UNIVERSITÉ DU QUÉBEC À MONTRÉAL

PLANIFICATION À SATISFACTION PARTIELLE SOUS INCERTITUDE TEMPORELLE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

SYLVAIN LABRANCHE

JUIN 2015

REMERCIEMENTS

Je tiens d'abord à remercier mon directeur de maîtrise, le professeur Éric Beaudry. Sans son encadrement et ses encouragements, la motivation m'aurait peut-être manqué face à la charge de travail monumentale nécessaire pour accomplir le travail présenté dans ce mémoire.

Ensuite, j'aimerais remercier les personnes qui ont participé à la création du jeu *SimRealEstate* : Nicolas Sola, Émeric Morin, Sophie Callies et Christophe Gigax. Leur travail acharné m'a permis de tester mon approche de planification dans le jeu *SimRealEstate*. Sans leur contribution, je n'aurais probablement pas pu rédiger l'article associé ni aller le présenter en Allemagne.

Merci aussi au Conseil de recherche en sciences naturelles et en génie du Canada, pour le financement des projets auxquels j'ai pu participer.

Je termine par les remerciements plus «humains». Merci grand-maman de n'être pas trop déçue du fait que j'aie choisi la voie de la connaissance au lieu de celle de l'argent. Merci maman de n'avoir jamais douté de moi, de m'avoir toujours encouragé et de ne jamais juger mes choix, mais bien de les accepter. Merci Mathieu de faire ton travail de frère. Merci Émilie, de te plaindre juste ce qu'il faut lorsque je me lève tôt le matin pour travailler, mais de quand même me laisser partir avec le sourire.

[Cette page a été laissée intentionnellement blanche]

TABLE DES MATIÈRES

[Cette page a été laissée intentionnellement blanche]

LISTE DES TABLEAUX

[Cette page a été laissée intentionnellement blanche]

LISTE DES FIGURES

[Cette page a été laissée intentionnellement blanche]

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

APL	*ActuPlan Language*
ActuPlan	*Action Contingency Time Uncertainty Planner*
BCPE	*Bounded Cost Plan Existence*
GOAP	*Goal-Oriented Action Planning*
ICAPS	*International Conference on Automated Planning and Scheduling*
IPC	*International Planning Competition*
PDDL	*Planning Domain Definition Language*
MDP	Processus décisionnel de Markov
PNJ	Personnage non-joueur
PSP	Planification à satisfaction partielle

[Cette page a été laissée intentionnellement blanche]

RÉSUMÉ

Le présent mémoire introduit d'abord la planification automatisée, une branche active de l'intelligence artificielle. Cette introduction permet de saisir la pertinence de la problématique principale du mémoire : la planification à satisfaction partielle. Le modèle classique de la planification automatisée est contraint par plusieurs hypothèses qui limitent son applicabilité dans la vie réelle. Sous le modèle classique, tous les objectifs doivent être satisfaits, sinon aucune solution n'est retournée. En planification à satisfaction partielle, cette hypothèse est relaxée et le problème devient celui de la sélection des objectifs et de la planification. Dans ces pages, nous décrivons d'abord l'état de l'art dans un contexte déterministe. Par la suite, nous relaxons l'hypothèse du monde déterministe et expliquons en quoi les méthodes actuelles ne suffisent pas. Nous présentons la méthode par actions d'abandon que nous avons développée et qui permet de choisir les objectifs pendant l'exécution du plan en tenant compte d'une incertitude sur la durée des actions.Cette méthode a été présentée lors de deux conférences, sous la forme d'un article complet (Labranche et Beaudry (2014a)) et d'une affiche (Labranche et Beaudry (2014b)). Dans une seconde partie, nous montrons comment la planification à satisfaction partielle peut être utilisée dans un jeu vidéo pour animer des personnages non-joueurs. Nous proposons une approche novatrice pour la sélection autonome d'objectifs par des agents intelligents. Cette approche est implémentée, intégrée et expérimentée dans le jeu sérieux de simulation immobilière SimRealEstate. Ces travaux ont également mené à une publication et ont été communiqués lors d'une conférence (Labranche *et al.* (2014)).

MOTS-CLÉS : intelligence artificielle, planification automatisée, planification à satisfaction partielle, incertitude, jeux sérieux.

[Cette page a été laissée intentionnellement blanche]

INTRODUCTION

L'intelligence artificielle est une branche de l'informatique dont le but ultime est la création d'agents intelligents (Russell et Norvig, 2009). Cette notion d'intelligence est toutefois subjective et englobe des visions très différentes les unes des autres. Dans Russell et Norvig (2009), les auteurs font état de quatre manières de voir l'intelligence artificielle, soit le développement, d'un côté, d'agents qui *pensent* ou *agissent* humainement ou, de l'autre, d'agents qui *pensent* ou *agissent* rationnellement.

En effet, nous pouvons en premier lieu désirer que les agents que nous développons se rapprochent le plus possible d'un être humain. Nous dirons ainsi que ces agents *pensent* ou *agissent* **humainement**. Les recherches dans cette direction visent donc la création d'agents qui, ultimement, pourraient tromper un être humain en lui faisant croire qu'il a affaire à un autre être humain. Le test de Turing (Turing, 1950) en rend compte : un être humain discute avec, d'une part, un autre être humain et, de l'autre, un agent artificiel. Si le premier humain ne peut déterminer lequel de ses deux interlocuteurs est une machine, le logiciel passe le test. Cette vision de l'intelligence artificielle est celle qui, historiquement, soulève le plus de questions philosophiques. Plusieurs médias traitent de la question, notamment au cinéma avec des films comme *Artificial Intelligence : AI* (2001), où un enfant robot aspire à être un «vrai» humain, ou encore *Her* (2013) dans lequel un homme tombe amoureux d'un système d'exploitation.

Si la notion d'agents intelligents reproduisant le comportement humain suscite plus fortement l'imagination, ses applications utiles restent toutefois limitées. En

effet, l'humain est souvent émotif au détriment de son côté rationnel. Un tel comportement mène souvent à des choix non optimaux, si ce n'est à des choix carrément dommageables. Ainsi, nous pourrions plutôt souhaiter d'un agent intelligent qu'il pense ou agisse **rationnellement**, c'est-à-dire que les décisions qu'il prend sont optimales compte tenu des informations qui lui sont disponibles. Dans Russell et Norvig (2009), une distinction est faite entre le *penser* et l'*agir*.

Un agent qui *pense* rationnellement sera en mesure de construire et comprendre des syllogismes et donc de procéder à un raisonnement logique. Ces machines peuvent servir à résoudre des problèmes logiques, mais l'utopie pousse l'idée plus loin. À terme, un tel agent pourrait, lorsqu'on lui fournit un ensemble d'axiomes, déduire toutes les implications qui en découlent. En d'autres mots, il pourrait démontrer tous les théorèmes possibles. Andrew Wiles (Singh, 1998) aurait ainsi pu gagner dix ans de sa vie lorsqu'il s'est appliqué à démontrer le théorème de Fermat-Wiles : dans un monde idéal, une machine l'aurait simplement déduit des axiomes de base.

Un agent qui *agit* rationnellement sera quant à lui en mesure de poser ou de suggérer des gestes concrets et d'interagir avec son environnement d'une manière rationnelle. Quelques exemples mettent en lumière cette idée : pensons notamment aux voitures qui se conduisent d'elles-mêmes, aux logiciels de reconnaissance vocale, à l'ordinateur Deep Blue qui a battu Gary Kasparov aux échecs en 1997, au combat contre le pourriel grâce aux algorithmes évolutifs ou encore à la traduction automatisée. Dans tous les cas, ce sont des machines qui posent des gestes rationnels. Dans le domaine de la robotique, des robots autonomes sont développés afin d'accomplir des tâches qui sont impossibles à réaliser par des humains, ou encore tout simplement pour leur faciliter la vie. C'est le cas, par exemple, des robots envoyés sur Mars, qui sont difficiles à téléopérer en raison de la grande distance et qui doivent donc posséder une certaine autonomie.

Dans le cadre de ce mémoire, nous nous concentrons sur une branche de l'intelligence artificielle où l'on doit sélectionner et ordonner une série d'actions qu'accomplira un agent : il s'agit du domaine de la planification automatisée. Dans ce domaine, on fournit à un planificateur une description du monde dans lequel l'agent évolue, une liste d'actions qu'il peut accomplir ainsi qu'une série d'objectifs à satisfaire.Selon la situation initiale, le planificateur retourne un plan qui dictera la marche à suivre pour satisfaire tous les objectifs.Ce plan peut prendre diverses formes, la plus simple étant une séquence d'actions.

Fournir une description du monde et des actions possibles sous-entend de devoir modéliser la réalité. Or, modéliser parfaitement la réalité est une tâche ardue, voire pratiquement impossible, et c'est pourquoi la modélisation pour la planification dite classique est faite sous une série d'hypothèses qui rendent la modélisation et la résolution du problème plus simples. Par exemple, on suppose que le monde est totalement observable, qu'aucun événement extérieur ne viendra perturber l'agent lors de l'exécution du plan ou encore que les effets des actions sont appliqués instantanément (Ghallab *et al.*, 2004). Le chapitre 1 détaille en profondeur la planification automatisée sous le modèle classique.

Le chapitre 2 traite pour sa part des travaux qui ont été faits pour relaxer l'hypothèse suivante : en planification classique, l'ensemble des objectifs doit être satisfait.En effet, sous cette hypothèse, il suffit qu'un seul des objectifs ne soit pas satisfaisable pour que le planificateur ne retourne aucune solution. Cette hypothèse peut être un facteur contraignant lorsque les ressources et le temps sont limités et qu'il est difficile de pouvoir estimer le nombre d'objectifs qu'un agent pourra accomplir (Smith, 2004). De plus, satisfaire tous les objectifs peut se révéler impossible ou alors ne pas être bénéfique. Ce raffinement du problème est nommé planification à satisfaction partielle (PSP). Dans ce chapitre, nous détaillons donc les changements nécessaires au modèle ainsi que les diverses pistes de résolution

proposées dans la littérature.

Au chapitre 3, nous relaxons une autre hypothèse : celle qui dicte que les actions ont un effet instantané. Non seulement les effets ne sont plus immédiats, mais les durées des actions sont même soumises à une incertitude, modélisée par des lois de probabilité. Nous expliquons dans un premier temps en quoi les solutions proposées pour la résolution du problème déterministe de la PSP sont limitées dans ce nouveau contexte. Nous détaillons ensuite nos propres contributions au domaine, qui ont donné lieu à deux publications : (Labranche et Beaudry, 2014a) et (Labranche et Beaudry, 2014b). L'approche se base sur l'introduction d'une nouvelle action, qui permet d'abandonner des objectifs et, à l'aide de plans conditionnels, de choisir ceux qui sont satisfaits pendant l'exécution du plan. Ces contributions sont implémentées dans ActuPlan (*Action Contingency Time Uncertainty Planner*) (Beaudry *et al.*, 2010a,b).

En dernier lieu, nous traitons d'un domaine d'application pratique de la planification automatisée : les jeux vidéo. En effet, si traditionnellement les personnages non-joueurs (PNJ) étaient dirigés selon une suite de règles laborieuses à écrire (des *scripts*), les développeurs de jeux font de plus en plus appel à la planification automatisée pour créer plus facilement des PNJ (Orkin, 2004; Champandard, 2013). Cependant, la marge entre la théorie et la pratique étant encore assez grande, nous proposons un nouvel outil afin de la réduire. Il s'agit d'une utilisation novatrice de la planification automatisée visant à contrôler les PNJ. En effet, nous utilisons la PSP afin que le PNJ choisisse de manière autonome ses objectifs et le nombre de ceux-ci. Nous appliquons ces nouvelles méthodes de création de PNJ dans un jeu sérieux de simulation immobilière nommé *SimRealEstate*[1]. Le chapitre 4 présente le jeu ainsi que nos approches, présentés à la IEEE Conference on Computational

1. *SimRealEstate* était le nom à un stade antérieur du développement. Le jeu se nomme maintenant Game Of Homes, mais nous conservons l'appellation initiale dans ce mémoire.

Intelligence and Games (Labranche *et al.*, 2014).

[Cette page a été laissée intentionnellement blanche]

CHAPITRE I

PLANIFICATION AUTOMATISÉE

L'idée de la planification automatisée est de pouvoir contrôler des agents à l'aide d'objectifs, au lieu de manuellement décider de leurs actions (Ghallab *et al.*, 2004). Par exemple, il est désirable, au lieu de téléopérer un robot pour accomplir certaines tâches, de pouvoir simplement lui spécifier quelles tâches nous aimerions voir accomplies et de le laisser lui-même décider de la marche à suivre.

Ainsi, l'objectif de la planification est de fournir un plan d'action à un agent. Afin d'y arriver, nous devons fournir à un planificateur une description de la réalité la plus fidèle possible. Cette description consiste en un modèle du monde dans lequel l'agent agira, des actions qu'il peut accomplir, des objectifs voulus ainsi que de la situation initiale. Le plan généré par le planificateur, souvent une suite d'actions partiellement ordonnée, permettra ensuite à l'agent d'accomplir les objectifs voulus à partir de la situation initiale.

Ce chapitre explique les grandes lignes de la planification automatisée, en commençant par décrire le concept d'agent intelligent (section 1.1). Nous y voyons comment modéliser la réalité et représenter ce modèle dans un langage compréhensible pour le planificateur aux sections 1.2 et 1.3. Puis, nous montrerons comment résoudre un problème de planification (section 1.4). Ce chapitre s'inspire en grande partie du livre de Ghallab *et al.* (2004).

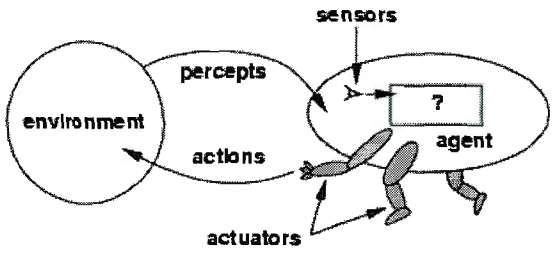


Figure 1.1: Structure d'un agent intelligent (Source : (Russell et Norvig, 2009))

1.1 Agent intelligent

Suivant la définition fournie dans Russell et Norvig (2009), un agent constitue tout ce qui peut percevoir son environnement à l'aide de senseurs et agir à l'aide d'actuateurs. Cette idée est illustrée à la figure 1.1.

L'«intelligence» — ou la rationalité — d'un agent dépend d'une fonction :

$$f : P^* \rightarrow \mathcal{A}$$

où P^* est l'ensemble des perceptions possibles et $\mathcal{A}$ l'ensemble des actions que l'agent peut exécuter. Cette fonction f est de toute évidence très difficile à définir et constitue en quelque sorte le noyau de l'intelligence artificielle. La rationalité d'un agent dépend de quatre critères : une mesure de performance, sa connaissance préalable de l'environnement, les actions possibles et la séquence de perceptions. Russell et Norvig (2009, p.37, traduction libre) donnent la définition suivante d'un agent rationnel (ou intelligent) :

Définition 1.1.1. *Pour chaque séquence de perceptions possibles, un* ***agent rationnel*** *choisit une action dont le résultat espéré maximise la mesure de performance, en tenant compte des informations fournies par la séquence de perceptions et de sa connaissance préalable de l'environnement.*

La planification est en quelque sorte l'intelligence d'un agent automatisé. C'est à l'étape de la planification que la marche optimale à suivre est définie.

1.2 Modèle classique

De manière conceptuelle, la planification cherche à déterminer un plan qu'un agent pourra exécuter. Le modèle de la réalité utilisé pour trouver ce plan est appelé un **système de transition d'états**.

Définition 1.2.1. *Un* ***système de transition d'états*** *est un quadruplet*

$\Sigma = (\mathcal{S}, \mathcal{A}, \mathcal{E}, \gamma)$, *où :*

- $\mathcal{S} = \{s_1, ..., s_n\}$ *est un ensemble fini d'états ;*
- $\mathcal{A} = \{a_1, ..., a_n\}$ *est un ensemble fini d'actions ;*
- $\mathcal{E} = \{e_1, ..., e_n\}$ *est un ensemble fini d'événements et*
- $\gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{E} \longrightarrow 2^{\mathcal{S}}$ *est une fonction de transition d'états.*

Typiquement, nous pouvons représenter un tel système par un graphe orienté, où les sommets correspondent aux états et les arcs aux actions ou aux événements. Dans un tel graphe, un arc existe entre s et s' si $s' \in \gamma(s, u)$, où u est un couple $(a, e) \in \mathcal{A} \times \mathcal{E}$. L'événement neutre ϵ et l'action neutre *no-op* sont ajoutés pour dénoter respectivement les transitions dues strictement à une action et celles dues strictement à un événement. Un exemple détaillé d'un tel graphe est fourni à la section 1.4.

La figure 1.2 montre le schéma de la planification : on fournit en entrée à un planificateur un domaine, un but ainsi qu'un état initial et le planificateur nous retourne un plan. Or, déterminer le domaine, en d'autres mots modéliser la réalité, est une tâche ardue. Aussi, plus le modèle est fidèle à la réalité, plus les problèmes basés sur celui-ci sont difficiles à résoudre.

