre le temps de calcul et la taille des solutions, ce qui est logique puisque la majeure partie du temps de calcul est perdu à explorer de mauvais plans partiels. Le graphique nous indique qu'un peu moins de 2 000 plans partiels sont explorés dans des problèmes dont la taille de la solution est 8.

La figure 4.12 montre la relation entre le nombre de plans partiels explorés et le temps nécessaire à la planification. Encore une fois, il n'y a pas de surprise à obtenir le même type de relation qu'entre la taille de la solution et le temps de calcul.

La consommation de mémoire est une autre caractéristique importante d'un algorithme de recherche. Dans le cas particulier de A^*, la croissance de la consommation de mémoire peut rapidement devenir problématique. Rappelons que A^* garde en mémoire tous les plans partiels jusqu'à ce qu'ils ne puissent être développés davantage.

La relation entre le nombre de plans partiels explorés et la taille de la solution est à peu près linéaire. Or, c'est justement le nombre de plans partiels explorés qui détermine l'espace

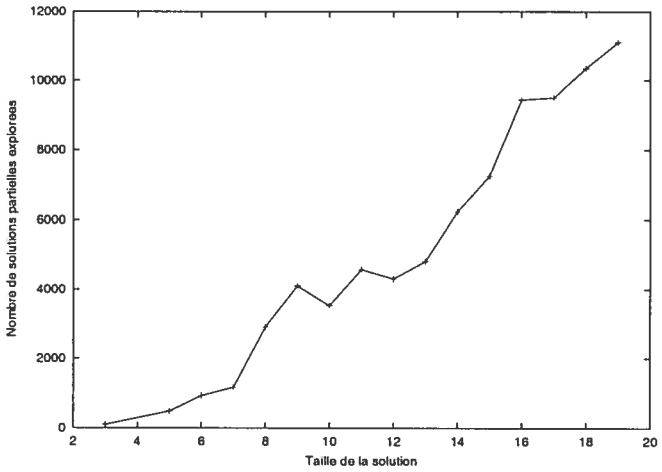

FIG. 4.11 – Nombre de plans partiels explorés en fonction de la taille de la solution

mémoire utilisé. Plus précisément, c'est le nombre de plans partiels toujours présents dans la liste des solutions partielles explorables qui consomme pratiquement toute la mémoire. Ils serait donc légitime de poser la question : "pourquoi la mémoire devient-elle rapidement un problème si sa consommation croît linéairement ?". La réponse se trouve dans nos choix d'implantation. Il s'avère que la mémoire est toute utilisée lorsqu'environ 30 000 plans partiels sont maintenus en mémoire. Cela survient lorsqu'environ 40 000 plans partiels ont été explorés. Ces chiffres nous semblent petits pour occasionner un problème de mémoire. Toutefois, nous devons admettre que la structure que nous utilisons pour représenter un plan est relativement lourde, ce qui constitue à notre avis la raison pour laquelle des problèmes de mémoire surviennent aussi vite.

4.3.4 Compromis entre le nombre de solutions et la qualité de ces solutions

Notre planificateur produit le meilleur plan, mais est aussi conçu pour produire *les* meilleurs plans. Nous nous intéresserons ici au coût d'utilité de plans produits en plus du plan optimal, du temps et de l'exploration supplémentaires nécessaires pour les trouver.

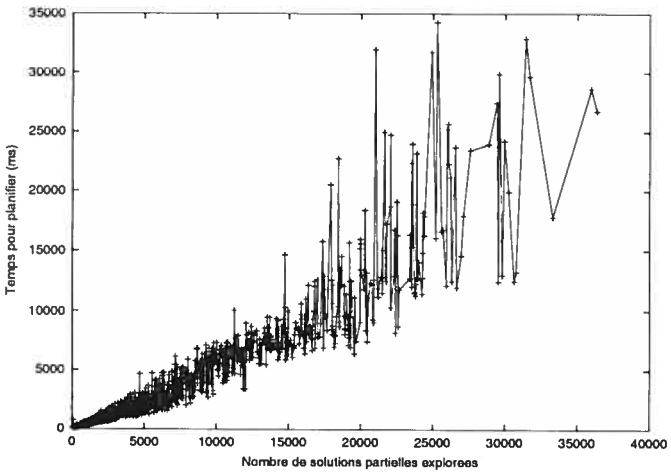

FIG. 4.12 – Temps de planification en fonction du nombre de plans partiels explorés

Méthodologie du test

Ce test a été fait pour un scénario de test simple (voyage d'une semaine dans la capitale d'un pays étranger) et avec une fonction d'utilité accordant une importance égale à chaque facteur. Nous avons laissé le planificateur trouver tous les plans faisables pour ce voyage ; il y en avait 326.

