Chapitre 9

La planification en intelligence artificielle

RÉGIS SABBADIN (INRA), FLORENT TEICHTEIL-KÖNIGSBUCH (ONERA) et VINCENT VIDAL (ONERA)

Dans ce chapitre nous proposons une revue, non exhaustive, des travaux de la communauté de l'IA autour de la planification classique et de la planification dans l'incertain. Nous présentons tout d'abord le cadre de la planification classique STRIPS propositionnelle, puis ses extensions basées sur le langage de description de problèmes PDDL devenu un standard dans la communauté. Nous traiterons ensuite brièvement de la thématique de l'analyse structurelle de problèmes, à l'origine du développement de planificateurs performants, et des principaux algorithmes de planification classique et planificateurs associés. Ensuite, nous décrirons le cadre des processus décisionnels markoviens (PDM), issu du domaine de la recherche opérationnelle, mais que la communauté de l'intelligence artificielle a mobilisé pour la planification sous incertitude. Nous décrirons enfin des algorithmes novateurs de résolution (approchée ou non) de PDM proposés récemments, ainsi que certains progrès de l'IA en termes de représentation des connaissances (logique, réseaux bayésiens) ayant été mis à profit afin d'améliorer le pouvoir d'expression du modèle PDM traditionnel, au service de la planification dans l'incertain.

9.1 Introduction

La communauté de la *planification* en intelligence artificielle s'est intéressée depuis les années 1960 à la génération de *plans* (séquences d'actions) visant à atteindre un objectif fixé pour des problèmes exprimés dans un langage concis d'opérateurs de transformation d'état, souvent proche de la logique propositionnelle (voir [Ghallab et al., 2004], par exemple, pour une synthèse plus complète de cette famille d'approches). Depuis le premier système de planification à base d'opérateurs, le système GPS (« General Problem Solver ») de Newell et Simon [Newell and Simon, 1963] dont l'ambitieux objectif était de donner à une machine la capacité

de simuler le raisonnement humain, le domaine de la *planification classique* a connu un essor considérable (tout en revoyant ses ambitions à la baisse...) ; tant pour la richesse de modélisation des problèmes de planification que pour l'efficacité des systèmes de génération de plans (aussi appelés *planificateurs*). La planification classique repose sur deux hypothèses simplificatrices fortes : d'une part on dispose d'une connaissance parfaite, à tout instant, de l'état du système et des effets des actions ; et d'autre part les seules modifications de l'état du système proviennent de l'exécution des actions du plan. Ces hypothèses ont permis la conception de planificateurs capables de résoudre des problèmes de grande taille largement hors de portée de l'être humain.

Plus récemment, le domaine de la *planification dans l'incertain* s'est développé, proposant d'intégrer des actions à effet probabiliste puis des fonctions d'utilité additives sur les buts, conduisant à une famille d'approches pour la planification basées sur la théorie de la décision [Blythe, 1999]. Le cadre des processus décisionnels de Markov (PDM) est alors naturellement devenu l'approche privilégiée pour la représentation et la résolution de problèmes de planification dans l'incertain en intelligence artificielle. De nombreux travaux récents ont visé à améliorer ce cadre en le dotant du pouvoir expressif des langages de représentation traditionnellement utilisés en intelligence artificielle : logique, contraintes ou réseaux bayésiens. L'utilisation de tels langages de représentation fait « exploser » la complexité des algorithmes de résolution classiques des PDM. La résolution de ces problèmes *structurés* est ainsi devenue un défi pour la communauté de l'intelligence artificielle et de nombreuses approches, centralisées [Boutilier et al., 2000, Jensen, 2001, Givan et al., 2003] ou distribuées [Guestrin et al., 2003] ont été développées ces dernières années.

Dans ce chapitre nous offrons un bref panorama des approches développées ces dernières années pour la planification, en France et à l'étranger. Nous avons centré ce panorama sur les approches basées sur le modèle STRIPS [Fikes and Nilsson, 1971] et ses extensions pour la planification classique, et sur une représentation probabiliste de l'incertitude en planification et des critères issus de la *théorie de la décision*. Dans un premier temps, nous définirons le cadre de modélisation STRIPS pour la planification classique, et décrirons ses principales extensions basées sur le langage de description de problèmes de planification PDDL [McDermott et al., 1998]. Après avoir décrit les principales avancées dans la thématique de l'analyse structurelle des problèmes de planification, qui permet entre autres la conception d'heuristiques efficaces, nous établirons un panorama des principales techniques de résolution et planificateurs associés. Dans un second temps nous décrirons le cadre mathématique des *Processus Décisionnels de Markov* (PDM) [Puterman, 1994], cadre classique de représentation de problèmes de décision séquentielle dans l'incertain. Ensuite, nous montrerons comment la communauté de la planification s'est emparée de ce cadre et l'a étendu, afin de modéliser et résoudre des problèmes de planification dans l'incertain. Nous finirons ce chapitre par une brève description d'extensions du cadre des PDM, proposées ou adoptées par la communauté de l'intelligence artificielle : prise en compte de la non-observabilité de l'état du monde, apprentissage, représentations non probabilistes de l'incertitude...