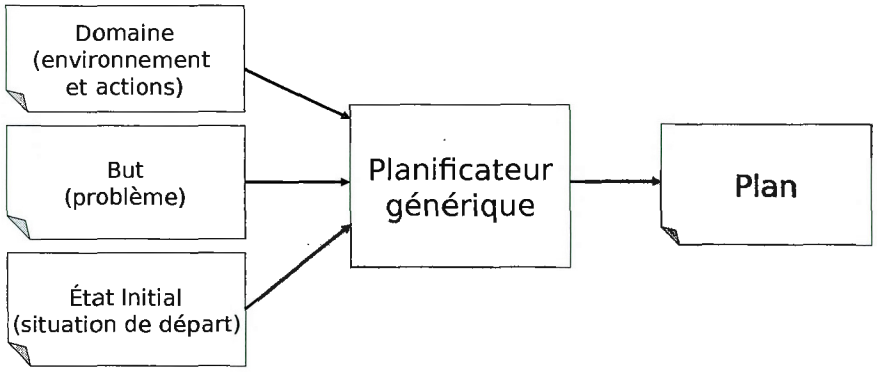


Figure 1.2: Schéma de la planification

Afin de simplifier la modélisation et la résolution des problèmes associés, le **modèle classique** — ou **modèle restreint** — a été proposé. Ce modèle se base sur huit hypothèses simplificatrices. Il n'est pas censé être directement utilisable pour résoudre des problèmes de la vie réelle, mais doit plutôt servir de référent de base pour l'écriture de modèles plus complexes. Ces huit hypothèses sont les suivantes :

H1 **Le système Σ possède un nombre fini d'états.**

H2 **Le système Σ est totalement observable** : La connaissance de l'état courant est toujours totale.

H3 **Le système Σ est déterministe** : Pour tout état s et pour chaque événement ou action u, $|\gamma(s,u)| \leq 1$. En d'autres mots, si une action est applicable dans un état s, elle mène invariablement toujours au même état s', dicté par γ.

H4 **Le système Σ est statique** : Aucun événement extérieur ne vient modifier l'état courant. Ainsi, l'ensemble $\mathcal{E}$ contient seulement l'événement neutre ϵ.

H5 **Le système Σ possède un ensemble d'objectifs restreints et indissociables** : Les objectifs complexes, comme certains états à éviter ou la

poursuite d'une trajectoire de résolution particulière, ne peuvent être traités : on ne considère comme objectifs que les affectations à des variables d'état. De plus, les objectifs sont indissociables : un état final doit satisfaire le but (tous les objectifs) en entier. Si un tel état n'existe pas, le planificateur retourne une erreur. Sous cette hypothèse, les objectifs sont dits *durs*.

H6 **Les plans sont séquentiels** : Un plan est une séquence finie d'actions ordonnées.

H7 **Le temps est implicite** : Les effets des actions sont appliqués immédiatement.

H8 **La planification se fait hors-ligne** : Un plan est d'abord calculé, puis exécuté. Le planificateur ne peut modifier un plan pendant qu'il est exécuté.

Les plans déterminés sous ces hypothèses très restrictives sont évidemment difficilement applicables à des situations réelles. C'est pourquoi, dans le cadre de ce mémoire, nous nous appliquons à trouver des solutions pour relaxer au maximum ces hypothèses et ainsi obtenir des modèles plus complexes et des solutions plus applicables.Dans le chapitre 2, c'est l'hypothèse **H5** qui est en partie relaxée. Nous expliquons en effet les changements nécessaires à apporter pour rendre les objectifs dissociables l'un de l'autre. Le chapitre 3 traite pour sa part de la relaxation supplémentaire des hypothèses **H3**, **H6** et **H7**, pour lesquelles des plans à branchements (donc des plans non séquentiels) sont nécessaires pour tenir compte d'une incertitude sur la durée des actions.

Nous commençons toutefois par expliquer comment représenter et résoudre un problème de planification classique et donc assujetti au modèle restreint.

1.3 Représentation

Si le modèle classique fournit des pistes pour la modélisation, il faut tout de même traduire ce modèle dans un langage manipulable. Typiquement, il existe plusieurs manières de représenter un modèle de planification, dont celle basée sur la théorie des ensembles, celle dite classique et celle à variables d'état. Nous détaillons et utilisons un hybride entre les variables d'état et la représentation classique à l'aide d'un exemple provenant du domaine Transport.

1.3.1 Exemple : le domaine Transport

Afin d'évaluer la performance des planificateurs, la *International Conference on Planning and Scheduling* (ICAPS) tient tous les deux ans la *International Planning Competition*(IPC). Des modèles prédéterminés (des domaines) sont utilisés comme références afin d'uniformiser l'évaluation, et le domaine Transport en fait partie (Gerevini *et al.*, 2009).

Ce domaine est donné par un ensemble de lieux, un ensemble de camions et un ensemble de boîtes. Un problème est posé par les liens existant entre les lieux (les routes), les positions initiales des camions et des boîtes ainsi que les lieux de livraison des boîtes (les objectifs). La figure 1.3 montre une instance d'un tel problème, que nous utilisons pour illustrer la représentation.

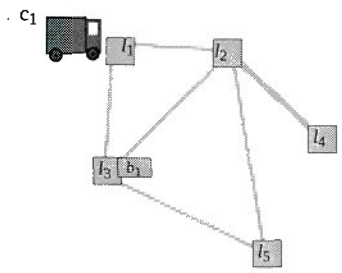


Figure 1.3: État d'un problème du domaine Transport

Dans ce cas précis, nous devons représenter les éléments suivants :

- Objets : Les lieux $L = \{l_1, l_2, l_3, l_4, l_5\}$, la boîte $B = \{b_1\}$ et le camion $C = \{c_1\}$.
- Variables du monde : Ce sont les affectations à certaines variables constantes, ici les différents liens entre les lieux.
- Variables d'état : Ce sont les positions des objets mobiles $X = \{P_{c_1}, P_{b_1}\}$ où $P_{c_1} \in L$ et $P_{b_1} \in (L \cup C)$. Ainsi, une boîte peut être soit à un lieu, soit sur le camion.
- But : $\mathcal{G} = \{P_{b_1} = l_5\} \subset \mathcal{S}$. Le but est donc l'ensemble d'états où la variable P_{b_1} a pour valeur l_5.

Un état est donné par l'assignation d'une valeur à toutes les variables d'état. Dans notre exemple, nous avons pour état initial : $s_0 = (P_{c_1} = l_1, P_{b_1} = l_3) \in \mathcal{S}$. Cet état se trouve dans l'espace d'états $\mathcal{S}$ (tous les états possibles) donné par le produit cartésien : $\mathcal{S} = \otimes_X = L \times (L \cup C)$. Ces états incluent aussi implicitement les assignations fixes aux variables dites du monde (qui ne peuvent être modifiées par une action). Ici, les liens entre les lieux en font partie.

De manière plus générale, un problème de planification est défini par un quadruplet :

$$\mathcal{P} = \langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$$

où $\mathcal{F}$ est un ensemble fini de variables (du monde et d'état), $\mathcal{I}$ un état initial, $\mathcal{A}$ un ensemble fini d'actions et $\mathcal{G} = \{g_1, g_2, .., g_n\}$, un but qui consiste en un ensemble fini d'objectifs.

1.3.2 Langage

Dans le domaine de la planification, il existe plusieurs langages. Dans ce mémoire, nous utilisons le langage de représentation APL, semblable au *Planning Domain Definition Language* (PDDL) , le langage officiellement utilisé dans la IPC (Gerevini *et al.*, 2009). Chacune des actions est spécifiée par ses préconditions, ses effets et ses paramètres. Les préconditions sont les conditions nécessaires avant l'application de l'action, les effets sont les nouvelles assignations aux variables d'état après l'application et les paramètres sont les variables figurant dans les préconditions ou dans les effets.

La figure 1.4 montre l'action `Conduire` pour le domaine Transport. Nous constatons que, pour appliquer l'action (les conditions), l'origine doit être différente de la destination, que le camion doit se situer à l'origine et qu'une route doit exister entre l'origine et la destination. Naturellement, l'effet de l'action est de déplacer le camion jusqu'à la destination.

Une fois les actions définies, il faut spécifier un problème à résoudre. La figure 1.5 illustre une partie de notre exemple. Les objets du problème sont d'abord définis, puis l'état du monde est donné en assignant les routes et les distances correspondantes. Les variables d'états sont instanciées aux valeurs qui correspondent à l'état initial. Il est à noter que *indéfini* ne signifie pas qu'il y a une incertitude pour savoir la boîte b_1 est sur quel camion, mais plutôt qu'elle n'est présentement sur aucun camion. Finalement, le but est donné.

1.4 Algorithme de planification

Typiquement, la résolution d'un problème de planification se fait de manière itérative et plusieurs stratégies sont possibles, dont les deux suivantes : l'exploration de l'espace d'états et l'exploration de l'espace des plans Ghallab *et al.* (2004).

```
action Conduire(Camion c, Lieu origine, Lieu destination){
   conditions:
     @debut: origine != destination;
     @debut: PositionCamion(c) = origine;
     @debut: existeRoute(origine, destination);
   effets:
     @fin: PositionCamion(c) = dest;
 }

action Charger(Camion c, Lieu p, Objet o){
   conditions:
     @debut: PositionCamion(c) = p;
     @debut: PositionObjet(o) = p;
   effets:
     @fin: SurQuelCamion(o) = p;
}

action Décharger(Camion c, Lieu p, Objet o){
   conditions:
     @debut: PositionCamion(c) = p;
     @debut: SurQuelCamion(o) = c;
   effets:
     @fin: SurQuelCamion(o) = indéfini
     @fin: PositionObjet(o) = p;
 }
```

Figure 1.4: Spécification des actions pour le domaine Transport

```
Probleme {
  Objets {
    Lieu l1,l2,l3,l4,l5;
    Objet b1;
    Camion c1;
  }
  EtatMonde {
     Route(l1,l2);
     Route(l1,l3);
     ...
     Distance(l1,l2)=80;
     Distance(l1,l3)=70;
     ...
  }
  EtatInitial{
     PositionCamion(c1)=l1;
     PositionObjet(b1)=l3;
     SurQuelCamion(b1)=indéfini;
  }
  But{
     PositionObjet(b1)==l5;
  }
}
```

Figure 1.5: Spécification APL du problème du domaine Transport correspondant à la figure 1.3

Nous nous concentrons ici sur l'espace d'états.

L'idée est de rechercher un chemin dans le graphe qui représente un espace d'états. L'approche est d'appliquer, à chaque état, chaque action possible de manière non déterministe, et ce, jusqu'à ce qu'un état final (un état où tous les objectifs sont satisfaits) soit atteint. La complexité du problème augmente donc exponentiellement par rapport au nombre moyen d'actions applicables dans chaque état, appelé le facteur de branchement. Lorsqu'un état non final déjà visité est revisité ou qu'aucune action n'est applicable, l'algorithme de planification retourne en arrière pour continuer à explorer à partir d'un autre état. Lorsqu'un état final est atteint, il suffit de suivre le chemin en ordre inverse pour retrouver les actions nécessaires à la composition du plan.

Il faut cependant s'assurer que cette exploration mène à l'atteinte la moins coûteuse possible (en planification classique : celle qui nécessite le moins d'actions) d'un état final. Plusieurs algorithmes existent, mais le plus populaire et le plus connu est A^* (prononcé *A étoile*) (Hart *et al.*, 1968). L'idée est d'explorer en priorité à partir de l'état dont le coût total final estimé est le plus faible. Ce coût final est donné par $f = g + h$ où g est le coût pour se rendre à cet état et h une estimation du coût restant avant d'atteindre un état final. Cette estimation se fait à l'aide de fonctions heuristiques.

La figure 1.6 montre un extrait d'une exploration de l'espace d'états, où l'exemple de la section précédente est repris.

Les flèches indiquent les états qu'il est possible d'atteindre (les actions applicables) et celles en pointillé donnent la suite d'actions optimale pour la composition du plan.

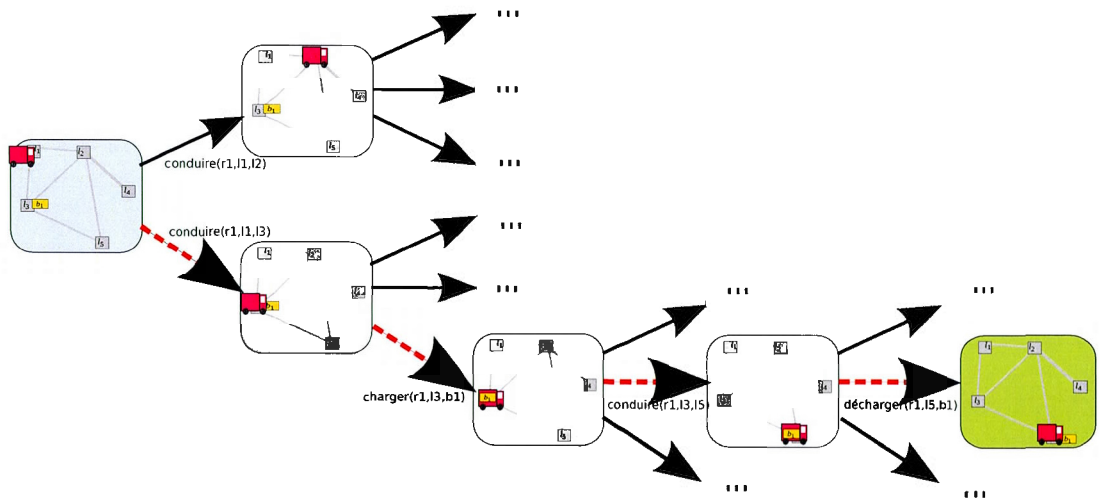


Figure 1.6: Exploration de l'espace d'états

[Cette page a été laissée intentionnellement blanche]

CHAPITRE II

PLANIFICATION À SATISFACTION PARTIELLE

La condition que tous les objectifs doivent être satisfaits dans un état final peut être parfois contraignante. En effet, dans certaines situations où le temps, les ressources et les connaissances sont limités, il est difficile d'estimer avec justesse le nombre d'objectifs qu'il sera possible d'accomplir. C'est notamment le cas pour les robots que l'on envoie sur Mars (Smith, 2004). Ces expéditions, très coûteuses, doivent être rentabilisées et, puisque la téléopération n'est pas possible (une instruction envoyée de la Terre prend en moyenne plus de 10 minutes pour être reçue sur Mars), des plans fiables sont nécessaires. Il a été estimé que les échecs de plans ont été la cause pour laquelle le robot Sojourner a été inactif de 40% à 75% du temps lors de la mission Pathfinder (Bresina *et al.*, 2002). Aussi, satisfaire tous les objectifs peut se révéler tout simplement impossible ou alors ne pas être bénéfique en termes de coûts. Ainsi, il est pertinent de développer des outils de planification permettant aussi de sélectionner un sous-ensemble réalisable et bénéfique d'objectifs parmi tous ceux demandés. C'est ce que la planification à satisfaction partielle (PSP) propose de faire.

Le problème de la PSP est double : il faut sélectionner le meilleur sous-ensemble possible d'objectifs et planifier pour ces objectifs. Comme illustré à la section 2.2, l'approche naïve (résoudre le problème pour tous les sous-ensembles d'objectifs

et choisir le meilleur) n'est pas réaliste. Historiquement, les premières approches se sont penchées sur des manières de choisir les objectifs d'abord (c'est-à-dire de trouver le meilleur sous-ensemble), puis de planifier. Les sections 2.4 et 2.5 rendent compte des deux pistes principales explorées dans cette optique. Cependant, nous voyons à la section 2.6 qu'il est possible de réduire le problème de la PSP à la planification classique. Nous terminons en discutant à la section 2.7 de la complexité algorithmique du problème.

2.1 Modélisation par bénéfice net

Nous complexifions le problème de la planification en donnant la définition suivante de la planification à satisfaction partielle. Un problème de PSP est un quadruplet :

$$\mathcal{P} = \langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$$

selon la même définition qu'à la section 1.3.1. Cependant, nous ajoutons que chaque action $a \in \mathcal{A}$ a un coût correspondant à un nombre constant d'unités quelconques et que les objectifs de $\mathcal{G}$ peuvent être mous ou durs. Il n'est pas nécessaire de satisfaire un objectif mou dans un état final alors qu'un objectif dur doit être satisfait. La qualité, ou la valeur, d'un plan est donnée selon un bénéfice net (somme des utilités des objectifs satisfaits moins le coût total des actions nécessaires). Certaines approches ont tenté d'accorder des valeurs qualitatives aux objectifs (Bienvenu *et al.*, 2006), mais ce sont les approches par valeurs quantitatives (utilités) qui dominent (Smith, 2004; Briel *et al.*, 2004; Nigenda et Kambhampati, 2005; Meuleau *et al.*, 2006; Keyder et Geffner, 2009; Coles, 2012). Avec des utilités, le problème se réduit à celui de maximiser le bénéfice net. Ainsi, à chaque objectif est associée une utilité.

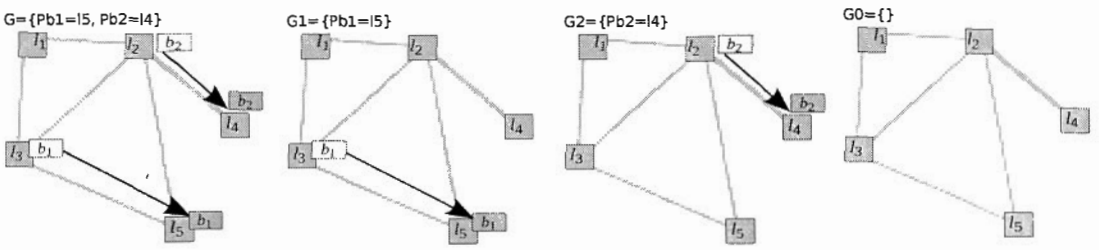


Figure 2.1: Approche naïve : le problème est décomposé et résolu selon tous les ensembles possibles d'objectifs

Il peut être difficile de déterminer avec précision l'utilité d'un objectif, une valeur quantitative exprimant à quel point tel objectif est plus profitable qu'un autre. Cette considération relève toutefois de la modélisation du problème et ne concerne pas directement les travaux présentés dans ce mémoire. Des recherches pour attribuer automatiquement des utilités ont été menées et sont disponibles notamment dans (Roijers *et al.*, 2014).