Résultats obtenus

La figure 4.13 montre l'évolution du coût d'utilité avec le rang des solutions. Le rang représente l'ordre de découverte des plans. Par exemple, un plan au rang i serait le *ième* plan trouvé. Pour cet exemple particulier, la décroissance de l'utilité (ou l'augmentation du coût d'utilité) est relativement faible. En fait, moins de 50% de l'utilité est perdu entre le meilleur plan et le pire. Évidemment, seul le contenu de la base de données est responsable de ce résultat. Nous le présentons seulement afin de caractériser l'exemple que nous analysons.

Ce qui est plus intéressant, c'est le temps marginal nécessaire à la découverte des plans supplémentaires. La figure 4.14 montre la relation entre le nombre de plans trouvés et les différentes mesures du temps. Le temps de calcul semble converger asymptotiquement. Il est donc très peu coûteux de demander des plans supplémentaires une fois que quelques-uns ont

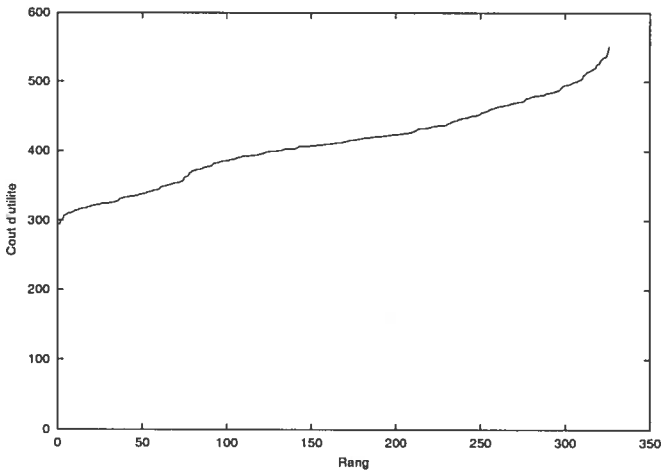

FIG. 4.13 – Coût d'utilité des plans de différents rangs

déjà été produits.

Regardons finalement le nombre de plans partiels explorés en fonction du nombre de plans complets trouvés (figure 4.15). La relation est croissante comme nous pouvions nous y attendre, mais remarquez le changement soudain de la pente entre le centième et le cent cinquantième plan. Une branche complète a permis de trouver les 100 premiers plans et il a ensuite fallu explorer une nouvelle branche ce qui explique le changement de pente.

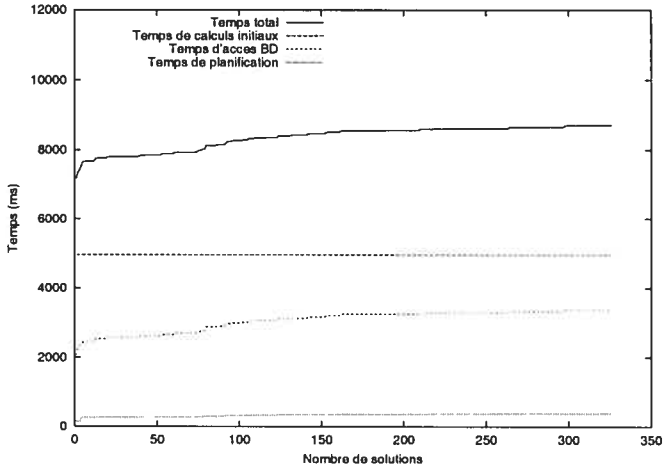

FIG. 4.14 – Temps d'exécution en fonction du nombre de plans trouvés

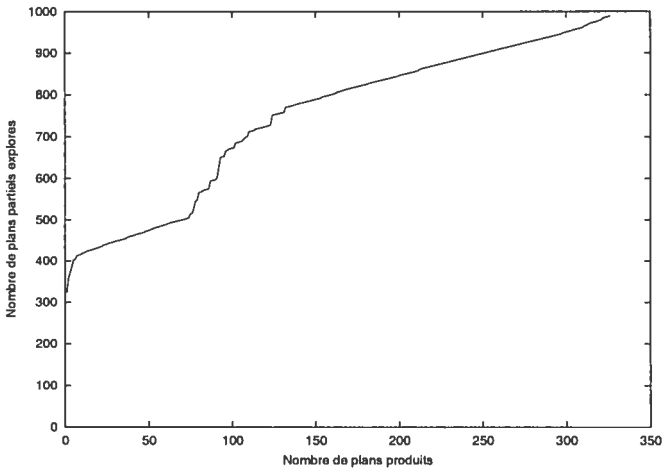

FIG. 4.15 – Nombre de plans partiels explorés en fonction du nombre de plans trouvés