9.2 La planification classique

9.2.1 Le cadre de la planification STRIPS propositionnelle

Le système fondateur STRIPS (« STanford Research Institute Problem Solver ») [Fikes and Nilsson, 1971] utilisé pour la contrôle du robot SHAKEY a posé les fondations sur lesquelles reposent encore aujourd'hui l'essentiel des travaux en planification classique. Nous nous plaçons ici dans le cadre de la planification STRIPS propositionnelle, ne faisant intervenir que des actions définies à l'aide d'un ensemble fini de symboles propositionnels atomiques. Un problème STRIPS propositionnel peut éventuellement être obtenu en instanciant des schémas d'actions décrits dans un langage plus riche, par exemple PDDL, à l'aide des constantes définies pour un problème particulier représentant les objets de l'univers.

Dans ce cadre, un *état du système* est un ensemble d'atomes issus d'un ensemble fini A représentant l'ensemble des faits de l'univers. Une *action* (ou *opérateur*, ce terme étant plus généralement utilisé pour désigner des schémas d'action contenant des variables pouvant être instanciées) est un triplet $a = \langle pr, ad, de \rangle$ où pr, ad et de dénotent des ensembles finis d'atomes de A ; $\mathrm{prec}(a)$, $\mathrm{add}(a)$, $\mathrm{del}(a)$ dénotent respectivement les ensembles pr, ad, de et représentent les *préconditions*, *ajouts* et *retraits* de a. Les préconditions définissent les conditions requises dans un état donné pour pouvoir y appliquer une action : une action a est applicable sur un état s si et seulement si $\mathrm{prec}(a) \subseteq s$. Les ajouts sont les faits que l'action va créer dans cet état, et les retraits sont ceux détruits par l'action ; l'état s' résultant de l'application d'une action a dans un état s étant défini par $s' = (s \setminus \mathrm{del}(a)) \cup \mathrm{add}(a)$. Un *problème de planification STRIPS propositionnel* est alors défini comme un quadruplet $\Pi = \langle A, O, I, G \rangle$ où A dénote un ensemble fini d'atomes, O dénote un ensemble fini d'actions construites à partir des atomes de A, $I \subseteq A$ dénote un ensemble fini d'atomes qui représente l'*état initial* du problème, et $G \subseteq A$ dénote un ensemble fini d'atomes qui représente le *but* du problème. Un *plan solution* est une sequence d'actions $(a_1, \ldots, a_n)$ telle que pour $s_0 = I$ et pour tout $i \in \{1, \ldots, n\}$, les états intermédiaires définis par $s_i = (s_{i-1} \setminus \mathrm{del}(a_i)) \cup \mathrm{add}(a_i)$ sont tels que $prec(a_i) \subseteq s_{i-1}$ et $G \subseteq s_n$.

L'objectif d'un planificateur va donc être de trouver un plan solution pour un problème donné, sachant que l'espace des états atteignables à partir d'un état initial est en général trop vaste pour être entièrement calculé et représenté en extension sous forme de graphe par exemple, où les noeuds représenteraient les états et les arcs représenteraient les actions permettant la transition d'un état vers un autre (un même état pouvant être atteint par différentes séquences d'actions). Par ailleurs, la complexité théorique du problème de décision concernant l'existence d'un plan solution pour la planification STRIPS propositionnelle est PSPACE-complet [Bylander, 1994] ; des classes polynomiales ont cependant été déterminées [Bäckström and Nebel, 1995], ainsi que des classes de problèmes pour lesquels certains algorithmes ont un comportement polynomial [Helmert, 2003, 2008]. De nombreux algorithmes ont été conçus pour réaliser une exploration partielle de l'espace des solutions, et sont généralement basés sur des techniques issues d'autres domaines de recherche : recherche heuristique, programmation par contraintes, satisfaction de bases de clauses, model checking, algorithmes évolutionnaires, etc. La *qualité* d'un plan solution est aussi un point important, qui influence fortement le choix d'une technique donnée. Pour la planification classique, l'objectif est en général de minimiser la longueur du plan solution ; nous verrons par la suite que d'autres critères existent, liés à l'amélioration de l'expressivité du langage de description de problèmes de

planification.