2.2 Approche naïve

Intuitivement, nous pourrions être tentés de résoudre le problème de planification classique pour chacun des 2^n sous-ensembles formés à partir des n objectifs et de simplement choisir le meilleur. Or, puisque la complexité algorithmique de la résolution d'un seul problème est déjà assez élevée, il n'est évidemment pas réaliste de trouver le plan optimal pour chacun des 2^n sous-ensembles et de les comparer. La figure 2.1 rend compte de cette difficulté pour un problème où seulement deux objectifs sont considérés, mais l'explosion exponentielle du nombre de problèmes à résoudre est tout de même manifeste. Ainsi, pour seulement deux objectifs, il y aurait quatre problèmes à résoudre.

2.3 Approche par valeurs qualitatives

Il est possible d'attribuer une valeur qualitative aux buts. C'est ce qui est proposé dans Bienvenu *et al.* (2006), où un langage logique est développé pour permettre de formaliser les préférences et l'interdépendance entre les différents objectifs. Le problème devient alors de résoudre un n-tuplet composé d'un état initial, d'un but et d'une formule générale de préférence. À titre d'exemple, nous pouvons considérer les préférences associées au choix du souper d'une personne. Voici comment exprimer qu'une personne ne mange jamais de nourriture chinoise :

$$toujours(\neg((\exists(x).occurrence(manger(x)) \wedge chinois(x)))$$

Il est clair que la difficulté de modéliser adéquatement une situation croît et devient rapidement hors de contrôle, surtout en considérant qu'une formulation quantitative permettrait de simplement allouer une utilité numérique aux objectifs, ce qui aurait pour effet d'ordonner les préférences. Même si l'interdépendance entre objectifs est plus difficile à déceler, c'est cette simplicité de modélisation qui favorise l'utilisation de méthodes quantitatives.

2.4 Approches par estimation du bénéfice net

En estimant le bénéfice net rapporté par un ensemble d'objectifs, la sélection du meilleur sous-ensemble peut être accélérée. Par la suite, il suffit de résoudre le problème de planification associé à l'ensemble choisi.

Dans Nigenda et Kambhampati (2005) et Benton *et al.* (2009), des méthodes heuristiques pour cette estimation ont été proposées. Il faut noter que c'est un problème difficile parce qu'en général les objectifs sont interdépendants : la satisfaction de l'un influence celle de l'autre. Cette dépendance est traitée en construisant itérativement l'ensemble optimal à partir de l'ensemble vide, en rajoutant un

objectif à la fois. On détermine lequel est ajouté en identifiant à quels points les objectifs sont dépendants et en vérifiant quelles actions ils partagent (puisqu'elles leur sont à tous nécessaires pour être satisfaits). Celui qui est le plus bénéfique par rapport aux objectifs déjà présents est ajouté. Cependant, les algorithmes peuvent choisir des objectifs qui ne font pas partie d'un ensemble optimal et la détection de ces erreurs augmente de beaucoup le temps d'exécution.

Il est aussi possible, au lieu d'itérer à partir de l'ensemble vide, de débuter à partir de $\mathcal{G}$ au complet et de retirer itérativement des objectifs. C'est ce qui est proposé dans Briel *et al.* (2004), mais les mêmes critiques que pour l'approche précédente valent ici aussi.

2.5 Approches par hiérarchisation d'objectifs

Dans Smith (2004), le problème est abstrait en tant que problème d'orientation [1]. Dans cette abstraction, les villes sont les variables d'état qui créent naturellement des dépendances entre les objectifs ou qui sont à tout le moins importantes pour leur satisfaction. Par exemple, dans le cas d'un robot sur Mars, plusieurs expérimentations (prendre des photos, extraire un échantillon de sol, etc.) peuvent être menées à une même position. Les variables de position seraient donc les villes dans cette transposition du problème. L'ensemble de ces villes (ou attributs) constitue «l'ensemble de base». Il importe de noter que cet ensemble peut dégénérer et s'étendre à tout $\mathcal{F}$, ne résultant en aucune optimisation. Par la suite, la résolution du problème d'orientation pour cet ensemble de base fournit une heuristique accélérant la résolution du problème original. Cette abstraction est basée sur une décomposition hiérarchique des objectifs inhérente à la PSP, comme il a été montré dans Amir et Engelhardt (2003).

---

1. À titre d'indication, le problème du voyageur de commerce est une instance simplifiée d'un problème d'orientation.

Cette idée a par la suite été poussée plus loin dans les travaux de Meuleau *et al.* (2006, 2009) pour résoudre des problèmes où le résultat des actions est incertain.Les auteurs, plutôt que de créer une hiérarchie d'objectifs, proposent une hiérarchie de processus décisionnels de Markov (MDP). Ces processus peuvent par la suite être résolus individuellement et la concaténation des résultats donne une solution au problème en entier [2]. Cette décomposition en plusieurs petits problèmes permet de contourner la complexité exponentielle associée aux MDP, qui doivent expliciter tout (ou en très grande partie) l'espace d'états. Afin de contourner cette complexité, les auteurs utilisent des techniques de factorisation de MPD (Amir et Engelhardt, 2003), qui réduisent grandement la taille des politiques (des plans dans le contexte des MPD), mais qui ne garantissent l'optimalité que sous certaines hypothèses.

2.6 Réduction de la PSP en planification classique

Dans une note de recherche, il a été montré qu'un problème de PSP pouvait se traduire à un problème de planification classique (Keyder et Geffner, 2009). En d'autres mots, l'introduction d'objectifs mous (donc le problème de sélection d'un sous-ensemble optimal) n'ajoute aucune complexité algorithmique et peut être contournée : il suffit d'apporter quelques transformations au problème pour le rapporter à un problème de planification classique. Leurs résultats empiriques suggèrent une résolution plus rapide qu'avec les méthodes présentées aux sections précédentes. Cette section expose les grandes lignes de leur méthode.

L'idée est d'ajouter une nouvelle variable d'état nommée *mode-terminal* qui est fausse au départ et une action *activerModeTerminal* qui l'active. Toutes les actions du problème ont comme précondition que *mode-terminal* soit fausse et aucune

2. Les MDP sont un outil couramment utilisé et performant pour résoudre des problèmes de planification sous incertitude sur l'effet des actions. (Boutilier *et al.*, 1999)

action ne peut rendre *mode-terminal* fausse. Deux autres actions sont ajoutées, `Collecter` (*collect*) et `Oublier`(*forgo*), qui ont comme précondition que le mode terminal soit activé. `Collecter` peut être appliquée à un objectif satisfait et a un coût nul. `Oublier` peut être appliquée à un objectif non satisfait et a un coût équivalent à l'utilité de l'objectif. La planification est terminée lorsqu'une des deux actions a été appliquée à chacun des objectifs. Le plan optimal est celui qui a le plus petit coût total.

En d'autres mots, le planificateur tente d'arrêter la planification à chaque nouvel état exploré. Il vérifie ensuite quels objectifs ont été accomplis et calcule le coût total du plan associé en considérant qu'abandonner un objectif revient en quelque sorte à devoir payer son utilité.

Une limitation de cette approche, que nous approfondissons au chapitre suivant, est qu'elle n'offre aucune indication de *quand* un objectif a été abandonné. En effet, lorsque le plan est construit, nous savons seulement quels objectifs seront satisfaits et lesquels ne le seront pas.

2.7 Complexité du problème

Les résultats de cette section proviennent tous de l'article (Aghighi et Jonsson, 2014).

D'abord, mentionnons que deux formulations du problème de la satisfaction partielle peuvent être considérées : (1) satisfaire un maximum d'objectifs ou (2) satisfaire le sous-ensemble rapportant le plus grand bénéfice net (somme des utilités rapportées moins le coût total des actions). Si nous nous sommes jusqu'ici implicitement concentrés sur le problème du bénéfice net, c'est que la version du plus grand sous-ensemble d'objectifs peut être réduite en temps polynomial à un problème de bénéfice net. Il suffit en effet de mettre le coût des actions à 0 et

l'utilité de chacun des objectifs à 1. Ainsi, la complexité algorithmique des deux problèmes est la même.

En planification classique, il s'agit normalement de déterminer si oui ou non un plan qui satisfait tous les buts existe et, si oui, quel est ce plan. Il existe un raffinement du problème nommé ***Bounded Cost Plan Existence*** (BCPE), où les actions ont des coûts explicites et où il s'agit de déterminer si un plan existe dont le coût total des actions est plus petit qu'un entier K. Il est d'abord montré que si BCPE est dans NP [3], alors PSP est aussi dans NP. Ensuite, en montrant que BCPE est dans NP, les auteurs concluent que PSP est lui-même dans NP.

La nature des actions influence toutefois la complexité du problème. Le tableau 2.1 montre les différents résultats obtenus selon le nombre maximal de préconditions et de postconditions (effets) des actions. La colonne ≥ 2 indique que le résultat tient lorsque le nombre est fixé et celle $*$ lorsqu'il ne l'est pas.

Tableau 2.1: Résultats de complexité pour le problème de PSP en fonction du nombre de préconditions et de postconditions par action

pre/post	1	≥ 2	*
0	P	NP-complet	NP-complet
1	NP-difficile	NP-difficile	PSPACE-complet
≥ 2	NP-difficile	PSPACE-complet	PSPACE-complet
*	PSPACE-complet	PSPACE-complet	PSPACE-complet

3. Brièvement, un problème est dans NP (pour *nondeterministic polynomial time*) si, pour une solution donnée, il est possible de prouver qu'elle est correcte en un temps polynomial. Un problème est NP-complet s'il est dans NP et qu'aucun algorithme connu ne permet de trouver une solution en un temps polynomial.

CHAPITRE III

PLANIFICATION À SATISFACTION PARTIELLE SOUS INCERTITUDE TEMPORELLE

Reprenons notre exemple des robots sur Mars, qui ont entre autres motivé le besoin de développer la planification à satisfaction partielle. Si le fait de pouvoir choisir un sous-ensemble optimal d'objectifs parmi un ensemble plus grand grâce à la PSP est déjà une amélioration considérable par rapport à la planification classique, il subsiste néanmoins quelques obstacles lorsque l'on désire modéliser adéquatement la réalité d'un robot sur Mars.

En effet, nous pouvons d'abord nous intéresser au temps de réalisation d'un objectif. Prenons par exemple une roche que le robot doit échantillonner. Il est fort possible que la fenêtre temporelle pour accomplir cet objectif ne soit pas infinie : une tempête météorologique pourrait empêcher le robot de s'y rendre ou même ensevelir la roche. Pour gérer ces situations, nous introduisons la notion de *temps limite* (*deadline*) sur l'accomplissement des objectifs. Cet ajout permettra d'associer à certains objectifs un temps limite, au-delà duquel ils ne pourront plus être accomplis.

De plus, il peut être très difficile d'estimer la durée de certaines actions. Par exemple, même si la distance entre deux lieux sur Mars peut être connue de manière assez précise, il devient plus difficile d'avoir une description parfaite du

terrain (ex : obstacles au sol, pentes, reliefs, etc.) et des conditions météorologiques (i.e. la force du vent), deux facteurs qui peuvent influencer la durée de déplacement. Afin de contourner ce problème, nous permettons aux actions d'avoir des durées incertaines, modélisées par des distributions probabilistes. Planifier en tenant compte de cette incertitude réduit les risques d'échec des plans puisqu'ils pourront s'adapter dynamiquement selon les durées effectives des actions.

Aussi, comme mentionné à la fin du chapitre 3, les méthodes actuelles qui traitent de la PSP n'offrent aucun moyen de savoir *quand* un objectif est abandonné. C'est normal, puisque précédemment, le problème n'a été traité que dans un cadre déterministe. Cependant, lorsque la durée des actions est incertaine, il n'est pas toujours possible de savoir à l'avance quels les objectifs pourront être satisfaits et lesquels devront être abandonnés. Ce que nous désirons obtenir comme information est donc le temps maximal à la suite duquel un objectif doit être abandonné parce que sa satisfaction est soit irréaliste (compte tenu de son temps limite) ou moins bénéfique que le coût attendu pour le satisfaire. C'est selon cette philosophie que nous articulons la méthode proposée.

Ainsi, nous désirons offrir la possibilité de résoudre un problème de PSP qui permet aussi de modéliser des temps limites sur les objectifs, une incertitude sur la durée des actions et qui abandonne un objectif seulement lorsqu'il n'est plus réaliste ou bénéfique de tenter de le satisfaire. Ce chapitre présente la méthode que nous avons développée dans le cadre du projet de maîtrise et qui permet de gérer ces nouvelles contraintes. Cette méthode a été présentée au programme principal de la conférence Canadian AI (Labranche et Beaudry, 2014a) et sous forme d'affiche à la conférence de l'Association for the Advancement of Artificial Intelligence (Labranche et Beaudry, 2014b).

Nous prenons d'abord le temps de formaliser le problème à la section 3.1, puis

nous présentons un court état de l'art en décrivant une approche dont l'idée de base se rapproche de la nôtre (section 3.2). Nous décrivons par la suite en détail la méthode par actions d'abandon à la section 3.3. La section 3.4 traite des expérimentations. Nous terminons en discutant (section 3.5) de méthodes alternatives à la PSP sous incertitude : la réparation de plans et la replanification.

3.1 Formalisation du problème

Nous reprenons la définition d'un problème de PSP du chapitre 2 en y ajoutant les contraintes suivantes. Un problème de PSP sous incertitude est :

$$\mathcal{P} = \langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$$

selon la même définition qu'à la section 2.1. Nous ajoutons que les objectifs peuvent avoir un temps limite, moment à partir duquel ils ne peuvent plus être satisfaits. De plus, les actions ont une durée incertaine modélisée par des distributions probabilistes. Le coût d'une action (et d'un plan) est sa durée.

Afin de gérer l'incertitude liée à a durée des actions, les plans ne peuvent plus simplement être un ensemble partiellement ordonné d'actions. Pour pallier ce problème, nous introduisons donc la notion de plans conditionnels. Les plans conditionnels comportent des branchements dont les choix d'une branche ou d'une autre dépendent du temps d'exécution du plan. Ces temps sont calculés selon un seuil de confiance α, qui représente la probabilité non nulle que le plan s'exécute correctement. En effet, puisque les objectifs ont des temps limites et que les actions ont des durées incertaines, il subsiste pratiquement toujours une probabilité que le plan échoue. L'intuition derrière un branchement conditionnel est qu'au branchement, un plan rapportant plus de bénéfice net pourrait être exécuté si l'exécution avant le branchement a été plus courte qu'un temps donné.Passé ce temps, il faut choisir

une branche moins bénéfique, mais qui respecte le seuil minimal de probabilité de réussite.

3.2 Incertitude quant à la consommation de ressources

Des travaux ont déjà été réalisés pour résoudre la PSP lorsque des ressources autres que le temps sont limitées et que leur rythme de consommation est incertain (Coles, 2012). L'auteure étend des idées d'une version antérieure d'ActuPlan Beaudry *et al.* (2010a) en déterminant d'abord un plan séquentiel et pessimiste, c'est-à-dire que pour chaque action on estime que la consommation de ressources sera plus grande que celle réellement espérée. Par la suite, chaque action est considérée comme un point de branchement conditionnel candidat et on calcule la quantité de ressources que l'action et celles subséquentes devraient idéalement consommer afin qu'il soit possible de satisfaire plus d'objectifs qu'avec le plan pessimiste original. S'il est effectivement possible d'accomplir au moins un objectif supplémentaire, l'action devient un point de branchement et le plan correspondant est ajouté. Les quantités de ressources restantes deviennent les conditions sur lesquelles les choix de branchement sont faits.

3.3 Notre approche par actions d'abandon

Nous présentons une approche qui étend les idées de Keyder et Geffner (2009) Beaudry *et al.* (2010a, 2012) et Coles (2012) pour créer des plans conditionnels où les points de branchement peuvent être des actions qui abandonnent des objectifs. Un plan conditionnel est un agrégat de plans dont le branchement s'effectue selon l'observation d'une variable d'état à un certain temps (Ghallab *et al.*, 2004). Cette méthode permet une gestion dynamique du sous-ensemble d'objectifs qui seront satisfaits, ce qui n'est pas possible dans un cadre déterministe. Si la méthode proposée dans Coles (2012) permet une telle gestion, il peut être désirable d'avoir

plus de contrôle sur la gestion du sous-ensemble d'objectifs choisis.

Afin de fournir ce contrôle, nous fournissons la possibilité d'établir, comme pour les objectifs, un temps limite sur les actions d'abandon. Cela permet de savoir avec certitude (à l'exception d'un possible échec du plan en rapport au seuil de tolérance) que, passé son temps limite d'abandon, un objectif ne pourra plus être abandonné et qu'il sera satisfait. En d'autres mots, il passera de mou à dur. Ce genre de contrôle est nécessaire lorsque la planification et l'exécution s'entrelacent. C'est le cas lorsque les plans sont très complexes et longs à obtenir et qu'il faut commencer à planifier les prochaines tâches pendant qu'un plan s'exécute. Ces situations surviennent par exemple pour les activités d'un robot sur Mars. Avec le temps limite sur l'abandon, les scientifiques auront une meilleure idée et un meilleur contrôle des objectifs qui seront satisfaits et pourront planifier les activités du lendemain alors que le robot s'exécute déjà.

La philosophie que nous adoptons est ainsi un peu à l'inverse de celle pessimiste proposée par Coles (2012) : notre plan conditionnel de départ tente de satisfaire tous les objectifs et certains commencent à être abandonnés seulement si cela n'est plus possible.

3.3.1 Actions d'abandon

En reprenant la formulation du problème de la section 3.1, nous introduisons les actions d'abandon de la manière suivante.