4.4 Discussion

Ces résultats nous permettent de porter un jugement sur la qualité des solutions produites par notre implantation et d'en déduire sa limite pratique, c'est-à-dire la complexité maximale des problèmes que nous avons pu résoudre. Au niveau de la qualité des solutions, notre stratégie nous apparaît excellente puisqu'elle trouve toujours les meilleures solutions et que ces solutions s'adaptent très bien aux préférences du voyageur. L'optimalité a toutefois un coût : la complexité en temps et en espace mémoire est potentiellement très élevée. L'analyse en 4.3.2 indique que notre algorithme a une complexité en pire cas qui se situe quelque part entre une croissance linéaire et exponentielle dans la taille de la solution. Bien que ces bornes soient très larges, elles nous indiquent tout de même que la complexité sera toujours moindre qu'exponentielle, ce qui constitue un résultat appréciable lorsqu'on utilise une recherche A^*.

Les temps d'exécution sont assez courts (quelques secondes) pour des problèmes raisonnables (moins de 10 opérateurs). C'est la consommation de mémoire qui limite la taille des problèmes résolvables avec notre planificateur, le plus gros problème que nous ayons pu résoudre comptait 24 opérateurs. Remarquez que la limite s'exprime en terme du nombre d'opérateurs et non en terme du nombre de contraintes car il est possible qu'un problème comptant peu de contraintes ait une solution optimale comptant de nombreux opérateurs ; l'ajout d'un opérateur obligeant parfois l'ajout d'une nouvelle contrainte à satisfaire.

Conclusion

Nous avons montré comment les techniques de planification traditionnelles peuvent être adaptées afin de tenir compte des préférences d'un agent évoluant dans un environnement dynamique et incertain. Notre façon de faire nous a permis de proposer une solution au problème particulier de la planification de voyages composés d'avions et d'hôtels. L'utilisation d'une recherche A^* dirigée par le coût d'utilité des plans partiels nous permet de trouver un plan optimal en fonction des préférences du voyageur. Comme les résultats de nos tests l'ont montré, les plans produits s'adaptent très bien aux préférences de l'agent et le temps de calcul, qui aurait pu être problématique en raison de la complexité de A^*, est en fait très raisonnable parce que la taille des solutions est relativement petite.

Nous considérons notre travail comme une fusion de plusieurs idées, allant de l'intelligence artificielle à la recherche opérationnelle, en passant par les sciences économiques et les statistiques. Nous n'avons toutefois pas approfondi chacun de ces domaines, notre objectif étant surtout de fournir une manière de combiner différentes techniques afin de développer une stratégie de solution pour un problème précis. Il serait maintenant intéressant d'approfondir chacune des facettes de notre travail afin de proposer des modèles plus réalistes. Dans le volet intelligence artificielle, il serait à notre avis pertinent d'utiliser des méthodes d'apprentissage afin de pondérer des heuristiques de recherche comme le propose d'ailleurs Zimmerman et Kambhampati [ZK03]. À titre d'exemple, notre hypothèse voulant que les avions soient plus rares que les hôtels n'est pas toujours vraie dans la réalité, or il serait possible d'adapter cette stratégie en fonction des périodes de l'année par exemple. Du côté de la recherche opérationnelle, nous croyons qu'une implantation de plusieurs algorithmes de recherche pourrait permettre au planificateur de déterminer lui-même la stratégie de recherche la plus appropriée. Par exemple,

pour un problème ayant une solution nécessitant beaucoup d'opérateurs et donc pour laquelle l'algorithme A^* n'est pas bien adapté, le planificateur pourrait opter pour une stratégie gloutonne, moins gourmande en mémoire. Celle-ci ne fournirait peut-être pas la meilleure solution, mais elle pourrait au moins en fournir une qui soit acceptable. Les économistes pourraient de leur côté se pencher sur le problème des fonctions d'utilité. Il est possible, par des techniques de préférences révélées, de trouver des fonctions d'utilité représentant beaucoup mieux les préférences d'un individu que les fonctions que nous avons employées. Finalement, une recherche en statistique pourrait permettre de fournir des modèles d'incertitude plus justes que le nôtre afin qu'une incertitude beaucoup plus complexe puisse être modélisée. Dans le cas de la planification de voyage, il est clair qu'une incertitude sur le prix ne représente pas adéquatement l'incertitude réelle quant aux informations disponibles sur Internet. Il faudrait minimalement tenir aussi compte de la disponibilité des opérateurs.