9.2.2 Le langage de description de problèmes PDDL

Conçu pour permettre une représentation commune des problèmes de planification pour la première compétition internationale de planification [1], le langage PDDL (« *Planning Domain Definition Language* ») a connu depuis de nombreuses évolutions. Ce langage et les compétitions associées ont permis de fédérer les recherches en planification classique, et grandement facilité la conception et le partage des meilleures techniques. Nous décrivons ci-dessous les principaux jalons de l'évolution du langage PDDL devenu un standard, liés aux compétitions successives :

PDDL 1.2 [McDermott et al., 1998] : version utilisée pour les compétitions de 1998 [McDermott, 2000] et 2000 [Bacchus, 2001]. Elle est inspirée du langage utilisé par le planificateur UCPOP [Penberthy and Weld, 1992] et définit la représentation de base des opérateurs (préconditions, effets) et des problèmes (état initial, but). La représentation STRIPS des opérateurs, où les préconditions, ajouts et retraits se réduisent à des conjonctions d'atomes, est étendue par les constructions du langage ADL [Pednault, 1989] : préconditions négatives et disjonctives, quantificateurs existentiels en précondition et universels en précondition et effets, effets conditionnels. Ces extensions permettent une plus grande concision dans la représentation des problèmes, mais l'expressivité reste la même par rapport au langage STRIPS de base, vers laquelle une transformation (potentiellement coûteuse) est possible [Gazen and Knoblock, 1997].

PDDL 2.1 [Fox and Long, 2003] : version utilisée pour la compétition de 2002 [Long and Fox, 2003]. Elle introduit deux extensions majeures : la prise en compte du temps, et les variables numériques. La finesse de prise en compte des aspects temporels se situe à mi-chemin de la représentation complexe à base de chroniques temporelles utilisée par exemple dans les planificateur IxTeT [Ghallab and Laruelle, 1994] et ASPEN [Chien et al., 2000] ou plus récemment dans le cadre CNT (« Constraint Networks on Timelines ») [Verfaillie et al., 2010, Pralet and Verfaillie, 2010], et de la représentation simplifiée du planificateur ZENO [Smith and Weld, 1999] étendant la relation d'exclusion mutuelle de GRAPHPLAN [Blum and Furst, 1997]. En PDDL, les préconditions des actions dont on définit la durée d'exécution doivent être vérifiées à l'instant de début ou de fin d'une action, et/ou dans l'intervalle de temps ouvert compris entre ces instants ; les effets sont produits soit à l'instant de début, soit à l'instant de fin. La concurrence temporelle entre actions est possible à condition qu'à tout instant de l'exécution d'un plan, un même atome ne soit à la fois produit ou requis par une action et retiré par une autre. La concurrence peut même être nécessaire pour l'obtention d'un plan dans certains problèmes, comme en témoignent divers travaux sur la notion de planification dite temporellement expressive [Cushing et al., 2007a,b] pour laquelle la complexité théorique du problème de l'existence d'un plan est EXPSPACE-complet [Rintanen, 2007]. Les variables numériques sont des quantités dont une valeur (minimale, maximale ou exacte) est requise en précondition, et qui sont modifiées par l'exécution de l'action par augmentation, diminution ou assignation. Ces variables numériques servent notamment

1. International Planning Competition : http ://ipc.icaps-conference.org

à représenter des ressources, notion couramment utilisée dans le domaine de l'ordonnancement [Laborie and Ghallab, 1995, Laborie, 2003]. Le langage permet aussi de définir une fonction objectif sur la qualité du plan, exprimée comme une combinaison linéaire de la durée globale d'exécution du plan (le *makespan*) et des variables numériques. L'extension de PDDL 2.1 dénommée PDDL+ [Fox and Long, 2006] inutilisée lors des compétitions étend les notions temporelles et numériques de PDDL 2.1 à la modélisation de processus continus, permettant entre autres la représentation d'évènements exogènes et une prise en compte plus fine de la modification des variables numériques (vue comme un processus continu linéaire) augmentant la possibilité de concurrence entre les actions.

PDDL 2.2 [Hoffmann and Edelkamp, 2004] : version utilisée pour la compétition de 2004 [Hoffmann and Edelkamp, 2005]. Deux extensions mineures sont proposées : les prédicats dérivés et les littéraux temporels initiaux. Les prédicats dérivés sont des atomes non affectés par les actions, dont l'apparition se produit lorsqu'une formule donnée (similaire à une précondition d'action) est vérifiée dans un état, permettant éventuellement le déclenchement de nouvelles actions. Ils fournissent entre autres un moyen concis pour représenter la fermeture transitive d'une relation, et augmentent strictement l'expressivité du langage : il peut s'avérer impossible de les compiler vers une version antérieure du langage [Thiébaux et al., 2003]. Les littéraux temporels initiaux permettent la représentation d'une forme réduite d'évènements exogènes, dont la date d'apparition est connue à l'avance et exprimée dans l'état initial du problème.

PDDL 3.0 [Gerevini and Long, 2005] : version utilisée pour la compétition de 2006 [Gerevini et al., 2009]. Elle introduit deux extensions importantes : les contraintes sur les trajectoires et les préférences sur les buts. Ces extensions sont motivées par le fait que dans les applications réelles, d'une part la forme d'un plan est au moins aussi importante que son obtention, et d'autre part il est souvent impossible de satisfaire entièrement tous les buts d'un problème. Les contraintes sur les trajectoires permettent de contraindre certains enchaînements d'actions et d'états intermédiaires atteints lors de l'exécution du plan, et peuvent être dures (à satisfaire obligatoirement) ou molles (leur satisfaction est associée à un poids entrant dans le calcul de la qualité globale du plan). Les préférences sur les buts permettent de différencier les buts dont l'obtention est obligatoire, des buts dont l'obtention ne l'est pas ; mais entrent, comme pour les contraintes de trajectoire molles, dans le calcul de la qualité du plan. Si les contraintes molles sont étudiées depuis longtemps en CSP [Meseguer et al., 2006], leur prise en compte en planification classique est plus récente et fragmentaire.