À l'ensemble des actions $\mathcal{A}$ est ajouté l'ensemble $\{Abandon(g_1), Abandon(g_2)..., Abandon(g_{|\mathcal{G}|}\}$ de taille $|\mathcal{G}|$ dans lequel chaque $Abandon(g_i)$ est une action avec précondition que g_i soit dans $\mathcal{G}$ et qui a pour effet de supprimer (ou de rendre vrai) g_i de $\mathcal{G}$. À chaque $Abandon(g_i)$ est associé un coût d'abandon, représentant à quel point il est coûteux de ne pas satisfaire l'objectif i. Le coût d'abandon d'un

objectif est tout simplement son utilité. Il est à noter que pour uniformiser les coûts des actions, les utilités doivent elles aussi être données en unités de temps. Un temps limite est également assigné à chaque $Abandon(g_i)$, indiquant le temps maximum auquel l'objectif i peut être abandonné. Ainsi, les états sont maintenant modélisés avec un ajout de $|\mathcal{G}|$ variables d'état, chacune indiquant si l'objectif g_i a été abandonné ou non. Comme en planification classique, un état final s_f est un état où chaque g_i de $\mathcal{G}$ est satisfait et un plan est une séquence partiellement ordonnée d'actions menant de $\mathcal{I}$ à s_f. Un planificateur cherchera à minimiser le coût total.

Cette approche permet naturellement de traiter des problèmes où il y a à la fois des objectifs durs et mous. Pour spécifier un objectif dur, il suffit de fixer le coût d'abandon à l'infini ($+\infty$). À l'inverse, si le coût est plus petit que l'infini, cet objectif est mou. Une fois que le temps limite d'une action d'abandon est passé, l'objectif correspondant devient dur et doit être satisfait. Cette approche est aussi une réduction des objectifs mous comme proposé dans Keyder et Geffner (2009) et signifie qu'un planificateur classique peut solutionner des problèmes de PSP en utilisant notre modélisation.

3.3.2 Équivalence

Nous montrons dans cette section que notre méthode solutionne effectivement le problème pour le même sous-ensemble optimal d'objectifs que la méthode de réduction de la PSP présentée à la section 2.6. Dans Keyder et Geffner (2009), les auteurs démontrent que les plans retournés avec leur compilation de la PSP sont équivalents à ceux retournés par les méthodes utilisant le bénéfice net. Pour montrer que notre méthode est équivalente à la leur, nous montrons qu'elle est équivalente à celle par bénéfice net. Puisque ces méthodes trouvent le sous-ensemble optimal d'objectifs, il suffit de montrer que l'approche par actions d'abandon per-

met de toujours trouver le même sous-ensemble. Cette preuve assure l'optimalité des plans non conditionnels utilisés aux branchements du plan conditionnel.

Proposition 3.3.1. *La méthode par actions d'abandon choisit le même sous-ensemble optimal d'objectifs que les méthodes par bénéfice net.*

Démonstration. Soit $\mathcal{G}_s$ le sous-ensemble optimal d'objectifs sélectionné par une méthode à bénéfice net, $\mathcal{A}_p$ l'ensemble des actions du plan nécessaires pour satisfaire $\mathcal{G}_s$, $\mathcal{U}$ les utilités des objectifs et $\mathcal{C}$, le coût des actions.

Si $\mathcal{G}_s$ est optimal, il s'en suit que $\sum_{i=1}^{|\mathcal{G}_s|} \mathcal{U}(g_i) - \sum_{i=1}^{|\mathcal{A}_p|} \mathcal{C}(a_i)$, le bénéfice net, est maximal. Le bénéfice est aussi ≥ 0 puisqu'il est plus profitable de ne satisfaire aucun objectif plutôt que de perdre des ressources à en satisfaire.

Soit $\mathcal{G}_a$ l'ensemble des objectifs non satisfaits par une méthode à bénéfice net. Nous considérons ces objectifs comme abandonnés et définissons les coûts d'abandon $coût(g_i)$ comme les mêmes que les utilités correspondantes dans $\mathcal{U}$. Il est à noter que $\mathcal{G}_a$ est donné par $\mathcal{G} \setminus \{\mathcal{G}_s\}$ (un objectif est soit satisfait, soit abandonné). Le coût total d'un plan retourné par la méthode à actions d'abandon est :

$$\begin{aligned} \sum_{g \in \mathcal{G}_a} coût(g) + \sum_{a \in \mathcal{A}_p} \mathcal{C}(a) &= \sum_{g \in \mathcal{G} \setminus \mathcal{G}_s} \mathcal{U}(g) + \sum_{a \in \mathcal{A}_p} \mathcal{C}(a) \\ &= \sum_{g \in \mathcal{G}} \mathcal{U}(g) - \sum_{g \in \mathcal{G}_s} \mathcal{U}(g) + \sum_{a \in \mathcal{A}_p} \mathcal{C}(a) \\ &= \sum_{g \in \mathcal{G}} \mathcal{U}(g) - (\sum_{g \in \mathcal{G}_s} \mathcal{U}(g) - \sum_{a \in \mathcal{A}_p} \mathcal{C}(a)) \end{aligned} \tag{3.1}$$

Puisque $\sum_{g \in \mathcal{G}_s} \mathcal{U}(g) - \sum_{a \in \mathcal{A}_p} \mathcal{C}(a) \geq 0$ et est maximal, il s'en suit que le coût total est minimal. Puisque notre méthode retourne le plan avec le coût minimal, les mêmes objectifs que par bénéfice net sont abandonnés et donc le même sous-ensemble

optimal est satisfait. □

3.3.3 Planificateur ActuPlan

Le planificateur dans lequel nous avons implémenté notre méthode est Actuplan Beaudry *et al.* (2010a,b, 2012).

ActuPlan (***A**ction **C**ontingency **T**ime **U**ncertainty **Plan**ner*) est un planificateur qui permet de gérer la contingence d'actions, c'est-à-dire des plans où plusieurs actions peuvent s'exécuter en même temps. Cette fonctionnalité fait aussi partie des complexifications possibles au modèle de la planification classique, mais dépasse le cadre de ce mémoire. Ici, il est assumé que les valeurs sont données en entrée. ActuPlan permet aussi de gérer les actions à durées incertaines de manière efficace (Beaudry *et al.*, 2012), ce qui a motivé notre choix.

La planification est faite en utilisant une recherche à chaînage avant (algorithme A^*) guidée par l'heuristique H_{max} (Haslum et Geffner, 2000). Afin de représenter le temps de manière continue, le planificateur utilise un réseau bayésien composé de variables de temps. Cette représentation permet de gérer le temps de manière indépendante de la représentation des états et du fait, de réduire grandement la complexité de l'espace d'états, en comparaison avec des planificateurs qui utilisent une représentation discrète du temps Mausam et Weld (2008).

Puisque les durées des actions sont incertaines et que les objectifs ont des temps limites, le planificateur retourne un plan avec une probabilité de succès ≤ 1. ActuPlan cherche des plans non conditionnels qui satisfont le but avec un certain seul de probabilité de succès (niveau de confiance α) (Beaudry *et al.*, 2010a). Ces plans peuvent être assemblés pour former des plans conditionnels basés sur le temps d'exécution. Plusieurs plans sont générés puis assemblés pour former un plan conditionnel avec différents points de branchement où un choix est fait selon

le temps à l'exécution (Beaudry *et al.*, 2012).

Avec notre méthode, ces choix incluent la possibilité d'abandonner un objectif dynamiquement, ce qui est impossible avec la plupart des approches qui traitent de la PSP, puisqu'elles utilisent un sous-ensemble statique d'objectifs (voir 3.2 pour l'autre méthode s'en rapprochant) .

3.3.4 Exemple

Afin d'illustrer la méthode, son application et les plans retournés, considérons l'exemple provenant du domaine Transport de la figure 3.1.

Un camion t, initialement à l_0, doit livrer quatre boîtes aux différentes positions l_1, l_2 et l_3. Ici, $\mathcal{G} = \{g_0 = BoiteA(b_0, l_1), g_1 = BoiteA(b_1, l_2), g_2 = BoiteA(b_2, l_3), g_3 = BoiteA(b_4, l_3)\}$ et $\mathcal{A} = \{Conduire(t, l_o, l_d), Charger(t, l, b), Decharger(t, l, b)\}$. À titre d'indication, il y a 38 différentes actions instanciées possibles ($|\mathcal{A}| = 38$). L'instanciation d'une action signifie l'allocation de valeurs à ses paramètres, par exemple $Decharger(B_1, l_1)$. *Conduire* a un coût donné par une distribution normale paramétrée par 1.2 fois la distance entre les deux positions, *Charger* et *Decharger* ont un coût avec probabilité uniforme d'entre 30 et 60 unités. Le camion peut transporter une seule boîte à la fois.

Ajoutons à l'ensemble $\mathcal{A}$ les actions $Abandon(g_0)$, $Abandon(g_1)$, $Abandon(g_2)$ et

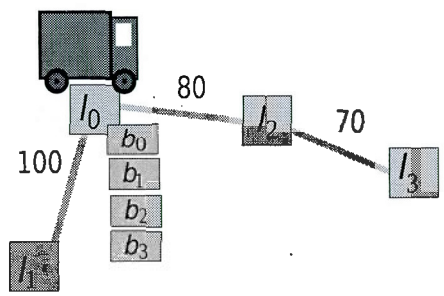


Figure 3.1: Exemple du domaine Transport

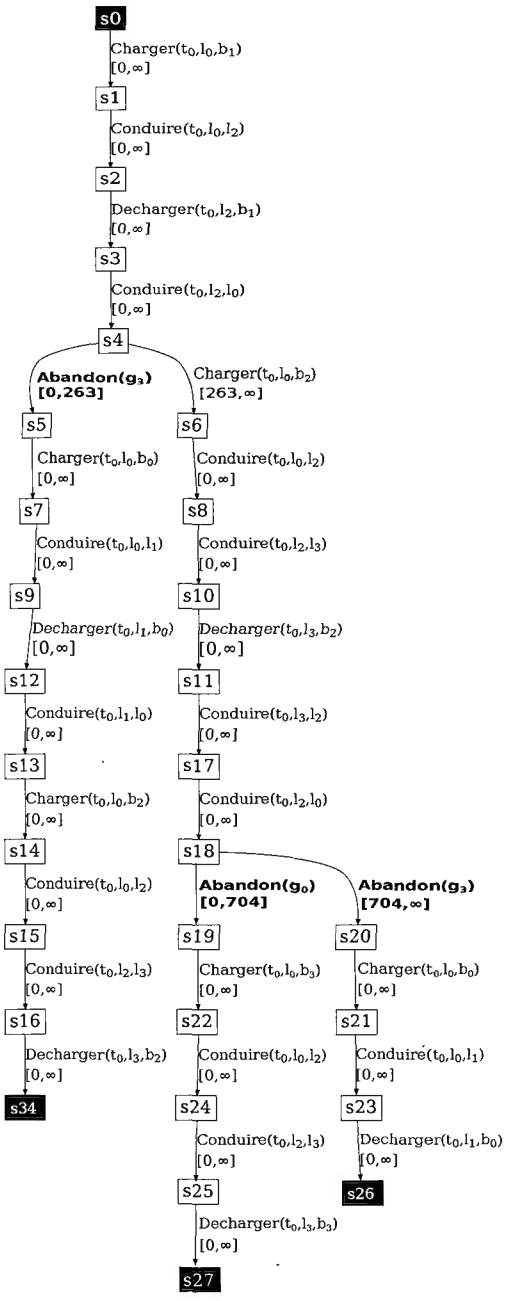
Abandon(g_3) avec coûts respectifs de 350, ∞, 600 et 500. Ainsi, les objectifs g_0, g_2 et g_3 sont mous tandis que g_1 est dur, car son coût d'abandon est ∞. Les objectifs ont aussi un temps limite pour leur satisfaction de, respectivement, ∞, 600, 900 et 1000. Un temps limite infini indique qu'un objectif peut être satisfait n'importe quand.

Deux problèmes seront comparés. Dans l'un, il n'y aura pas de temps limite sur les actions d'abandon, puis, dans l'autre, il faudra prendre la décision d'abandonner ou non l'objectif g_1 avant 500 unités de temps.

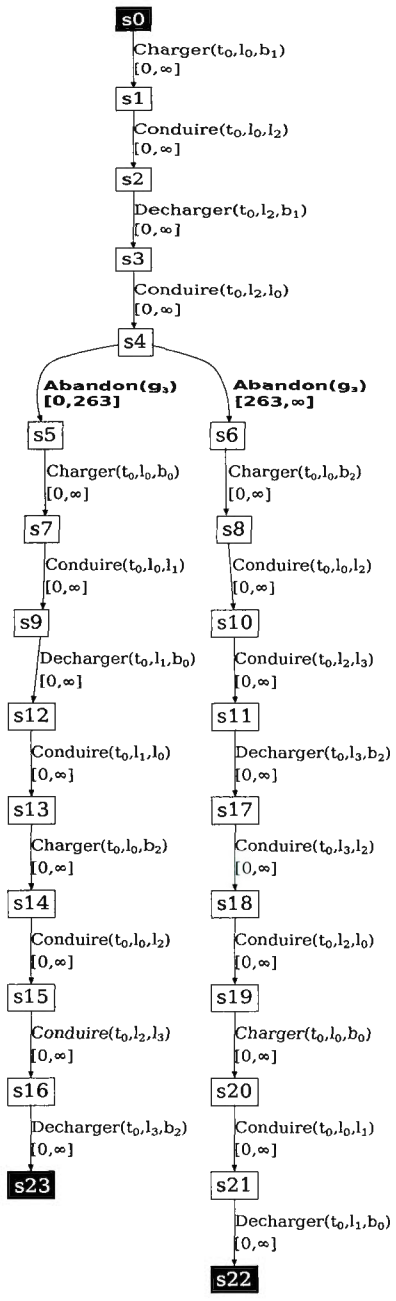
Les plans conditionnels résultants sont montrés à la figure 3.2. s_0 est l'état initial et les intervalles sous les actions représentent les conditions de branchement. À l'exécution, les actions sont choisies selon ces intervalles (Beaudry *et al.*, 2012).

Le plan sans temps limite sur l'abandon (figure 3.2(a)) est d'abord analysé pour montrer comment l'approche par actions d'abandon fonctionne. Puisque g_1 est un objectif dur, il doit être satisfait. Le plan dicte que la marche à suivre est de le satisfaire en premier (états s_0 à s_2) afin de respecter son temps limite. De l_2, le camion retourne à l_0 après avoir déchargé la boîte 1 (état s_3). Après l'état s_4, il y a un point de branchement : un choix est fait selon le temps d'exécution. Si g_2 a été satisfait assez rapidement (en moins de 263 unités), il est plus profitable d'abandonner g_3 et de satisfaire g_0 (états s_5 à s_9) puis g_2 (états s_{12} à s_{16}).

Si, cependant, le camion arrive à l'état s_4 après 263 unités, il est alors plus profitable de prendre la seconde branche. Comme pour la branche de gauche, il est possible de savoir qu'un objectif sera abandonné, mais pas encore lequel. Il y a en effet une fenêtre de temps pour laquelle il est plus profitable d'abandonner g_0 et de satisfaire g_3. Cette décision est prise après l'état s_{18} et sera prise selon que le camion se retrouve dans cet état avant 704 unités non.



(a) Sans temps limite sur l'abandon



(b) Avec temps limite sur l'abandon

Figure 3.2: Plans conditionnels générés

Il importe de noter que plus une branche est à gauche, plus elle est profitable (moins le coût total du plan est grand). Si ce n'était pas le cas, il ne servirait à rien de conditionner le plan sur le temps d'exécution, le branchement de gauche n'apportant aucun bénéfice. Seule la branche la plus à droite existerait : nous aurions simplement un plan séquentiel.

Considérons maintenant le plan de la figure 3.2(b). Un temps limite de 500 unités a été ajouté à l'action *Abandon*(g_0), signifiant que si l'objectif g_0 n'est pas satisfait avant, il devient dur et doit être satisfait à tout prix. En d'autres mots, ce temps limite indique que nous désirons savoir, à l'exécution, si g_0 sera satisfait ou non avant 500 unités. Ainsi, s'il n'est pas abandonné passé ce temps, nous avons l'assurance qu'il sera satisfait avant la fin de l'exécution du plan.

Les états s_0 à s_4 sont les mêmes, menant au branchement après s_4. Notons cependant qu'en comparaison du plan précédent, il y a une branche en moins et les deux dictent d'abandonner d'office g_3. Ainsi, alors que précédemment il y avait une fenêtre de temps dans laquelle il était possible de satisfaire g_3 au lieu de g_0, celle-ci n'est plus disponible. C'est normal, puisque le branchement s'effectuait à 704 unités, après quoi il est davantage profitable d'abandonner g_3. En d'autres mots, à 500 unités, il est impossible de déterminer si l'exécution du plan mènera à temps (avant 704 unités) à l'état s_{18} du plan de la figure 3.2(a). Nous sommes donc forcés d'abandonner d'office g_3 pour nous assurer de pouvoir fournir l'information sur la satisfaction de g_0 avant le temps limite fixé pour son action d'abandon.L'obtention de cette information a été coûteuse, puisqu'elle a forcé l'élimination d'une branche plus profitable que celle exécutée, augmentant le coût espéré total du plan 3.2(b) par rapport à 3.2(a).

À titre indicatif, le coût espéré d'un plan non conditionnel qui résout ce problème est de 1402,00 unités. Le plan conditionnel sans temps limite sur les ac-

tions d'abandon coûtera quant à lui 1396, 46 unités pour être exécuté. Lorsque nous ajoutons le temps limite sur l'action $Abandon(g_0)$, le coût espéré augmente légèrement pour atteindre 1396, 91. Cette très légère augmentation indique que la branche centrale du plan de la figure 3.2(a) a très peu de chance d'être empruntée et qu'elle n'apporte qu'une petite amélioration par rapport à la branche de droite. Ces tests représentent le coût moyen de 1000 instances de planification.