Pour continuer avec les travaux pouvant faire suite à ce mémoire, nous croyons que le développement d'un algorithme de tri des plans pourrait être très utile. Dans ce travail, nous suggérions que le classement des solutions alternatives soit fait sur une base de dominance stochastique. Or, si nous considérons qu'il y a un coût associé à l'exploration des plans afin d'en vérifier la faisabilité et d'en mesurer l'utilité réelle, un classement selon la moyenne et la variance de la distribution d'utilité des plans n'est pas nécessairement la meilleure. Des recherches en sciences économiques suggèrent que les alternatives devraient être classées selon un critère tenant compte de la distribution d'utilité d'une solution et du coût d'exploration qu'elle nécessite. Ainsi, les alternatives pourraient être explorées selon leur ordonnancement et l'exploration de nouvelles alternatives pourrait s'arrêter lorsque l'utilité réelle d'une solution explorée atteint un seuil certifiant que le coût d'exploration de la solution suivante serait trop élevé par rapport à son utilité espérée. Nous suggérons un article de Weitxman [Wei79] comme point de départ pour cette recherche.

Soulignons aussi que notre travail s'est limité à un environnement dans lequel évolue un seul agent. Or, il y a de plus en plus d'intérêt pour les systèmes multi-agents. Dans ce genre de système, la planification revêt une dimension beaucoup plus complexe puisqu'elle doit prendre en considération les plans des autres agents, qu'ils soient amis ou ennemis. Nous référons le lecteur intéressé à approfondir cette problématique aux travaux de Boutilier et Brafman

[BB01].

Même si notre application est limitée à la planification de voyages, la stratégie que nous avons développée pourrait être utilisée pour toutes sortes de problèmes ayant des caractéristiques similaires au nôtre. En plus de devoir être exprimée sous la forme d'opérateurs *STRIPS*, ces problèmes devraient avoir un modèle d'incertitude connu du planificateur et des fonctions d'utilité bien définies, permettant de diriger la recherche. L'algorithme de recherche A^* que nous utilisons exige aussi que la taille de la solution soit relativement petite afin d'éviter que l'explosion combinatoire ne crée des problèmes d'espace mémoire et de temps de calcul.

Nous espérons que ce mémoire aura permis d'illustrer les immenses possibilités offertes par une combinaison de techniques chevauchant plusieurs domaines. Cette façon de faire permet d'ouvrir les horizons de recherche et de proposer des solutions originales. Au-delà de la planification de voyages pour laquelle il est évident que notre stratégie s'applique très bien, nous croyons que la façon de faire générale proposée dans ce mémoire devrait être applicable à tous les problèmes pour lesquels l'agent doit agir en fonction de préférences pouvant être exprimées sous la forme d'une fonction d'utilité.

Bibliographie

[BAVGL+02] H. Ben-Ameur, S. Vaucher, R. Gérin-Lajoie, P. Kropf et B. Chaib-draa. « Towards an Agent Based Approach for Multimarket Package e-Procurement ». In *Proceedings of the Fifth International Conference on Electronic Commerce Research (ICERC-5)*, http ://www2.cirano.qc.ca/ benameuh/en/publications.php, Montréal, 2002.

[BB01] C. Boutilier et R. I. Brafman. « Partial-order planning with concurrent interacting actions ». *Journal of Artificial Intelligence Research*, volume 14, numéro 1, pages 105–136, 2001.

[BDH99] C. Boutilier, T. Dean et S. Hanks. « Decision theoretic planning : structural assumptions and computational leverage ». *Journal of AI Research*, volume 11, numéro 1, pages 1–94, 1999.

[BF97] A. L. Blum et M. L. Furst. « Fast planning trough planning graph analysis ». *Artificial intelligence*, volume 90, numéro 1-2, pages 281–300, 1997.

[Byl92] T. Bylander. « Complexity results for serial decomposability ». In *Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 729–734, San-Jose, Californie, 1992.

[DHW94] D. Draper, S. Hanks et D. Weld. « Probabilistic planning with information gathering and contingent execution ». In *Proceedings of the Second International Conference on AI Planning Systems*, pages 31–36, Chicago, Illinois, 1994.