PDDL 3.1 [Helmert et al., 2008a] : version utilisée pour les compétitions de 2008 [Helmert et al., 2008b] et 2011. Elle introduit les variables d'état binaires, permettant une représentation plus concise des faits de l'univers, non plus énumérés comme des atomes de la logique propositionnelle ou du premier ordre mais comme des valeurs du domaine d'une variable d'état. Ces variables sont inspirées d'une part du formalisme SAS+ [Bäckström and Nebel, 1995] ayant permis la découverte de classes polynomiales de problèmes de planification, et d'autre part de leur intégration dans un langage plus flexible et expressif [Geffner, 2000]. L'engouement récent et fort de la communauté pour ces variables d'état, généralement extraites par une pré-compilation de problèmes exprimés en PDDL classique, est dû au fait qu'elles permettent une analyse plus fine

des problèmes de planification, pour en tirer notamment des heuristiques et autres éléments comme les *landmarks* (points de passage obligatoires) [Helmert, 2004, Richter et al., 2008, Karpas and Domshlak, 2009]. Un autre apport de cette version du langage est une forme réduite de variables numériques permettant d'associer un coût positif à chaque action, l'objectif étant alors de minimiser la somme des coûts des actions du plan.

Le schema d'action ci-dessous, venant du domaine TPP (« *Travelling and Purchase Problem* ») écrit pour la compétition de 2006 et inspiré d'un problème étudié en recherche opérationnelle, illustre certains des concepts que nous venons d'introduire :

```
(:durative-action drive
 :parameters (?t - truck ?from ?to - place)
 :duration (= ?duration (drive-time ?from ?to))
 :condition (and (at start (at ?t ?from)) (over all (connected ?from ?to))
                 (preference p-drive
                   (at start (forall (?g - goods)
                                (< (ready-to-load ?g ?from) 1)))))
 :effect (and (at start (not (at ?t ?from))) (at end (at ?t ?to))
              (at end (increase (total-cost) (drive-cost ?from ?to)))))
```

L'action `drive` déplace le camion `?t` du lieu `?from` vers le lieu `?to`. Les variables `?t`, `?from` et `?to` sont instanciées par des constantes représentant les objets d'un problème donné. La durée de ce déplacement dépend des lieux en question, et prend la valeur de la variable numérique `(drive-time ?from ?to)` définie dans l'état initial. Pour pouvoir appliquer cette action, le camion `?t` doit être au lieu `?from` à l'instant de début de l'action (`at start`) et les lieux `?from` et `?to` doivent rester connectés dans l'intervalle ouvert compris entre les instants de début et de fin (`over all`). La préférence `p-drive` est formulée pour représenter le fait que toutes les marchandises `?g` présentes sur le lieu de départ doivent avoir été vendues (`(ready-to-load ?g ?from)` a la valeur 0), donc ne pas être en attente d'un chargement dans un camion. Cette action a pour effet de déplacer le camion (`(at ?t ?from)` est retiré et `(at ?t ?to)` est ajouté), et le coût total du plan `(total-cost)` est augmenté du coût `(drive-cost ?from ?to)` du déplacement. La fonction objectif à minimiser peut être augmentée d'une valeur à définir si la préférence `p-drive` est violée durant l'exécution du plan.

9.2.3 L'analyse structurelle de problèmes en planification classique

L'analyse structurelle en planification consiste à extraire automatiquement des informations à partir de la description d'un problème, permettant de guider ou contraindre la recherche d'une solution vers les portions de l'espace de recherche les plus prometteuses. On trouve les prémisses de ces techniques dans [Pearl, 1983], où une simplification manuelle de la description d'un problème est proposée pour calculer l'heuristique de Manhattan pour le problème du taquin (cf. chapitre 2, volume 2 sur les jeux) ; plus tard, une technique automatique implémentée dans le planificateur UNPOP [McDermott, 1996] calcule par une régression à partir des buts ignorant les retraits des actions une estimation de la longueur d'un plan à partir de l'état courant de la recherche ; enfin, les idées introduites par le planificateur GRAPHPLAN [Blum and Furst, 1997] ont littéralement révolutionné le domaine.