3.4 Expérimentations

Des tests empiriques ont été réalisés avec trois configurations différentes implémentées dans ActuPlan, toutes trois utilisant les actions d'abandon. Premièrement, nous utilisons un planificateur non conditionnel pour trouver un plan séquentiel ($ActuPlan^{NC}$ (Beaudry *et al.*, 2012)). Ensuite, deux planificateurs conditionnels sont testés, l'un pour lequel il n'y a pas de temps limites pour les actions d'abandon ($ActuPlan^{C}$ (Beaudry *et al.*, 2010a)) et l'autre pour lequel il y en a ($ActuPlan^{C+}$). Pour chacune des versions, nous imposons un seuil de probabilité de succès de $\alpha = 0.9$.

Nous utilisons les domaines Transport et Ascenseurs adaptés de la IPC de ICAPS (Gerevini *et al.*, 2009). Nous ajoutons un coût d'abandon pour chaque objectif. Le domaine Transport est sensiblement le même que celui utilisé pour notre exemple, sauf que les camions peuvent avoir un nombre illimité d'objets chargés. Dans le domaine Ascenseurs, des personnes désirent se rendre au bon étage en utilisant les ascenseurs à leur disposition. Les actions possibles sont de bouger les ascenseurs d'un étage à l'autre et de faire entrer ou sortir les personnes.

Ces tests ont été faits sur une machine possédant un processeur Intel QuadCore i5-3470 @ 3.20GHz. Chaque instance a été testée 100 fois. Les tableaux contiennent les temps machines moyens ainsi que les coûts espérés moyens. Le temps de planification et la mémoire étaient respectivement limités à 360 secondes et à 2 Go.

3.4.1 Résultats

Le tableau 3.1 contient les résultats pour le domaine Transport. Les colonnes sont $|C|$ le nombre de camions et $|\mathcal{G}|$ le nombre d'objets à livrer. Le nombre de lieux et les liens entre eux ont été gardés constants.

Tableau 3.1: Résultats pour le domaine Transport

Taille		Résultats					
$\|C\|$	$\|\mathcal{G}\|$	$ActuPlan^{NC}$		$ActuPlan^{C}$		$ActuPlan^{C+}$	
		Temps(s)	*Coût*	*Temps*(s)	*Coût*	*Temps*(s)	*Coût*
1	1	0,09	1525,0	0,08	1506,8	0,08	1506,6
1	2	0,74	1975,0	0,68	1914,6	0,86	1914,3
1	3	22,80	2175,0	23,79	2114,5	24,22	2114,3
2	1	0,40	1525,0	0,40	1506,9	0,41	1506,3
2	2	6,76	1600,0	6,59	1506,3	8,00	1507,0
2	3	48,76	1510,3	50,13	1506,6	56,33	1506,5

Tableau 3.2: Résultats pour le domaine Ascenseurs

Taille		Résultats					
$\|C\|$	$\|\mathcal{G}\|$	$ActuPlan^{NC}$		$ActuPlan^{C}$		$ActuPlan^{C+}$	
		Temps(s)	*Coût*	*Temps*(s)	*Coût*	*Temps*(s)	*Coût*
1	1	0,60	83,0	0,06	81,5	0,06	81,5
1	2	0,14	92,0	0,13	90,5	0,16	90,5
1	3	99,25	192,0	118,60	190,5	154,57	190,5
2	1	0,20	68,0	0,13	65,5	0,13	65,5
2	2	3,64	83,0	4,30	76,1	5,56	76,1
2	3	> 360	–	> 360	–	> 360	–

Le tableau 3.2 présente les résultats pour le domaine Ascenseurs. Les colonnes sont $|A|$ le nombre d'ascenseurs et $|\mathcal{G}|$ le nombre de personnes à amener au bon étage. Le nombre d'étages a été gardé constant.

Il était attendu que les plans non conditionnels auraient un coût espéré plus grand que les plans conditionnels. Les branchements conditionnels servent justement à

ce que l'agent puisse s'adapter si les actions s'exécutent rapidement, chose impossible avec un plan strictement séquentiel. Ce comportement est confirmé par les résultats empiriques, les coûts de plans non conditionnels étant toujours plus grands.

Cependant, cette optimisation vient à un prix et un plan conditionnel devrait être plus long à générer qu'un plan séquentiel. Cela est normal puisque, pour trouver un plan conditionnel, il faut trouver plusieurs plans non conditionnels. Selon le seuil de probabilité de succès, le nombre de temps limites sur les objectifs et la grandeur du but, une grande quantité de plans séquentiels peut être trouvée avant que la fusion en un plan conditionnel n'ait lieu. Ce comportement n'est pas retrouvé pour les problèmes plus petits, alors que les temps d'exécution sont sensiblement les mêmes pour les trois versions. Puisque les problèmes sont petits, il peut ne pas y avoir de point de branchement et le plan conditionnel est identique à celui non conditionnel. À l'inverse, dans les deux domaines, plus le problème se complexifie, plus il faut un temps significativement plus grand pour obtenir un plan conditionnel.

Nous nous attendions à ce que les temps de calcul pour $ActuPlan^{C+}$ soient plus grands que pour $ActuPlan^{C}$. Ajouter les temps limites sur les actions d'abandon rend le problème plus complexe, mais en contrepartie, on en tire plus d'informations. Cette complexification ne semble pas influencer les temps de calcul pour les plus petits problèmes, pour les mêmes raisons que dans le cas de la comparaison entre $ActuPlan^{NC}$ et $ActuPlan^{C}$, mais les résultats pour les plus gros problèmes montrent bien qu'il y a une influence.

Dans certains cas, les plans d'$ActuPlan^{C+}$ devraient avoir un coût espéré plus élevé, parce qu'un objectif mou non profitable peut devenir dur à cause du temps limite sur l'abandon. Les ressources sacrifiées pour le satisfaire auraient pu être

investies pour satisfaire un objectif plus profitable, comme il a été montré dans l'exemple de la section 3.3.4. Il importe de préciser que c'est le comportement voulu. Les temps limites sur l'abandon sont ajoutés pour s'assurer qu'un objectif soit satisfait s'il n'a pas été abandonné avant un certain temps. Les résultats montrent que les coûts espérés sont équivalents pour $ActuPlan^{C+}$ et $ActuPlan^{C}$. Cela indique que les temps limites n'ont pas forcé un objectif non profitable à être satisfait. Ces résultats restent tout de même intéressants, parce qu'ils démontrent qu'il est possible d'obtenir de l'information sur les temps maximaux d'abandon des objectifs sans nécessairement générer des coûts supplémentaires à l'exécution.

En général, les temps de planification deviennent rapidement assez grands en comparaison de ceux relatifs à des problèmes de taille semblable dans un monde déterministe. C'est pourquoi les tests sont limités à des problèmes plutôt petits. L'approche par actions d'abandon cause une explosion de la taille de l'espace d'états. Chaque état est modélisé avec $|\mathcal{G}|$ nouveaux prédicats pour garder la trace des objectifs qui ont été abandonnés. Le facteur de branchement est augmenté de $|\mathcal{G}|$ car à chaque objectif correspond une action d'abandon. Heureusement, avec les temps limites sur l'abandon, ce facteur diminue au fur et à mesure que les temps sont dépassés.

Des travaux futurs sur ce problème incluent l'exploration de l'extensibilité de cette approche à des problèmes plus larges. Afin d'y parvenir, des heuristiques adaptées à cette approche devront être développées.

3.5 Réparation de plans et replanification

Il est à noter que l'approche que nous proposons n'est pas la seule possible. La génération de plans conditionnels prenant en compte divers scénarios est un processus coûteux en temps et en mémoire. Pour éviter cette complexité, on peut simplifier le problème, c'est-à-dire ignorer l'incertitude, et replanifier (ou réparer

le plan) au besoin (Nebel et Koehler, 1992; Cushing et Kambhampati, 2005; Krogt et Weerdt, 2005; Fox *et al.*, 2006; Talamadupula *et al.*, 2013). Il semble donc avantageux d'utiliser simplement un plan séquentiel qui sera modifié ou refait en entier si jamais il y a des problèmes à l'exécution.

Nebel et Koehler se sont penchés sur le problème de vérifier s'il est plus rapide de tenter de modifier le plan courant pour le corriger lorsque les objectifs et l'état initial sont modifiés que de tout simplement replanifier à partir de zéro (*replanning from scratch*) Nebel et Koehler (1992). Ils y démontrent entre autres que le problème de réparation de plan est de la même complexité algorithmique que la planification elle-même, mettant en relief certains résultats empiriques de l'époque qui tendaient à montrer qu'il était beaucoup plus rapide de modifier un plan. C'est particulièrement vrai si le nouveau plan doit respecter le principe du conservatisme, c'est-à-dire la réutilisation au maximum de l'ancien plan. Dans ce cas, il peut aussi, à l'inverse, être plus difficile de réparer l'ancien plan que d'en trouver un nouveau à partir de zéro. Cette approche demeure pertinente lorsque l'on planifie en ligne (en même temps que l'exécution). On désire alors un plan modifié qui ne diffère pas trop du plan original afin de limiter les perturbations.

Même si la complexité algorithmique théorique est la même dans les deux cas (réparation et replanification), certaines heuristiques ont été développées pour accélérer la réparation de plans. Ces dernières permettent en pratique, dans certains cas, de réparer un plan plus rapidement que d'en refaire un. Cette supériorité pratique est notamment étudiée dans Krogt et Weerdt (2005), où les auteurs proposent un cadre général pour unifier les méthodes de réparation. Ils argumentent aussi que si autant d'efforts étaient investis dans les techniques de réparation que dans les techniques de planification, le gain pratique serait encore plus grand.

Même s'il est possible en pratique de modifier un plan plus rapidement que d'en

refaire un, il faut tout de même s'assurer que le nouveau plan est stable. Cette stabilité permet à l'agent de respecter ses engagements et de limiter les actions inutiles lorsque la modification est faite en ligne. Les engagements sont une notion très pertinente dans un contexte multi agents : si un agent s'attend à ce qu'un autre accomplisse un objectif et que ses propres actions en dépendent, il faut que cet engagement soit respecté. Une justification détaillée du besoin de stabilité ainsi qu'une méthode permettant d'adapter le plan modifié à son contexte sont fournies dans Fox *et al.* (2006). Ces derniers définissent la stabilité d'un plan en mesurant à quel point il diffère de l'ancien. Il est montré empiriquement que pour certains domaines, les plans réparés sont en général plus stables que les plans refaits à partir de zéro.

Dans Cushing et Kambhampati (2005), les auteurs tentent plutôt de développer des techniques de replanification qui prennent en compte la stabilité ainsi que les engagements déjà pris par l'agent. On parle alors de réutilisation de plans. Les auteurs argumentent que la métrique utilisée dans Fox *et al.* (2006) n'est pas pertinente et que la stabilité ne dépend pas de la stabilité entre le nouveau plan et l'ancien. Afin de replanifier en respectant les engagements et la stabilité, ils cherchent à savoir ce qui a pu causer un échec et à l'éviter absolument (contraintes dures) tout en évitant au maximum de briser un engagement (contraintes molles).Cette approche est dans un certain sens analogue à la PSP.

Puisqu'elles sont plus rapides, les méthodes de réparation de plans et de replanification semblent préférables pour certaines applications. Cependant, dans certains domaines, nous disposons de beaucoup de temps avant l'exécution pour planifier adéquatement. C'est le cas des robots sur Mars, pour lesquels la planification des activités quotidiennes se fait la veille. De plus, à l'exécution, il est préférable d'avoir un plan contingent prévoyant les principales possibilités d'échec puisque nous voulons optimiser le temps du robot sur la planète. En effet, pour réparer

les plans ou en replanifier, le robot doit être en arrêt. Rappelons que lors de la mission Pathfinder (Bresina *et al.*, 2002), le robot Sojourner a été inactif de 40% à 75% du temps puisqu'il devait constamment replanifier ou corriger ses plans. C'est pourquoi des approches comme celles que nous proposons, soit la création de plans conditionnels grâce à la PSP sous incertitude, nous semblent plus justifiées pour ce genre d'applications. De plus, notre méthode assure implicitement la stabilité puisque le plan n'est jamais modifié.

3.6 Conclusion

Ce chapitre présente la PSP dans un contexte où de nouvelles hypothèses de la planification classique sont abandonnées. En effet, nous ajoutons que les actions ont des durées incertaines suivant des distributions probabilistes. Nous avons justifié la nécessité d'une nouvelle méthode permettant d'avoir un sous-ensemble d'objectifs satisfaits dynamique à l'exécution, ce qui n'était pas possible avec les approches pour le problème déterministe. Pour ce faire, nous avons introduit de nouvelles actions spéciales d'abandon d'objectifs. Ces actions permettent de créer des points de branchement dans un plan conditionnel décidant de la poursuite ou non d'un objectif, permettant une adaptation de l'agent pendant l'exécution. Afin de permettre un contrôle sur ces abandons, nous permettons aussi un temps limite sur l'abandon d'objectifs. Ainsi, il est possible de décider qu'un objectif doit être satisfait ou abandonné avant un certain temps. Un exemple détaillé montre la pertinence de l'approche. Nous avons aussi discuté d'alternatives plus rapides à la PSP sous incertitude, soit la replanification et la réparation de plans, mais qui sont moins adaptées au genre de problématiques présentées.

[Cette page a été laissée intentionnellement blanche]

CHAPITRE IV

INTÉGRATION DE LA PLANIFICATION DANS UN JEU SÉRIEUX

Dans ce chapitre, nous nous intéressons au développement d'agents intelligents et réalistes dans les jeux vidéo. Nous limitons cependant l'application de nos méthodes à un jeu sérieux de simulation. Un jeu sérieux est un jeu qui a une finalité dite sérieuse, comme l'apprentissage, en plus d'être ludique. Des études ont montré que le réalisme facilite l'apprentissage du joueur (Raybourn, 2007). Dans une simulation, il peut y avoir un grand nombre de personnages non-joueur PNJ interagissant avec le joueur et ces PNJ doivent donc tous agir le plus possible comme des humains afin de conserver le réalisme.

La planification automatisée est une solution naturellement intéressante pour créer des PNJ qui sont à la fois adaptatifs et autonomes. En effet, la planification permet d'obtenir un plan d'action convenant à n'importe quel problème seulement à partir d'une description du monde et d'une liste des actions possibles. Si du travail a déjà été fait pour coordonner de manière optimale des agents coopératifs (Scharpff *et al.*, 2013; Talamadupula *et al.*, 2013), cette coordination optimale peut altérer le réalisme du jeu puisque le joueur va rapidement se rendre compte qu'il a affaire à des PNJ agissant «comme des robots».

En plus d'être en compétition avec le joueur, les PNJ peuvent aussi être en compétition directe avec d'autres PNJ. Ainsi, la coordination et l'optimisation des

actions ne sont pas des caractéristiques désirables puisque la compétition mène naturellement à des choix globalement sous-optimaux (il suffit de penser au traditionnel dilemme du prisonnier en théorie des jeux (Rasmusen, 2007)). Ces agents en compétition ne devraient planifier que selon leur propre connaissance de l'état du monde et changer leur plan lorsqu'il échoue. Cette replanification doit être faite en ligne (pendant que le jeu continue) et rapidement afin que la simulation ne contienne pas de PNJ inactifs.

Ces considérations nous ont poussés à chercher des moyens de créer et de gérer des PNJ en utilisant la planification automatisée. Une limite actuelle dans le domaine est que les objectifs à satisfaire doivent être sélectionnés et fournis manuellement par un agent externe au planificateur. Il serait souhaitable qu'au contraire cette sélection puisse se faire de manière unifiée. Nous avons donc développé de nouvelles méthodes de distribution d'objectifs à des agents planifiant individuellement. Puis, nous avons utilisé la planification à satisfaction partielle afin que les agents puissent choisir eux-mêmes les objectifs qu'ils désirent accomplir ainsi que le nombre de ceux-ci. Aussi, la planification n'ayant que rarement été utilisée dans les jeux de simulation (Champandard, 2013), nous intégrons nos méthodes dans un jeu sérieux de vente immobilière nommé *SimRealEstate*. Ces travaux ont mené à une publication à la *IEEE Conference on Computationnal Intelligence and Games* (Labranche *et al.*, 2014).

Ce chapitre décrit d'abord le jeu *SimRealEstate* à la section 4.1, puis offre un bref état de l'art de l'utilisation de la planification automatisée dans les jeux (section 4.2). Les sections 4.3, 4.4 et 4.5 décrivent respectivement comment utiliser la planification automatisée dans cette simulation, les méthodes développées pour distribuer les objectifs et puis la méthode de sélection à l'aide de la PSP. Nous terminons par un compte-rendu des expérimentations effectuées.

4.1 *SimRealEstate*

SimRealEstate est un jeu sérieux dont les objectifs sont d'enseigner les bases des transactions immobilières. Ces bases incluent par exemple l'évaluation de la valeur d'une propriété et les différentes étapes d'une vente. Pour garder le joueur intéressé, le jeu est conçu pour être ludique et n'explicite pas ses objectifs pédagogiques. Pour réussir dans le jeu, le joueur doit maîtriser des concepts immobiliers réalistes, qu'il pourra par la suite appliquer à des situations de la vie réelle. Pour ce faire, il incarne un agent immobilier dont le but est d'être le meilleur de la ville. La performance est évaluée à partir de son portefeuille, mais aussi selon sa réputation (à quel point ses clients sont satisfaits en général).Pour l'instant, le jeu est surtout centré sur la partie «vente» des transactions immobilières. Le joueur offre ainsi ses services pour aider les vendeurs à vendre leur propriété.

La figure 4.1 montre une capture d'écran du jeu. Le joueur peut déplacer son avatar de maison en maison. En termes de planification automatisée, vendre une maison est considéré comme un objectif.