[Flo62] R. Floyd. « Algorithm 97 : Shortest path ». *Communications of the ACM*, volume 5, numéro 6, page 345, 1962.

[FN71] R. E. Fikes et N. J. Nilsson. « STRIPS : a new approach to the application of theorem proving to problem solving ». *Artificial Intelligence*, volume 59, numéro 1-2, pages 227–232, 1971.

[Gas79] J. Gaschnig. « Performance measurement and analysis of certain search algorithms. Ph.D. dissertation ». Rapport technique CMU-CS-79-124, Département d'informatique, Université de Carnegie-Mellon, 1979.

[Gre00] W. H. Green. *Econometric Analysis*. Prentice Hall, fourth édition, 2000.

[HH98] P. Haddawy et S. Hanks. « Utility models for goal-directed decision theoretic planners ». *Computational Intelligence*, volume 14, numéro 3, pages 392–429, 1998.

[HNR68] P. E. Hart, N. J. Nilsson et B. Raphael. « A formal basis for the heuristic determination of minimum cost paths ». *IEEE Transactions of Systems Science and Cybernetics*, volume SSC-4, numéro 2, pages 100–107, 1968.

[HNR72] P. E. Hart, N. J. Nilsson et B. Raphael. « Correction to "A formal basis for the heuristic determination of minimum cost paths" ». *SIGART Newsletter*, numéro 37, pages 28–29, 1972.

[KHW95] N. Kushmerick, S. Hanks et D. Weld. « An algorithm for probabilistic planning ». *Artificial Intelligence*, volume 76, numéro 1-2, pages 239–286, 1995.

[La00] D. Long et al. « The AIPS-98 planning competition ». *AI Magazine*, volume 21, numéro 2, pages 13–33, 2000.

[McD00] D. McDermott. « The 1998 AI planning systems competition ». *AI Magazine*, volume 21, numéro 2, pages 35–55, 2000.

[MCWG95] A. Mas-Colell, M. D. Whinston et J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[MR91] D. McAllester et D. Rosenblitt. « Systematic non-linear planning ». In *Proc. 9th Ann. Conference on Artificial Intelligence*, volume 2, pages 634–639, Anaheim, Californie, 1991.

[OP99] N. Onder et M. E. Pollack. « Conditional, probabilistic planning : A unifying algorithm and effective search control mechanisms ». In *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, Wisconsin, 1999.

[Pea84] J. Pearl. *Heuristics : Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, Massachusetts, 1984.

[PH99] M. E. Pollack et J. F. Horty. « There's more to life than making plans : Plan management in dynamic, multi-agent environments. ». *AI Magazine*, volume 20, numéro 4, pages 71–84, 1999.

[Poh70] I. Pohl. *Machine Intelligence 5*, chapitre First results on the effect of error in heuristic search, pages 219–236. Elsevier/Notrh Holland, Amsterdam, London, New-York, 1970.

[Poh77] I. Pohl. *Machine Intelligence 8*, chapitre Practical and theoretical considerations in heuristic search algorithms, pages 55–72. Ellis Horwood, Chichester, England, 1977.

[PS92] M. A. Peot et D. E. Smith. « Conditional nonlinear planning ». In *Proceedings of the First International Conference on AI Planning Systems*, College Park, Maryland, 1992.

[RN95] S. Russell et P. Norvig. *Artificial Intelligence : A Modern Approach.* Prentice Hall, 1995.

[Sac74] E. D. Sacerdoti. « Planning in a hierarchy of abstraction spaces ». *Artificial Intelligence*, volume 5, numéro 2, pages 115–135, 1974.

[War76] D. H. D. Warren. « Generating conditional plans and programs ». In *Proceedings of the AISB Summer Conference*, Edinburgh, 1976.

[Wei79] M. L. Weitzman. « Optimal search for the best alternative ». *Econometrica*, volume 47, numéro 3, pages 641–654, 1979.

[Wei99] G. Weiss, éditeur. *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence*, chapitre Intelligent Agents, pages 27–77. MIT Press, 1999.

[Wel94] D. S. Weld. « An introduction to least commitment planning ». *AI Magazine*, volume 15, numéro 4, pages 27–61, 1994.

[WJ95] M. Wooldridge et N. R. Jennings. « Intelligents agents : Theory and practice ». *The Knowledge Engineering Review*, volume 10, numéro 2, pages 115–152, 1995.

[ZK03] T. Zimmerman et S. Kambhampati. « Learning-assisted automated planning ». *AI Magazine*, volume 24, numéro 2, pages 73–96, 2003.