La principale innovation apportée par GRAPHPLAN est la construction d'un graphe en chaînage avant à partir de l'état initial, dans lequel les interactions négatives entre actions (lorsqu'une action retire une précondition ou un ajout d'une autre action) ou atomes (lorsque deux atomes ne peuvent être produits que par des actions en interaction négative) ne sont considérées qu'entre paires d'actions ou d'atomes, et non pas sur l'ensemble des actions pouvant être appliquées sur un état donné. Ces paires d'actions ou d'atomes en interaction négative, aussi appelés *exclusions mutuelles* ou *mutex*, sont propagées dans le graphe et retardent l'insertion d'actions et d'atomes, fournissant là une estimation minorante de la longueur des plans pouvant les atteindre. Ce graphe étant construit en temps polynomial par rapport au nombre total d'actions et d'atomes et de leurs relations, il peut être pré-calculé avant la recherche d'une solution, voire même calculé à chaque étape d'une recherche en chaînage avant dans les espaces d'états. Le lecteur intéressé trouvera une description plus détaillée de GRAPHPLAN et des planificateurs qui s'en inspirent directement comme IPP [Koehler et al., 1997] et STAN [Long and Fox, 1999] dans [Vidal, 2001] ou [Ghallab et al., 2004].

Ces calculs d'heuristique ont ensuite été repris et généralisés dans [Bonet et al., 1997], qui proposent une méthode polynomiale d'estimation de l'heuristique, pour le problème relaxé par la suppression des listes de retrait des actions. Cette heuristique estime la longueur d'un plan qui produit un ensemble d'atomes, en définissant le coût d'un atome a à partir d'un état s par une fonction h_s qui ajoute 1 au minimum des coûts des ensembles de préconditions des actions qui produisent a. Le coût d'un ensemble d'atomes est alors défini à l'aide d'une fonction d'agrégation des coûts des atomes qui le composent. Le coût d'un atome peut être défini récursivement, pour un état s et un atome a, par :

$$h_s(a) = \begin{cases} 0 & \text{si} \quad a \in s \\ i+1 & \text{si} \quad min_{o \in O \mid a \in add(o)}[H_s(prec(o))] = i \\ \infty & \text{sinon} \end{cases} \tag{9.1}$$

où H_s est l'heuristique calculant le coût d'un ensemble d'atomes G. Plusieurs possibilités peuvent ainsi être envisagées : l'heuristique des systèmes ASP [Bonet et al., 1997], HSP, HSPr [Bonet and Geffner, 1999] et HSP2 [Bonet and Geffner, 2001] additionne le coût de chaque atome de l'ensemble G :

$$H_s^{add}(G) = \sum_{a \in G} h_s(a) \tag{9.2}$$

[Hoffmann and Nebel, 2001] proposent une amélioration de cette heuristique additive basée sur le fait que la résolution par GRAPHPLAN du problème relaxé est polynomiale. En effet, puisque les retraits des actions ont été suprimés, le graphe de planification ne contient aucune exclusion mutuelle, donc la procédure d'extraction de la solution est triviale (sans aucun retour arrière). Le coût d'un ensemble d'atomes est alors constitué par la somme des actions du plan parallèle qui produit ces atomes. Elle prend donc en compte les effets positifs des actions, contrairement à l'heuristique additive qui suppose que tous les sous-buts sont indépendants. Les excellentes performance du planificateur FF lors de la troisième compétition internationale de planification a inspiré la définition de plusieurs variantes comme $h^{FF/a}$ et h^{sa} [Keyder and Geffner, 2008].

Cependant, ces heuristiques ne sont pas admissibles, car elles surestiment le nombre d'actions nécessaires pour résoudre le problème relaxé. Une possibilité pour les rendre admissibles est de remplacer la somme dans l'équation 9.2 par le maximum du coût de chaque atome de l'ensemble G [Bonet and Geffner, 2001], mais celle-ci n'est pas très informative :

$$H_s^{max}(G) = max_{a \in G} h_s(a) \tag{9.3}$$

Ces heuristiques ont été étendues pour la planification optimale et la planification temporelle [Haslum and Geffner, 2000, 2001] tout en généralisant le calcul de mutex introduit par GRAPHPLAN à des ensembles d'actions et d'atomes de taille quelconque.

Les heuristiques dites *landmarks* [Koehler and Hoffmann, 2000, Hoffmann et al., 2004], ont pour but d'estimer un ordre entre les atomes nécessaires (ou *landmarks*) dans toute solution possible d'un problème de planification. Un atome est dit nécessaire s'il est ajouté par au moins une action quelle que soit la solution du problème. La recherche de ce type d'atome est PSPACE-difficile en général [Hoffmann et al., 2004]. L'idée est donc d'approximer cette recherche, en utilisant par exemple la méthode de chaînage arrière proposée dans [Hoffmann et al., 2004], les graphes de transition de domaines [Richter et al., 2008] basés sur la représentation SAS+ [Bäckström and Nebel, 1995], l'utilisation de l'heuristique h_s comme proposé dans [Vidal and Geffner, 2005] ou encore l'utilisation d'un arbre ET/OU [Keyder et al., 2010]. Pour un état s, une heuristique dite *landmarks-possible* h_s^L peut être définie par $h_s^L = n - m + k$, où n est le nombre total de *landmarks* estimé pour le problème, m le nombre de *landmarks* atteints par le plan menant à s et k le nombre de *landmarks* déjà atteints mais devant être satisfaits une nouvelle fois. Plusieurs autres variantes ont été proposées dans [Karpas and Domshlak, 2009].