Pour vendre une maison, les courtiers du jeu peuvent accomplir les cinq actions suivantes :

- `aller(o, d)` : déplacer l'avatar d'une origine o jusqu'à une destination d.
- `offrir(m, taux)` : offrir par contrat ses services de courtier au vendeur m en offrant le taux de commission $taux$.
- `mettreEnVente(m, prix)` : fixer le prix de vente à $prix$, mettre une affiche devant la maison m et acheter de la publicité.
- `faireVisiter(m, a)` : faire visiter la maison m aux acheteurs potentiels a.
- `rAccepter(m, offres)` : recommander au vendeur m d'accepter la plus haute offre d'achat parmi $offres$ si celle-ci est plus haute qu'un certain

Figure 4.1: Capture d'écran du jeu *SimRealEstate*

seuil que le vendeur est prêt à accepter.

Quand une maison est vendue, le courtier obtient un montant égal au prix de vente multiplié par le taux de commission négocié. Son niveau de réputation est déterminé en fonction du temps qu'il lui a fallu pour vendre la maison ainsi que par la différence entre le montant espéré du vendeur et celui de la vente.

L'environnement simulé est dérivé de données véritables. Le jeu se déroule dans une vraie ville représentée par une carte importée du projet OpenStreetMap (OSM) (Haklay et Weber, 2008)[1]. Les caractéristiques des maisons sont obte-

1. `http://openstreetmap.org/`

nues à partir de données publiques de maisons réellement en vente dans la ville. Plus l'environnement est proche de la réalité, plus le joueur peut en apprendre sur la ville choisie et son marché immobilier réel.

L'environnement virtuel n'est cependant pas qu'une copie du marché d'une ville. *SimRealEstate* est aussi une simulation qui s'adapte aux compétences du joueur. En utilisant la planification automatisée, des scénarios uniques visant le développement d'une compétence particulière sont générés pour chaque joueur (Sola, 2015).

4.1.1 Simulation multi-agents

Le jeu est basé sur une simulation multi agents comportant trois types de PNJ : les vendeurs, les acheteurs et les courtiers. Ces agents sont créés et introduits dans le jeu à un taux suivant une distribution exponentielle. Les taux moyens d'apparition sont des paramètres ajustables permettant de modifier la difficulté du jeu.

Puisqu'il peut y avoir des centaines d'agents actifs à la fois, leur module d'intelligence artificielle doit être optimisé en termes de consommation de ressources de calcul. Dans un premier temps, l'intelligence artificielle de chacun a été écrite à l'aide de scripts. Afin d'être imprévisibles et réalistes, les scripts incluent des décisions aléatoires. L'écriture des scripts étant laborieuse et difficilement évolutive, c'est une des raisons qui nous poussent à vouloir utiliser la planification automatisée.

Pour que le jeu soit plus intéressant, le joueur n'est pas le seul courtier de la simulation. Il y a en effet plusieurs courtiers PNJ qui partagent le même but que le joueur et qui sont donc ses compétiteurs directs. Ces courtiers doivent agir comme de vrais courtiers immobiliers. C'est pour ces courtiers que nous proposons

un passage des scripts à la planification automatisée.

Les vendeurs et les acheteurs restent dans le cadre de ce mémoire des PNJ scriptés et ne font pas l'objet d'une étude approfondie.

4.2 La planification dans les jeux

Les trois bénéfices espérés de l'utilisation de la planification dans les jeux sont (1) que les PNJ puissent mieux réagir aux situations inattendues, (2) que les objectifs et les actions soient des parties constituantes facilement réutilisables, partageables et maintenables et (3), que l'architecture de planification soit séparée de l'architecture du jeu lui-même, permettant que l'intelligence artificielle soit développée en parallèle du jeu, sans jamais interrompre le flux de travail (Orkin, 2004). L'architecture *Goal-Oriented Action Planning* (GOAP) proposée dans Orkin (2004) définit les bases visant à fournir aux PNJ des actions et des objectifs au lieu de comportements scriptés. Cependant, la planification dans les jeux commerciaux est plutôt limitée à des réseaux de tâches hiérarchisées (Verweij, 2007; Champandard, 2013).Ses applications sont généralement aussi limitées aux jeux de tir à la première personne et de stratégie en temps réel.

D'un côté plus théorique, la planification a été utilisée pour simuler le comportement d'un soldat PNJ dans un jeu sérieux militaire (Menif *et al.*, 2013). Chaque soldat planifie pour lui-même, tandis qu'un module de coordination de haut niveau planifie le comportement général de groupes de soldats. Cette approche est intéressante dans un environnement coopératif. Les auteurs introduisent aussi l'idée de laisser un agent exécuter son plan tant qu'il n'échoue pas, puis de replanifier si c'est le cas.

D'un point de vue plus abstrait, la planification peut servir dans les systèmes tutoriels intelligents pour déterminer les actions optimales à suivre pour maximiser

l'apprentissage d'une personne (Rafferty *et al.*, 2011).

Des travaux portent sur un agent « conscient » et autonome qui utilise la planification (Liu et Schubert, 2014). L'agent apprend sur son monde, choisit lui-même ses buts et ensuite planifie ses actions. Cette approche offre du réalisme et une possibilité de compétitivité avec d'autres agents. Cependant, la force computationnelle nécessaire rend inapplicables ces travaux dans le cadre d'un jeu où plusieurs centaines de tels agents sont mobilisés.

Dans Talamadupula *et al.* (2013), des paradigmes de replanification sont présentés. Dans un contexte multi agents, les agents peuvent replanifier à partir de zéro, replanifier pour réduire les coûts computationnels ou encore replanifier en tenant compte des plans des autres agents pour optimiser leurs propres actions. Dans ce dernier paradigme, plus complexe et demandant, les plans des autres agents sont ainsi considérés comme des engagements qu'il serait bénéfique de satisfaire. Ces engagements peuvent être modélisés comme des objectifs mous et ainsi la PSP peut être utilisée pour résoudre le problème. La motivation des auteurs est cependant plutôt dirigée vers l'optimisation et la coopération.

Les auteurs de (Scharpff *et al.*, 2013) décrivent une manière de planifier avec des préférences inconnues et contradictoires des autres agents. Ils utilisent des MDP Bellman (1957) au niveau de l'agent pour modéliser l'incertitude sur les objectifs des autres agents. Ensuite, un module de haut niveau détermine des politiques communes efficaces. Si dans notre problème les objectifs des autres courtiers peuvent aussi être vus comme inconnus, nous pouvons éviter les coûts computationnels supplémentaires de cette méthode puisque l'optimalité individuelle des agents n'est pas primordiale.

4.3 La planification dans *SimRealEstate*

Comme mentionné plus haut, nous nous concentrons sur la modélisation de courtiers immobiliers non-joueurs. En nous inspirant des idées de Menif *et al.* (2013) concernant la coordination des agents coopératifs, nous devons coordonner des agents en compétition les uns avec les autres, qui planifieront individuellement pour maximiser leur propre score plutôt qu'un score commun. Nos préoccupations premières sont la simplicité de modélisation ainsi que la fluidité du jeu. Cependant, cela ne doit pas se faire au détriment du réalisme, puisqu'il est crucial dans un jeu sérieux que le réalisme soit maintenu à un niveau maximum pour favoriser l'expérience de l'apprenant (Raybourn, 2007).

Nous utilisons le paradigme de replanification à partir de zéro tel que décrit dans Talamadupula *et al.* (2013), où un nouveau plan est systématiquement trouvé à chaque échec d'exécution. Beaucoup de travail a été fait pour optimiser la planification distribuée multi agents (comme nous l'avons vu à la section 4.2), mais il importe de saisir qu'une exécution optimale n'est pas la finalité ici. En effet, des centaines de compétiteurs agissant de manière optimale entre eux altéreraient inévitablement le réalisme du jeu, puisque les courtiers immobiliers ne se coordonnent pas optimalement dans la vraie vie. Nous tentons donc d'appliquer des plans individuellement optimaux, mais globalement sous-optimaux, jusqu'à ce qu'un échec soit constaté pour offrir un comportement plus réaliste. Aussi, les plans trouvés doivent être simples et la représentation du monde interne des agents limitée pour qu'il soit plus facile de replanifier lorsque de nouvelles informations surviennent, notamment la raison pour laquelle le plan original a échoué. L'utilisation de ce paradigme mène à des échecs fréquents, parce que les PNJ sont en compétition pour les mêmes contrats, qui deviennent indisponibles lorsqu'un courtier les obtient, ce qui mène les plans des autres courtiers à échouer. La re-

planification permet une gestion plus efficace, car un nouveau plan est facile à trouver.

4.3.1 Objectifs et représentation d'un état

Lorsqu'un PNJ planifie ses actions, il considère uniquement la situation courante ; les actions possibles des autres PNJ sont ignorées. Deux méthodes de sélection des objectifs sont utilisées et comparées. D'un côté, un ensemble de n objectifs est choisi et envoyé au PNJ par un module de coordination central. Un plan est généré et doit satisfaire tous les objectifs, selon le modèle de la planification classique. Puisque le temps-machine est limité, nous fixons le nombre d'états qu'il est possible d'explorer à une taille raisonnable. Ainsi, n doit être petit et calibré minutieusement pour que le PNJ puisse planifier avec un nombre restreint d'états à explorer.De l'autre côté, cette sélection est faite automatiquement par la PSP (méthode décrite à la section 4.5).

Pour obtenir un espace d'états le plus petit possible, la représentation du monde d'un agent doit se limiter aux variables d'état reliées à ses objectifs. Une boucle alterne entre planification et exécution pour intégrer les changements à l'environnement. Ces changements incluent l'assignation de nouveaux objectifs et, par extension, l'ajout et la suppression de variables d'état.

Dans *SimRealEstate*, le but d'un courtier PNJ est un ensemble de maisons à vendre. Il n'a donc pas besoin de connaître l'emplacement de chaque maison de la carte, mais seulement de celles qu'il tente de vendre, de même pour l'état de ces maisons (en recherche de contrat, en vente, en attente de visite, en attente d'une réponse d'offre).

4.3.2 Actions

Cette section présente comment les actions sont décrites dans *SimRealEstate*. Pour que la planification soit plus simple, le coût des actions est représenté en temps seulement. Une gestion plus complexe des ressources (argent, essence, etc.) rendrait la tâche trop exigeante en termes de calculs. Le planificateur génère des plans qui minimisent le temps d'exécution total pour vendre toutes les maisons.

Les différentes actions possibles sont données à la figure 4.2. Certains événements surviennent à des temps constants et d'autres de manière aléatoire dans le jeu, mais nous utilisons une seule constante, *TempsMoyenEvenement*, qui estime le temps moyen. Ces différents événements sont :

1. Un vendeur accepte le taux de commission offert par le courtier (le courtier obtient le contrat) ;
2. Un visiteur vient visiter une maison en vente ;
3. Un acheteur fait une offre sur une maison et
4. Un vendeur accepte l'offre qui lui est faite.

La variable *TempsAction* représente le temps nécessaire pour accomplir une action. Nous utilisons la même pour toutes les actions pour des raisons de simplicité, mais des valeurs différentes peuvent être utilisées facilement sans que cela vienne complexifier le problème.Par exemple, dans la simulation, ce temps est de 30 minutes simulées.

Nous avons considéré l'utilisation de la PSP sous incertitude, telle que décrite au chapitre 3. Cependant, considérer l'incertitude entraîne une explosion de l'espace d'états, ce qui réduit grandement le nombre d'objectifs pour lesquels un agent peut planifier. L'avantage de la PSP sous incertitude est que les plans risquent moins d'échouer, mais comme il a été discuté précédemment, ce n'est pas un objectif en soi dans ce contexte.Ainsi, même si la replanification est moins souvent

```
action aller(Lieu o, Lieu d){
  duree: distance(o,d) / vitesse;
  conditions:
    @debut: position=o;
  effets:
    @fin: position=d;
}
action offrir(Maison m, Lieu l){
  duree: tempsAction;
  conditions:
    @debut: position(m)=l;
    @debut: position=l;
    @debut: !aContrat(m);
  effets:
    @fin + TempsMoyenEvenement: aContrat(m);
}
action mettreEnVente(Maison m, Lieu l){
  duree: tempsAction;
  conditions:
    @debut: position(m)=l;
    @debut: position=l;
    @debut: aContrat(m);
    @debut: !estEnVente(m);
  effets:
    @fin: estEnVente(m);
    @fin + TempsMoyenEvenement: visiteurs(h);
}
```

```
action faireVisiter(Maison m, Lieu l){
  duree: tempsAction;
  conditions:
    @debut: loc(h)=l;
    @debut: pos=l;
    @debut: visiteurs(h);
  effets:
    @fin: !visiteurs(h);
    @fin + TempsMoyenEvenement: offre(h);
}
action rAccepter(Maison m, Lieu l){
  duree: tempsAction;
  conditions:
    @debut: position(m)=l;
    @debut: position=l;
    @debut: offre(m);
  effets:
    @fin + TempsMoyenEvenement: vendu(m);
    @fin: !offre(hm;
}
```

Figure 4.2: Spécification partielle des actions *aller*, *offrir*, *mettreEnVente*, *faireVisiter* et *rAccepter*

nécessaire avec des plans conditionnels, les coûts computationnels supplémentaires nécessaires ne sont pas compensés.

4.3.3 Exemple de plan

Notre planification implémente l'algorithme A^* aidé de l'heuristique H_{max} (Haslum et Geffner, 2000) pour déterminer les plans des courtiers PNJ.

La figure 4.3 donne un exemple d'un plan dans le cadre du jeu pour un courtier qui doit planifier trois objectifs.

L'agent considère trois maisons, nommées *maison*1, *maison*2 et *maison*3, situées respectivement aux nœuds numéro 1, 23 et 55 (la carte OSM est un graphe). On peut constater que le courtier ne détient pas présentement les contrats pour la *maison*1 et la *maison*2, car le plan contient toutes les étapes de la vente pour

```
1. aller noeud1
2. offrir maison1
3. aller noeud23
4. offrir maison2
5. mettreEnVente maison2
6. aller noeud1
7. mettreEnVente maison1
8. aller noeud55
9. rAccepter maison3
10. aller noeud23
11. faireVisiter maison2
12. aller noeud1
13. faireVisiter maison1
14. aller noeud23
15. rAccepter maison2
16. aller noeud1
17. rAccepter maison1
```

Figure 4.3: Exemple de plan pour un courtier dans *SimRealEstate*

ces maisons (à savoir *offrir*, *mettreEnVente*, *faireVisiter* et *rAccepter*). Il détient le contrat pour la *maison3* : la maison est en vente et des offres d'achat sont en attente, puisque la seule action planifiée est *rAccepter*.

4.4 Distribution des objectifs

La sélection et la distribution des objectifs forment la considération principale de ce chapitre. Dans une simulation contenant des centaines de PNJ et tout autant d'objectifs, il est difficile d'assigner à chacun un sous-ensemble cohérent. Il importe aussi de garder en tête qu'un objectif peut être assigné à plusieurs PNJ différents. Nous proposons différentes méthodes de sélection et de distribution utilisant un module de coordination de haut niveau. Cette approche permet par la suite de résoudre les problèmes en utilisant la planification classique.

Premièrement, nous proposons de distribuer les objectifs de manière aléatoire. Puis, nous considérons la difficulté d'un objectif pour le PNJ et lui assignons ceux qui sont considérés comme les plus faciles. Comme troisième approche, nous calculons à quel point un objectif est bénéfique (son utilité) pour l'agent et lui assignons les plus bénéfiques en premier. Finalement, nous expérimentons une

combinaison de la difficulté et de l'utilité.

4.4.1 Sélection aléatoire

La manière la plus simple d'affecter des objectifs est de les distribuer de manière aléatoire. Dans le jeu, les vendeurs sont créés de manière aléatoire à différents temps. Une liste contient les maisons en recherche d'un courtier et le module de coordination choisi n maisons aléatoirement dans la liste.

Cela peut être problématique, car l'ensemble ainsi choisi peut contenir des maisons très éloignées l'une de l'autre, forçant le courtier à toujours parcourir de longues distances. Ce serait justifié si ces maisons avaient toutes une grande valeur, mais la distribution aléatoire ne le garantit en rien. Nous ne voulons pas nécessairement des PNJ optimaux, mais il ne faut pas non plus qu'ils soient trop naïfs.

4.4.2 Sélection par difficulté

Au lieu de considérer un résultat incertain pour chaque action afin de prendre en compte le comportement des autres PNJ, il est plus simple d'estimer ce comportement. Cette estimation peut être obtenue en déterminant à quel point un contrat pour une maison en vente est difficile à obtenir. Logiquement, plus il y a de courtiers l'ayant dans leurs objectifs, plus il sera difficile de l'obtenir. Nous définissons la probabilité a priori d'obtenir un contrat pour une maison m par :

$$P(OffreAcceptee(m)) = \frac{1}{NbCourtiersInteresses + 1}$$

avec *NbCourtiersInteresses*, soit le nombre de PNJ ayant la maison m dans leurs objectifs.Cette probabilité est comparable avec l'incertitude sur le comportement des autres agents.

Une autre considération en ce qui concerne la difficulté est le fait que le vendeur détermine lui-même le prix qu'il est prêt à accepter. Ce vendeur peut sous ou surévaluer la valeur de sa maison, ce qui la rend plus ou moins facile à vendre. Afin de déterminer s'il y a sous ou surévaluation de la valeur, le module de coordination cherche des maisons comparables et détermine un prix moyen de vente. Nous pouvons en extraire le rapport de qualité *RapportQualité* d'une maison m donné par :

$$RapportQualite(m) = \frac{PrixDemandeVendeur(m)}{PrixMoyenMaisonsComparables(m)}$$

et ainsi la difficulté estimée de la vente d'une maison :

$$Difficulte(m) = \frac{RapportQualite(m)}{P(OffreAcceptee(m))}$$

Le module de coordination assigne les objectifs en ordre croissant de difficulté. La figure 4.4 illustre l'architecture en jeu. Le module de coordination sélectionne un sous-ensemble $g_1 \subseteq \mathcal{G}$ et l'assigne à l'agent 1.L'agent 1 choisit les n moins difficiles, planifie pour les satisfaire puis transmet ses choix c_1. Le module calcule et assigne les nouvelles probabilités aux objectifs choisis par l'agent 1 et assigne un nouveau sous-ensemble g_2 à l'agent 2.