Certains travaux, comme [Helmert and Geffner, 2008] et [Helmert and Domshlak, 2009] proposent des généralisations et unification d'heuristiques existantes. Une littérature très abondante sur le sujet de l'analyse structurelle et du calcul d'heuristiques existe, nous n'avons pu en donner ici qu'une vision très partielle.

9.2.4 Principaux algorithmes et planificateurs

Comme évoqué précédemment, la planification classique emprunte des méthodes de résolution à de nombreux autres domaines de recherche. Suivant l'objectif d'un planificateur – trouver un plan optimal pour un critère donné, optimiser ce critère sans prouver l'optimalité, trouver un plan rapidement sans optimisation –, certaines méthodes s'avèrent plus adaptées que d'autres. Quelles que soient les méthodes de résolution choisies, les techniques d'analyse structurelle comme celles décrites précédemment s'avèrent indispensables pour obtenir de bonnes performances.

Les premiers types d'algorithmes parmi les plus employés sont les algorithmes de recherche heuristique tels A* [Hart et al., 1968], IDA* [Korf, 1985] et leurs variantes comme weighted-A* [Pohl, 1970] ou EHC (« *Enforced Hill Climbing* ») [Hoffmann and Nebel, 2001] (cf. chapitre 1, volume 2, sur la recherche heuristique). Ils sont utilisés pour effectuer une recherche dans les espaces d'états en chaînage avant ou arrière, ou bien dans les espaces de plans [Weld, 1994] où un nœud représente un plan partiel et un arc une opération de modification d'un plan partiel : introduction d'une action, relation de précédence entre actions, protection d'un lien causal. Ces algorithmes sont utilisés soit pour la planification non optimale, soit pour la planification optimale en nombre d'actions ou en coût total. Pour la planification non optimale, on peut citer les planificateurs HSP, HSPr [Bonet and Geffner, 1999] et HSP2 [Bonet and Geffner, 2001] qui utilisent h_s^{add} ; RePOP [Nguyen and Kambhampati, 2001] et VHPOP [Younes and Simmons, 2003] utilisent cette heuristique dans les espaces de plans partiels ; le planificateur FF [Hoffmann and Nebel, 2001] calcule à chaque état la longueur d'un

plan pour le problème relaxé ; le planificateur YAHSP [Vidal, 2004] ajoutant au précédent la construction gloutonne d'un *plan anticipé* dont l'état résultant est ajouté à la liste des noeuds à développer ; Metric-FF [Hoffmann, 2002] et SAPA [Do and Kambhampati, 2003] pour la planification avec variables numériques ; SGPLAN [Chen et al., 2006] qui partitionne un problème en plusieurs sous problèmes plus simples ; CRICKEY [Coles et al., 2008] et COLIN [Coles et al., 2009] pour la planification temporellement expressive ; Macro-FF [Botea et al., 2005] et Marvin [Coles and Smith, 2007] qui calculent des macros-opérateurs ; TFD [Helmert, 2006] qui calcule une heuristique basée sur les graphes de transition de domaines ; et enfin LAMA [Richter and Westphal, 2010] qui utilise une heuristique basée sur les landmarks. Concernant la planification optimale en nombre d'actions ou en coût, on peut citer GAMER [Edelkamp and Kissmann, 2008] qui utilise aussi des techniques de model-cheking comme les BDDs, ou des variations de TFD avec des heuristiques améliorées [Helmert et al., 2007, Cai et al., 2009]. Enfin, on peut citer le planificateur TP4 [Haslum, 2006] pour la planification temporelle optimale.

Les approches à base de contraintes, comme la programmation par contraintes (PPC) (cf. chapitre 6, volume 2, sur la PPC) ou la satisfaction de bases de clauses (SAT) (cf. chapitre 5, volume 2, sur le problème SAT), sont très utilisées en planification. Les planificateurs SATPLAN [Kautz et al., 1996] et BLACKBOX [Kautz and Selman, 1999] encodent un problème sous forme d'une base de clauses pour un horizon donné et utilisent ensuite un prouveur SAT pour trouver une solution (voir [Maris et al., 2008] pour plus d'informations sur la planification SAT). Des améliorations des codages et planificateurs SAT ont récemment été proposées [Rintanen, 2010, Huang et al., 2010]. Ces planificateurs calculent des plans parallèles optimaux, cas particulier de planification temporelle avec durées uniformes. Le planificateur CFDP [Grandcolas and Pain-Barre, 2007] utilise la PPC sur une structure inspirée du graphe de planification avec des règles d'élagage adaptées à la planification séquentielle optimale. GP-CSP [Do and Kambhampati, 2001] encode un problème sous forme de CSP et calcule un plan parallèle optimal. CPT [Vidal and Geffner, 2006] est un planificateur temporel optimal qui encode un problème en CSP suivant un schéma basé sur la planification dans les espaces de plans et introduit de nombreuses règles d'élagage basées notamment sur les actions qui ne font pas encore partie d'un plan partiel. On peut citer enfin le planificateur TLP-GP [Maris and Régnier, 2008] qui effectue une recherche dans une structure inspirée du graphe de planification encodant des contraintes temporelles, pour la planification temporellement expressive.

De façon ponctuelle, certaines autres techniques sont utilisées en planification classique : les réseaux de Petri pour la planification temporelle optimale [Hickmott et al., 2007], les automates pondérés pour la planification factorisée optimale en coût [Fabre et al., 2010], les algorithmes évolutionnaires [Bibai et al., 2010] permettant d'optimiser la qualité des plans produits par un planificateur sous-jacent (YAHSP) par l'évolution d'une population d'individus représentant des découpages en états successifs à atteindre.

Pour finir, nous pouvons mentionner l'intérêt croissant de la communauté pour la prise en compte des évolutions récentes du matériel de calcul vers les architectures à base de processeurs multi-cœurs à mémoire partagée et les grilles de calcul à mémoire distribuée. Plusieurs approches ont été proposées ; notamment [Burns et al., 2009, Vidal et al., 2010] pour la planification non optimale sur machines multi-cœurs, le premier par un découpage en blocs de la liste des nœuds à développer et le second pour un accès concurrent de multiples *threads* à cette liste ; [Kishimoto et al., 2009] pour la planification sequentielle optimale en coût pour ar-

chitectures à mémoire distribuée, par distribution des nœuds suivant une fonction de hachage ; et enfin [Valenzano et al., 2010] pour une approche basée sur un portefeuille de variations paramétriques d'un même algorithme s'exécutant en concurrence.

9.3 Planification probabiliste en représentation intensionnelle

Dans cette section, nous allons montrer comment le cadre de la planification déterministe a été étendu afin de prendre en compte des actions à effets incertains et des préférences sur les buts. Mais avant cela, nous décrivons brièvement le cadre des processus décisionnels de Markov (PDM) qui a servi de base sémantique à certaines des approches proposées pour la planification dans l'incertain. Les PDM sont une approche séquentielle de la décision dans l'incertain (cf. chapitre 14, volume 1), où un agent reçoit des récompenses en interagissant avec un environnement probabiliste.

9.3.1 Le cadre des processus décisionnels de Markov

Dans le cadre des *Processus Décisionnels de Markov (PDM)* [Puterman, 1994], l'espace des états est décrit *en extension* par un ensemble $\mathcal{X}$ et l'espace des actions par un second ensemble $\mathcal{A}$.

Le résultat de l'application d'une suite d'actions $(a_0, \ldots, a_{T-1})$ est une *trajectoire* du système, $\tau = (x_0, a_0, \ldots, x_{T-1}, a_{T-1}, x_T)$. Une autre spécificité du cadre des PDM est qu'il permet de modéliser une forme d'incertitude (probabiliste) sur l'effet des actions. Aussi, une suite d'actions fixée $(a_0, \ldots, a_{T-1})$ peut résulter en différentes trajectoires, avec une distribution de probabilité connue $p(\tau|x_0, a_0, \ldots, a_{T-1})$. Le cadre des processus décisionnels de Markov diffère également de celui de la planification classique en ce sens qu'il permet de mesurer *l'utilité* d'une trajectoire τ non seulement par le fait qu'elle atteigne ou non un état *but*, mais par la prise en compte (additive) de degrés de satisfaction associés à toutes les transitions d'état (x_t, a_t, x_{t+1}).

Puisque les effets des actions ne sont plus déterministes, les effets d'un plan d'actions (une simple suite d'actions) ne peuvent être connus à l'avance. Dans ce cas, des plans *réactifs*, qui définissent l'action courante à appliquer en fonction de l'observation de l'état du système, sont plus efficaces que des plans définis a priori. Dans le cadre des PDM, ces plans réactifs sont appelés *stratégies*, ou *politiques*. Une *politique* $\delta = \{\delta_t\}_{t\in 0\ldots T-1}$ est un ensemble de fonctions associant à toute trajectoire partielle $(x_0, \ldots, x_t)$ l'action $a_t = \delta_t(x_0, \ldots, x_t) \in A$ appliquée par cette politique. Si l'on définit maintenant $p(\tau|x_0, \delta)$, la probabilité de suivre la trajectoire τ en appliquant la politique δ à partir de x_0 et $u(\tau)$ la récompense obtenue lorsque l'on suit la trajectoire τ, nous pouvons définir formellement l'utilité espérée d'une politique δ :

$$EU_\delta(x_0) = \sum_\tau p(\tau|x_0, \delta).u(\tau). \tag{9.4}$$

Ce critère permet de classer les politiques par ordre d'intérêt.