Il subsiste un problème : si cette formule permet de calculer à quel point un objectif peut être difficile à accomplir, il ne permet pas de calculer à quel point il peut être bénéfique.

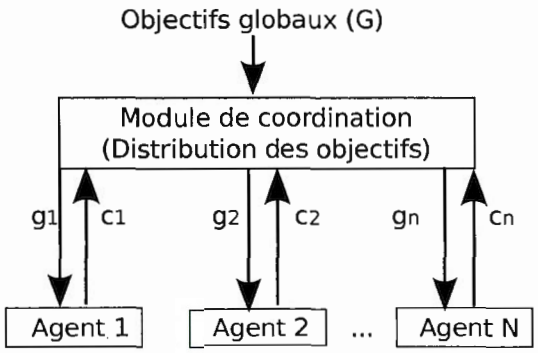


Figure 4.4: Communication entre le module de coordination et les agents pour modifier la difficulté des objectifs

4.4.3 Sélection par utilité

Une autre façon de sélectionner les objectifs est de déterminer à quel point ils peuvent être profitables pour un agent. Rappelons que c'est de cette manière que la PSP détermine quels objectifs satisfaire. Cette profitabilité n'est nulle autre que l'utilité.

Dans *SimRealEstate*, l'utilité d'une maison m peut naturellement être déterminée par son prix de vente multiplié par le taux de commission du contrat. Cependant, l'utilité devrait aussi dépendre de la vitesse à laquelle une maison peut être vendue. La vitesse de vente dépend entre autres de la distance entre les maisons de l'ensemble d'objectifs. En effet, plus elles sont rapprochées, plus le courtier peut s'y déplacer rapidement afin d'accomplir les différentes actions. Nous définissons donc l'utilité d'une maison m par :

$$Utilite(m) = \frac{TauxCommission(m) \times PrixVente(m)}{\frac{1}{|\mathcal{G}|+1} \times \left(\sum_{i=1}^{|\mathcal{G}|} Dist(m, g_i) + Dist(m, Agent) \right)}$$

où $\mathcal{G}$ est l'ensemble des maisons déjà assignées au courtier. Le module de coordi-

nation assigne les maisons en ordre décroissant d'utilité.

Diviser l'utilité par la moyenne des distances permet de s'assurer que beaucoup d'importance leur est accordée. Cela pousse les agents à développer naturellement un territoire de vente : ils prendront des contrats seulement dans une région restreinte. Cependant, si une maison de très grande valeur est mise en vente à l'extérieur de son territoire, l'agent pourrait toujours essayer d'obtenir le contrat. Ce comportement est analogue à celui observé dans la réalité, où les courtiers évitent de s'éparpiller un peu partout dans la ville. Un joueur attentif pourra observer ce comportement et tenter de le reproduire afin de maximiser son score.

4.4.4 Sélection par utilité et difficulté

Il est pertinent, puisque l'utilité et la difficulté offrent toutes deux des avantages, de tenir compte des deux à la fois. Comme nous voulons une utilité aussi grande que possible et une difficulté aussi petite que possible, nous définissons :

$$UtilDiff(m) = \frac{Utilite(m)}{Difficulte(m)}$$

et assignons les maisons en ordre décroissant d'*UtilDiff*.

4.5 Sélection des objectifs avec la PSP

En nous contentant de sélectionner les objectifs seulement selon un score, nous omettons de considérer une question importante du problème de sélection. En effet, la question du nombre d'objectifs à fournir à chaque agent demeure entière. Par exemple, dans *SimRealEstate*, les contrats ont des temps limites : ils expirent après un certain temps. Aussi, les visiteurs peuvent quitter les maisons où ils attendent et les acheteurs peuvent retirer leur offre si la réponse ne vient pas

assez vite. Ainsi, il faut prendre en compte le fait qu'un agent ne peut pas planifier pour une quantité trop grande d'objectifs, sinon il n'aura pas le temps de tous les réaliser. Au bout du compte, il pourrait même en satisfaire moins que s'il avait planifié à plus petite échelle.

Un autre problème avec un nombre fixe d'objectifs relié à leur sélection est que plus $|\mathcal{G}|$ est grand, plus il est difficile de trouver un plan (plus d'états doivent être explorés). Dans une simulation où des centaines d'agents doivent interagir de manière fluide, le nombre d'états qu'il est possible d'explorer doit être restreint. Adapter le nombre d'objectifs au nombre d'états nécessite un réglage laborieux, analogue à l'écriture de scripts, ce que nous tentons précisément d'éviter.

Il est donc désirable que les agents puissent déterminer par eux-mêmes le nombre d'objectifs pour lesquels ils peuvent planifier, en tenant compte des limitations du jeu (le temps dans *SimRealEstate* par exemple) et du nombre d'états. Notre approche par la PSP permet d'atteindre ces deux objectifs.

Avec les autres méthodes de distribution, tous les objectifs doivent être satisfaits, comme en planification classique. Dans un contexte de PSP, les maisons sont assignées, mais sous forme de contraintes molles, pouvant donc ne pas être satisfaites dans un état final. Nous utilisons l'approche par actions d'abandon présentée au chapitre 3. Il est à noter que si cette approche permet de traiter l'incertitude, elle demeure valide dans un contexte déterministe. Nous ajoutons au modèle du jeu l'action suivante.

```
action abandonner(Maison m){
  durée: tempsAction;
  conditions:
    @debut: Objectifs.contient(m);
    @debut: TempsValides(m) > TempsCourant
```

```
    @debut: TempsEspere(m) <= Utilite(m)
  effets:
    @fin: Objectifs.supprimer(m)
    @fin: !aContrat(m);
}
```

Les conditions indiquent que la maison doit faire partie de l'ensemble d'objectifs. Les *TempsValides* sont les temps limites associés à cette maison (fin du contrat, présence d'un visiteur, etc.). Si l'un d'eux est plus grand que *TempsCourant*, cela signifie que l'agent n'aura pas le temps d'accomplir cet objectif et qu'il serait plus profitable de l'abandonner.

La variable *TempsEspere* indique combien de temps la vente de la maison est censée prendre à l'agent. Ce temps est estimé à l'aide des temps espérés pour chacune des actions en lien avec cette maison. L'utilité d'un objectif doit donc aussi être considérée en terme de temps dans ce contexte, ce qui diffère de la notion d'utilité dans la section précédente. Ainsi, l'utilité est vue comme un temps maximal à investir pour la vente d'une maison et dépend de sa valeur et du temps total moyen nécessaire pour conclure une vente.

$$Utilite(h) = \frac{TauxComission \times PrixDeVente}{ValeurMoyenneContrat} \times TempsMoyenVente$$

ValeurMoyenneContrat est une valeur représentant combien d'argent par contrat gagnent en moyenne les courtiers et TempsMoyenVente est le temps total moyen nécessaire pour vendre une maison. Ainsi, si la vente est censée prendre plus de temps que pour une vente moyenne et que le contrat vaut moins que la valeur moyenne, l'agent ne la considérera pas intéressante. Si cette maison se retrouve dans l'ensemble d'objectifs pour lesquels il planifie, il l'abandonnera.

À la fin, l'objectif est retiré de l'ensemble d'objectifs et l'agent ne possède plus le contrat pour la maison (advenant qu'il l'avait déjà).

Cette approche cause une explosion du nombre d'états à explorer pour trouver une solution. Idéalement, l'agent considérerait toutes les maisons en vente, pour n'en conserver que la quantité réelle pour laquelle il peut planifier lorsqu'il est seulement restreint par les contraintes du jeu. Cependant, nous savons que nous devons aussi limiter le nombre d'états explorés. Pour contourner ce problème sans nécessiter de laborieux réglages, nous proposons la solution suivante. L'idée est simple : l'agent considère d'abord toutes les maisons en vente, puis tant qu'il ne réussit pas à trouver un plan avec le nombre d'états restreints, il réduit son ensemble d'objectifs potentiels d'une maison.

Cette méthode est analogue à une action d'abandon de la maison ayant l'utilité la plus petite. Lorsque l'agent réussit à satisfaire tout le but, il prend confiance et tentera de satisfaire un objectif de plus à la prochaine itération de planification. Il importe de préciser qu'il est tout à fait possible de ne satisfaire que k objectifs avec un certain nombre d'états explorés, mais que si les objectifs changent, il sera possible de trouver un plan pour $k + 1$ objectifs avec le même nombre d'états. Il est ainsi assuré que l'agent essaiera toujours de satisfaire le maximum d'objectifs.

Les méthodes de sélection par le module de coordination introduites à la section précédente peuvent aussi être réutilisées. Au lieu de considérer l'ensemble des maisons en vente au départ, nous pouvons fournir à l'agent un ensemble restreint déterminé par les scores. Nous savons qu'un courtier dans *SimRealEstate* ne peut planifier pour plus de quatre maisons lorsque nous limitons son nombre d'états à 50 000. Il semble donc futile de lui faire considérer au départ un ensemble de plusieurs centaines de maisons à vendre, nombre qu'il décrémenterait de toute façon jusqu'à cinq, quatre ou trois. Nous pouvons utiliser la sélection par utilité

et lui fournir un premier ensemble de départ ne contenant que six maisons.

4.6 Évaluation

Nous utilisons le jeu *SimRealEstate* tel que présenté à la section 4.1 pour mettre en application les différentes méthodes.

4.6.1 Expérimentations

La simulation débute normalement avec le joueur et aucun courtier, vendeur ou acheteur PNJ ; ces derniers sont créés au fur et à mesure. Pour bien évaluer les différences entre les types de distribution et de sélection d'objectifs, nous avons modifié ces paramètres. Nous avons lancé la simulation avec six courtiers PNJ, nommés *Aléa* (aléatoire), *Diff*, *Util*, *DiffUtil* et *PSP*. Puisqu'un courtier scripté était aussi disponible[2] , il est présent dans l'évaluation afin de comparer les performances. Le nombre d'états explorés était limité à 50 000, sauf pour le courtier PSP qui pouvait aller jusqu'à 100 000 pour tenir compte du facteur de branchement augmenté par les actions d'abandon. Nous avons limité le nombre d'objectifs à trois maisons à la fois. Le courtier PSP pouvait choisir le nombre de maisons pour lesquelles il désirait planifier.

Les prix réels sont souvent basés sur des critères subjectifs, comme le quartier et l'apparence des maisons. Afin de contourner ce possible biais, les maisons ont été normalisées pour tenir compte seulement des facteurs considérés dans le jeu, soient le nombre de chambres et la surface habitable. Ainsi, les maisons similaires avaient des prix similaires. Les aspects aléatoires des préférences des vendeurs et des acheteurs ont été conservés, ainsi que leurs taux de création, pour que les

---

2. Le script a été implémenté par Nicolas Sola. Grosso modo, le courtier scripté s'intéresse à la maison qui lui rapporterait le plus et fait une offre agressive en offrant une commission d'un dixième de point de pourcentage de moins que la meilleure offre de courtage présente.

Tableau 4.1: Résultats pour Île-des-Soeurs

Nom	Portefeuille			NbContrats			RMoy	VCMoy
	Min	Moy	Max	Min	Moy	Max		
Script	**92,494**	**172,060**	**250,811**	6	9	14	54	**18,304**
Aléa	39,558	53,128	61,404	4	6	7	70	9,487
Diff	5,345	41,041	89,043	1	5	10	65	8,922
Util	74,166	126,881	178,140	**9**	**14**	**19**	62	9,194
Diff	36,294	55,220	86,927	4	6	8	**76**	9,861
PSP	17,668	56,828	101,224	2	6	10	75	10,148

résultats restent cohérents avec une séance de jeu normale.

Les tests ont été exécutés sur deux cartes :

1. Île-des-Soeurs (une sous-région de Montréal) avec 109 maisons, 2923 nœuds OSM et 535 routes.

2. Montréal avec 2959 maisons, 83 980 noeuds OSM et 22 270 routes.

La simulation a été lancée cinq fois pour une durée de deux mois en temps simulé, ce qui correspond en temps réel à environ 3 heures 30 minutes. Cependant, le joueur peut contrôler la vitesse à laquelle le temps s'écoule et un joueur expérimenté peut compléter une telle séance en environ 1 heure.

Les tableaux 4.1 et 4.2 contiennent respectivement les résultats pour Île-Des-Soeurs et pour Montréal. Les valeurs de portefeuille correspondent aux capitaux que les courtiers possédaient à la fin de la simulation. NbContrats est le nombre de contrats complétés. RMoy et VCMoye sont respectivement les valeurs moyennes de réputation et de contrats. Les valeurs les plus grandes de chaque colonne apparaissent en gras. Dans chaque cas, une valeur plus grande indique une meilleure performance.

Tableau 4.2: Résultats pour Montréal

Nom	Portefeuille			NbContrats			RMoy	VCMoy
	Min	Moy	Max	Min	Moy	Max		
Script	39,201	**93,432**	**122,428**	**5**	**9**	11	63	**10,900**
Aléa	5,907	34,765	60,160	2	5	8	75	6,953
Diff	12,506	35,335	49,683	3	5	6	77	7,495
Util	13,923	40,407	87,267	3	6	10	78	7,071
DiffUtil	18,689	47,009	97,554	3	6	10	84	7,835
PSP	**44,118**	68,844	99,812	**5**	8	**12**	**88**	8,168

4.6.2 Analyse

Les résultats suggèrent qu'un courtier scripté performe mieux en termes d'argent gagné. Cela est normal puisque les scripts ont été écrits par des experts et encodent des stratégies. Rappelons que nous avons choisi consciemment un comportement sous-optimal pour les PNJ utilisant la planification pour offrir un comportement plus réaliste : les courtiers tentent d'obtenir un contrat puis, si ça ne fonctionne pas, ils replanifient. Plus important encore, les méthodes proposées ici se veulent une alternative à la laborieuse rédaction de scripts. Les résultats du PNJ scripté sont montrés à titre indicatif seulement, parce que les scripts étaient déjà disponibles.

Toutefois, si *PSP* ne peut compétitionner contre *Script* sur la carte plus petite, les performances sont comparables sur la plus grande. Les faibles résultats sur la carte plus petite sont explicables par le fait que la qualité des choix n'est pas très importante puisque toutes les maisons sont assez proches l'une de l'autre, tant au niveau des caractéristiques que géographiquement. Lorsque la carte est plus grande, il est pertinent de choisir des maisons qui sont près géographiquement les unes des autres et ainsi de se définir un territoire de vente. La figure 4.5 montre ce comportement : plusieurs vendeurs recherchent un courtier (maisons vertes) et sont dispersés sur toute la carte. Cependant, en utilisant la PSP, *PSP* a choisi de

prendre des contrats pour seulement trois maisons (en orange dans le cercle noir, l'avatar éclipsant la troisième) assez près l'une de l'autre. Ces choix permettent de tendre vers des ventes plus rapides et plus efficientes en termes de déplacement pour le courtier. Ce comportement, non scripté et résultant directement de l'utilisation de la PSP, est un outil d'apprentissage puissant pour le joueur, qui peut l'observer et tenter de le reproduire pour être à son tour plus efficace. Cette analyse suggère que l'utilisation de la PSP pour la création de PNJ est plus adaptée à des situations où le choix des objectifs est une composante importante du comportement.

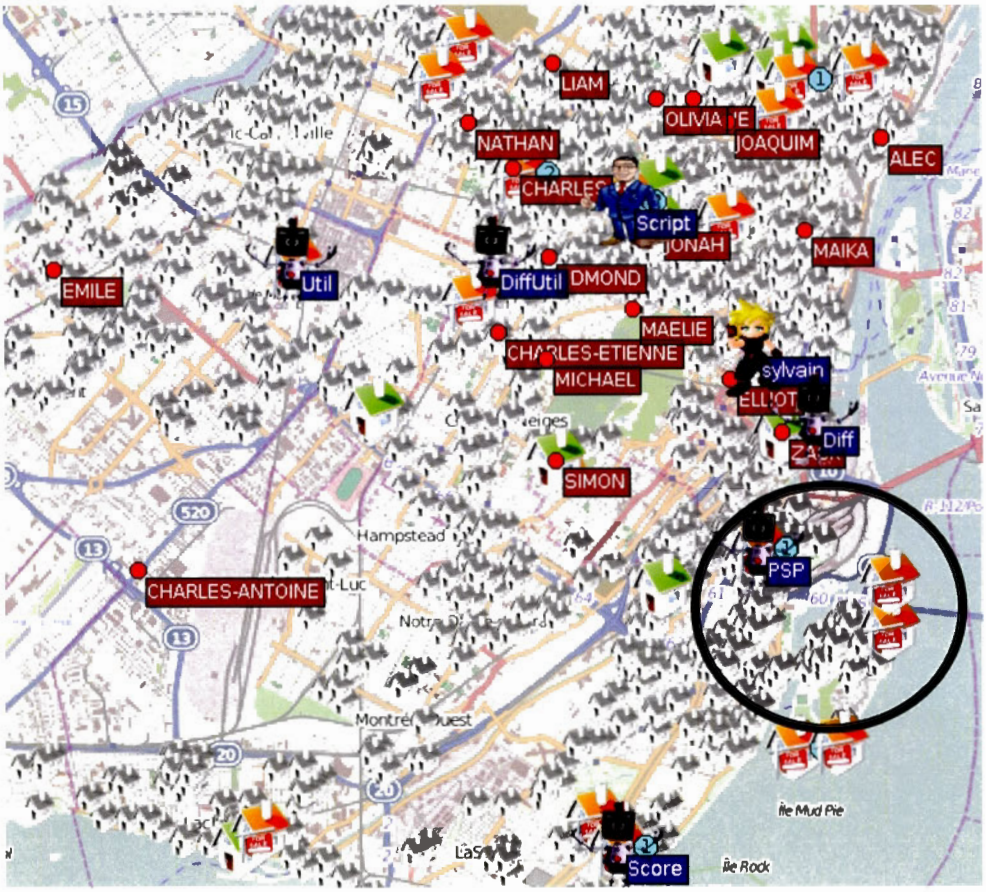


Figure 4.5: Territoire de vente de *PSP*

Pour les autres méthodes de distribution (utilisant seulement un module de coor-

dination central), elles semblent être un bon compromis entre réglage minutieux et autonomie. *Util* et *DiffUtil* performent bien, avec des résultats comparables avec ceux de *PSP* sur la carte de Montréal. Cela est normal, puisque le raffinement des choix offerts à *PSP* se base sur l'utilité, donc les maisons imposées à *Util* et *DiffUtil* et suggérées à *PSP* sont semblables. La difficulté considérée seule ne semble pas bien performer et une distribution aléatoire mène aux pires performances pour les deux villes. Les techniques de planification classique requièrent moins d'états que la PSP pour offrir des performances similaires. Ainsi, de tels PNJ pourraient être utilisés en plus grand nombre sans ralentir le jeu.

Ces méthodes sont toutes pertinentes puisqu'il est possible de s'en servir en utilisant le même modèle de planification pour créer différents niveaux de difficulté. Ainsi, le module de coordination pourrait décider quelle méthode de distribution utiliser selon la performance du joueur et ainsi s'adapter automatiquement.

Nous avons observé que *PSP* planifiait rarement pour plus de trois objectifs et jamais plus que pour quatre. Trois est une valeur que nous avions trouvée en réglant nous-même minutieusement le nombre d'objectifs par rapport au nombre d'états explorables. Il est intéressant de constater que *PSP* a trouvé cette limite moyenne par lui-même. Ainsi, la PSP pourrait être une approche pertinente pour trouver le nombre d'objectifs satisfaisables par rapport au nombre d'états.

L'utilisation de plusieurs PNJ utilisant la PSP ne résulte pas nécessairement en des comportements homogènes. Puisque les plans peuvent échouer et qu'il y a replanification, les courtiers se dispersent par eux-mêmes dans la ville et constituent différents territoires de vente.

Il n'est pas évident de mesurer à quel point l'utilisation de la planification est plus simple et rapide que l'écriture de scripts. Cependant, la section 4.3.2 montre à quel point il est facile d'ajouter, de supprimer ou de modifier des actions. Il en va

de même pour la représentation interne des états. En comparaison aux quelques centaines de lignes de code nécessaires pour le courtier scripté, les agents planificateurs ont simplement besoin d'une représentation d'états et des actions possibles. Avec la PSP, ils peuvent même choisir quels objectifs ils désirent accomplir. Nous croyons, et les expérimentations le suggèrent, que notre approche offre certains avantages en termes de temps et simplicité.

4.7 Conclusion

Ce chapitre traitait du problème de la modélisation de PNJ pour les jeux vidéo. Plus particulièrement, nous nous sommes penchés sur une approche alternative à l'écriture de scripts pour le contrôle des PNJ : la planification. Cette approche, de plus en plus utilisée dans le domaine, permet de créer des PNJ dont la maintenance est plus aisée plus rapidement et plus facilement.

Cependant, au meilleur de notre connaissance, le problème de la distribution automatique et unifiée des objectifs à ces PNJ n'avait pas à ce jour été étudié. Nous nous sommes donc posé la question. Premièrement, nous avons proposé des méthodes basées sur des scores pour une distribution par un module de coordination central. Puis, nous avons suggéré l'utilisation de la PSP, qui permet que les PNJ choisissent par eux-mêmes leurs objectifs ainsi que la quantité de ceux-ci.

Ces méthodes ont été implémentées dans le jeu sérieux de simulation immobilière *SimRealEstate*. Des courtiers utilisant les différentes approches de distribution et la PSP ont été comparés. L'usage de la planification s'est révélé à la fois plus simple à implémenter et à maintenir que les scripts qui étaient auparavant utilisés. Cependant, le nombre d'états restreint limite la complexité du comportement pouvant être atteinte. Tout de même, l'approche est adaptée pour un jeu de simulation où des centaines d'agents simples doivent interagir.

Si nous nous sommes concentrés sur une implémentation dans *SimRealEstate*, l'approche peut très bien être réutilisée pour modéliser d'autres PNJ. Nous avons vu qu'il suffisait d'une représentation d'état, d'actions et d'objectifs pour animer un personnage. Ainsi, les outils proposés peuvent être utilisés pour n'importe quel autre jeu.

CONCLUSION

Dans ce mémoire, nous avons abordé le problème de la planification à satisfaction partielle sous incertitude temporelle. Nous l'avons d'abord fait dans un cadre théorique, en constatant que les méthodes actuelles ne permettaient pas de tenir parfaitement compte de l'incertitude. Nous avons proposé une nouvelle approche d'abandons d'objectifs pour aborder le problème. Puis, dans le cadre plus pratique d'un jeu vidéo, nous nous sommes servi de la PSP pour améliorer la distribution d'objectifs à des PNJ dont le comportement est modélisé par des algorithmes de planification.

Ainsi, nous avons dans un premier temps décrit la planification automatisée. Afin de bien saisir les concepts en jeu, nous avons défini le modèle de la planification classique ainsi que les différentes hypothèses auxquelles il était sujet, notamment que tous les objectifs fournis à un agent doivent être satisfaits pour qu'un plan soit retourné par un planificateur.

Ensuite, nous avons discuté d'une complexification du problème : la planification à satisfaction partielle. La PSP est la planification automatisée lorsque l'hypothèse d'un but indissociable est abandonnée. En effet, ce modèle permet à des objectifs d'être insatisfaits dans un état final. Le problème de la PSP devient donc celui de la planification et de la sélection d'objectifs. Une revue des méthodes disponibles a montré que la PSP ne constituait pas un problème d'une complexité supérieure à celui de la planification classique et qu'un problème de PSP pouvait être traduit en un problème de planification classique.

Puis, nous avons abandonné deux autres hypothèses, celles du temps implicite et

du monde déterministe. En effet, nous nous sommes penchés sur le problème de la PSP, incluant des actions ayant des durées suivant des lois statistiques. Nous avons argumenté que les méthodes de résolution permettent uniquement de savoir qu'un objectif ne sera pas satisfait, mais pas quand cette décision a été prise. Ce temps est nécessaire à la composition de plans conditionnels, où l'ensemble d'objectifs est dynamique à l'exécution et où les branchements s'effectuent selon la décision de satisfaire ou non un objectif. Afin de régler ce problème, nous avons introduit la méthode par actions d'abandon, qui permet au planificateur d'abandonner des objectifs à tout moment durant le plan. Pour permettre un certain contrôle sur l'abandon, nous avons offert la possibilité d'imposer un temps limite à chacune des actions d'abandon. Ces temps limites assurent qu'un objectif sera satisfait s'il n'est pas abandonné avant un certain temps. Nous avons justifié la pertinence de cette approche avec la situation de robots sur Mars.

Nous avons par la suite intégré la PSP dans un jeu sérieux pour créer des personnages non-joueurs. La planification étant une méthode émergente pour le contrôle des PNJ, nous avons tenté de renforcer les liens entre la théorie et la pratique. Pour ce faire, nous avons d'abord constaté que les méthodes actuelles ne se penchaient pas sur le problème de la sélection des objectifs. Les planificateurs utilisés dans les jeux se font fournir les objectifs par un module de coordination externe à l'agent. Or, lorsque nous désirons que l'agent soit plus autonome, un problème se pose en deux temps : combien d'objectifs choisir et lesquels. Nous avons répondu à ces questions avec une approche utilisant la PSP, où l'agent sélectionne par lui-même ses objectifs et la quantité de ceux-ci. Cette approche a été testée avec succès dans *SimRealEstate*, un jeu sérieux de simulation immobilière où un joueur incarne un courtier immobilier et affronte d'autres courtiers. Ce sont ces derniers que nous avons modélisés grâce à la PSP.

Ainsi, ce mémoire compte deux contributions novatrices. La première, plus théo-

rique et concernant la résolution du problème sous incertitude temporelle, a mené à une publication à la conférence Canadian AI (Labranche et Beaudry, 2014a) ainsi qu'à une affiche à la conférence de l'Association for the Advancement of Artificial Intelligence (AAAI) (Labranche et Beaudry, 2014b). La deuxième, plus appliquée et concernant une application directe de la PSP, a mené à une publication à à la conférence IEEE Computational Intelligence and Games (Labranche *et al.*, 2014).

Les applications potentielles de la PSP ne se limitent pas à la robotique et aux jeux vidéo, mais s'étendent à de nombreux domaines devant jongler avec des ressources limitées.Ces limitations induisent nécessairement le besoin de compromis : c'est l'essence même de la PSP. Comme travaux futurs, nous entrevoyons son utilisation dans le domaine de la sécurité. Les situations de sécurité partagent une caractéristique commune : un immense territoire à sécuriser et des ressources limitées pour le faire, une couverture totale étant trop coûteuse (dans le cadre de la sécurité d'un aéroport par exemple, il ne peut pas y avoir une escouade canine anti-drogue pour chaque passager). Ce type de problèmes est donc directement relié à celui de la PSP, puisque de devoir couvrir un large territoire avec peu de ressources sous-entend de satisfaire partiellement la sécurité totale (Brown *et al.*, 2014).

Récemment, plusieurs chercheurs ont fait appel à la théorie des jeux pour aborder cette problématique, mais le défi majeur reste l'extensibilité : des algorithmes plus performants sont nécessaires (Tambe *et al.*, 2013). La planification automatisée étant déjà utilisée pour résoudre certains problèmes (Qian *et al.*, 2014), nous croyons que la planification à satisfaction partielle (PSP) pourrait augmenter le pouvoir d'expression des modèles. Ces éventuels travaux pourront s'inscrire dans la continuité de ceux présentés dans ce mémoire.

[Cette page a été laissée intentionnellement blanche]

RÉFÉRENCES

Aghighi, M. et Jonsson, P. (2014). Oversubscription planning : Complexity and compilability. Dans *Proceedings of the Conference of the Association for the Advancement Artificial Intelligence*, 2221–2227.

Amir, E. et Engelhardt, B. (2003). Factored planning. Dans *Proceedings of the International Joint Conference on Artificial Intelligence*, 929–935.

Beaudry, E., Kabanza, F. et Michaud, F. (2010a). Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 10–17.

Beaudry, E., Kabanza, F. et Michaud, F. (2010b). Planning with concurrency under resources and time uncertainty. Dans *Proceedings of the European Conference on Artificial Intelligence*, 217–222.

Beaudry, E., Kabanza, F. et Michaud, F. (2012). Using a classical forward search to solve temporal planning problems under uncertainty. Dans *Proceedings of the Workshops of the Association for the Advancement of Artificial Intelligence*, 2–8.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.

Benton, J., Do, M. et Kambhampati, S. (2009). Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence, 173*(5-6), 562–592.

Bienvenu, M., Fritz, C. et McIlraith, S. A. (2006). Planning with qualitative temporal preferences. Dans *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 134–144.

Boutilier, C., Dean, T. et Hanks, S. (1999). Decision-theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Bresina, J., Dearden, R., Meuleau, N., Smith, D. et Washington, R. (2002). Planning under continuous time and resource uncertainty : A challenge for AI. Dans *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 77–84.

Briel, M. V. D., Sanchez, R., Do, M. B. et Kambhampati, S. (2004). Effective approaches for partial satisfaction (over-subscription) planning. Dans *Proceedings of the Assocation for the Advancement of Artificial Intelligence*, 562–569.

Brown, M., An, B., Kiekintveld, C., Ordóñez, F. et Tambe, M. (2014). An extended study on multi-objective security games. *Autonomous Agents and Multi-agent Systems*, *28*(1), 31–71.

Champandard, A. J. (2013). *Planning in Games : An Overview and Lessons Learned.* Revue technique sous forme vidéo, Artificial Intelligence Game Developpers

Coles, A. J. (2012). Opportunistic branched plans to maximise utility in the presence of resource uncertainty. Dans *Proceedings of the European Conference on Artificial Intelligence*, 252–257.

Cushing, W. et Kambhampati, S. (2005). Replanning : A new perspective. Dans *Proceedings of the Poster Program of the International Conference on Planning and Scheduling*, 1–4.

Fox, M., Gerevini, A., Long, D. et Serina, I. (2006). Plan stability : Replanning versus plan repair. Dans *Proceedings of the International Conference on Planning and Scheduling*, 212–221.

Gerevini, A. E., Haslum, P., Long, D., Saetti, A. et Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition : PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, *173*(5-6), 619–668.

Ghallab, M., Nau, D. et Traverso, P. (2004). *Automated Planning : Theory and Practice*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

Haklay, M. M. et Weber, P. (2008). OpenStreetMap : User-generated street maps. *IEEE Pervasive Computing*, *7*(4), 12–18.

Hart, P. E., Nilsson, N. J. et Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Haslum, P. et Geffner, H. (2000). Admissible heuristics for optimal planning. Dans *Proceedings of the Artificial Intelligence Planning Systems*, 140–149.

Keyder, E. et Geffner, H. (2009). Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, *36*, 547–556.

Krogt, R. V. D. et Weerdt, M. D. (2005). Plan repair as an extension of planning. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 161–170.

Labranche, S. et Beaudry, E. (2014a). Partial satisfaction planning under time uncertainty with control on when objectives can be aborted. Dans *Proceedings of the Canadian Conference on Artificial Intelligence*, 167–178.

Labranche, S. et Beaudry, E. (2014b). Partial satisfaction planning under time uncertainty with control on when objectives can be aborted (version courtet). Dans *Proceedings of the Student Abstracts of the Conference of the Association for the Advancement of Artificial Intelligence*, 3114–3115.

Labranche, S., Sola, N., Callies, S. et Beaudry, E. (2014). Using partial satisfaction planning to automatically select NPCs' goals and generate plans in a simulation game. Dans *Proceedings of the IEEE Computational Intelligence and Games Conference*, 1–8.

Liu, D. et Schubert, L. (2014). Toward self-motivated, cognitive, continually planning agents. *Computational Intelligence*, 1–33.

Mausam et Weld, D. S. (2008). Concurrent probabilistic temporal planning. *Journal of Artificial Intelligence Research*, *31*, 33–82.

Menif, A., Guettier, C. et Cazenave, T. (2013). Planning and execution control architecture for infantry serious gaming. Dans *Proceedings of the Planning in Game Workshop of the International Conference on Automated Planning and Scheduling*, 31–34.

Meuleau, N., Benazera, E., Brafman, R., Hansen, E. et Mausam (2009). A heuristic search approach to planning with continuous resources in stochastic domains. *Journal of Artificial Intelligence Research*, *34*, 27–59.

Meuleau, N., Brafman, R. I. et Benazera, E. (2006). Stochastic over-subscription planning using hierarchies of MDPs. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 121–130.

Nebel, B. et Koehler, J. (1992). Plan modification versus plan generation : A complexity-theoretic perspective. Dans *Proceedings of the International Joint Conference on Artificial Intelligence*, 1436–1441.

Nigenda, R. S. et Kambhampati, S. (2005). Planning graph heuristics for selecting objectives in over-subscription planning problems. Dans *Proceedings of the Conference of the International Conference on Automated Planning and Scheduling*, 192–201.

Orkin, J. (2004). Symbolic representation of game world state : Toward real-time planning in games. Dans *Proceedings of the Challenges in Game Artificial Intelligence Workshop of the Conference of the Assocation for the Advancement of Artificial Intelligence*, 26–30.

Qian, Y., Haskell, W. B., Jiang, A. X. et Tambe, M. (2014). Online planning for optimal protector strategies in resource conservation games. Dans *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, 733–740.

Rafferty, A. N., Brunskill, E., Griffiths, T. L. et Shafto, P. (2011). Faster teaching by POMDP planning. Dans *Proceedings of the International Conference on Artificial Intelligence in Education*, 280–287.

Rasmusen, E. (2007). *Games and Information : An Introduction to Game Theory.* Malden, MA ; Oxford : Blackwell Publishers.

Raybourn, E. M. (2007). Applying simulation experience design methods to creating serious game-based adaptive training systems. *Interacting with Computers*, *19*(2), 206–214.

Roijers, D., Scharpff, J., Spaan, M., Oliehoek, F., Weerdt, M. D. et Whiteson, S. (2014). Bounded approximations for linear multi-objective planning under uncertainty. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 262–270.

Russell, S. J. et Norvig, P. (2009). *Artificial Intelligence : A Modern Approach.* Upper Saddle River, NJ, USA : Pearson Education.

Scharpff, J., Spaan, M. T. J., Volker, L. et de Weerdt, M. M. (2013). Coordinating stochastic multi-agent planning in a private values setting. Dans *Proceedings of the Distributed and Multi-Agent Planning Workshop of the International Conference on Automated Planning and Scheduling*, 17–25.

Singh, S. (1998). *Le dernier theoreme de Fermat.* Paris, France : Hachette Litteratures.

Smith, D. E. (2004). Choosing objectives in over-subscription planning. Dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 393–401.

Sola, N. (2015). *Estimation des connaissances d'un joueur-apprenant et génération de scénarios adaptés dans un jeu sérieux de simulation immobilière.* (Mémoire de maîtrise). Université du Québec à Montréal.

Talamadupula, K., Smith, D., Cushing, W. et Kambhampati, S. (2013). A Theory of intra-agent replanning. Dans *Proceedings of the Distributed and Multi-Agent Planning Workshop of the International Conference on Automated Planning and Scheduling.*

Tambe, M., Jiang, A. X., An, B. et Jain, M. (2013). Computational game theory for security : Progress and challenges. Dans *Spring Symposium on Applied Computational Game Theory of the Association for the Advancement of Artificial Intelligence*, 1–6.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*(236), 433–460.

Verweij, T. (2007). *A hierarchically-layered multiplayer bot system for a first-person shooter.* (Mémoire de maîtrise). Unviersité Vrije d'Amsterdam.