Résoudre un problème de décision séquentielle dans l'incertain consiste à déterminer une politique δ^* d'utilité espérée maximale, pour un état initial x_0, fixé ($EU_{\delta^*}(x_0) \geq EU_\delta(x_0), \forall\delta$), ou pour un ensemble d'états initiaux. Le cadre des *processus décisionnels de Markov* (PDM)

propose des algorithmes efficaces de calcul de politiques optimales, sous réserve que l'incertitude (mesurée par p) et les préférences (mesurées par u) sur les trajectoires satisfassent un certain nombre d'hypothèses.

Dans le cadre des PDM, l'hypothèse suivante est faite sur la probabilité conditionnelle $p(\tau|x_0, \delta)$:

$$p(\tau|x_0, \delta) = \prod_{t=0}^{T-1} p_t(x_{t+1}|x_t, \delta_t(x_0, \ldots, x_t) = a_t). \tag{9.5}$$

Cette hypothèse revient à dire que le processus stochastique défini à partir d'une politique fixée δ est *markovien*.

De même, la fonction d'utilité sur les conséquences est construite à partir de la somme de récompenses attachées aux transitions entre états :

$$\forall \tau = (x_0, a_0, \ldots, x_{T-1}, a_{T-1}, x_T), u(\tau) = \sum_{t=0}^{T-1} r_t(x_t, a_t, x_{t+1}). \tag{9.6}$$

Cette forme additive suppose que la trajectoire considérée est de longueur finie. Néanmoins, il est parfois plus commode de représenter et résoudre des problèmes de décision séquentielle dans l'incertain sur un horizon de temps infini, à condition que ces problèmes soient « stationnaires » (les probabilités de transition p_t et les récompenses r_t sont indépendantes du temps). Rien ne garantit alors que l'utilité $u(\tau)$ d'une trajectoire infinie, décrite par l'équation (9.6), soit finie. C'est pourquoi l'une des deux fonctions d'utilité suivantes est utilisée en général pour les problèmes dont l'horizon est infini, plutôt que la forme (9.6) :

$$u(\tau) = \lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t . r(x_t, a_t, x_{t+1}), 0 < \gamma < 1, \tag{9.7}$$

$$u(\tau) = \lim_{T\to\infty} \frac{1}{T}\Big(\sum_{t=0}^{T-1} r(x_t, a_t, x_{t+1})\Big). \tag{9.8}$$

Si la fonction de récompense r est bornée, les deux expressions (9.7) et (9.8) sont bien définies, quelle que soit la longueur de la trajectoire τ. La fonction d'utilité (9.7) est appelée *critère gamma-pondéré* alors que la fonction (9.8) est appelée *critère moyen*.

Résoudre un problème de décision séquentielle dans l'incertain revient à *optimiser* le choix d'une politique. En d'autres termes, on cherche à calculer, pour un problème de décision séquentielle dans l'incertain $(\mathcal{X}, \mathcal{A}, p, r, H)$, une politique δ^* telle que :

$$EU_{\delta^*}(x_0) \geq EU_\delta(x_0), \forall \delta, \forall x_0 \in \mathcal{X}. \tag{9.9}$$

A priori, rien n'indique qu'une telle politique optimale existe si nous exigeons que *la même politique* δ^* soit optimale en tout état de départ x_0. Néanmoins, on peut montrer qu'une telle politique optimale existe, pour les trois critères évoqués [Bellman, 1957, Puterman, 1994]. Qui plus est, dans le cas où l'horizon est infini, il existe une politique optimale *stationnaire* (indépendante de t), $\delta^* : \mathcal{X} \to \mathcal{A}$. δ^* choisit l'action optimale en fonction du seul état courant.

Le problème d'optimisation défini par (9.9) est classiquement résolu par des méthodes de type *programmation dynamique stochastique* [Bellman, 1957, Bertsekas, 1987, Puterman,

1994]. L'algorithme de *recherche arrière* est utilisé pour résoudre des problèmes à horizon fini et les algorithmes *d'itération de la politique* et *d'itération de la valeur* sont les plus couramment utilisés pour le critère gamma-pondéré.

9.3.2 Modèles intensionnels de PDM

Le cadre classique des PDM repose sur une représentation explicite des états, c'est-à-dire où chaque état est énuméré dans une table [Puterman, 1994]. Cette modélisation présente deux inconvénients non négligeables pour des application réalistes : les états sont souvent décrits par un ensemble de variables à domaines discrets ou continus, et, à cause de l'explosion combinatoire due au nombre de combinaisons possibles des valeurs des variables, les états ne peuvent pas être énumérés avant la résolution du problème.

Ainsi, depuis une dizaine d'années, des modèles de PDM basés sur une représentation factorisée de l'espace d'états par variables ont été développés. Ces modèles sont dits *intensionnels*, car ni les états ni les transitions ni les récompenses ne sont construits a priori, mais sont définis par des formules logiques portant sur les variables d'état du modèle. Chaque formule peut être *instanciée* en remplaçant les variables d'état par leurs valeurs à un instant donné, ce qui permet de construire au besoin l'état correspondant à ces variables instanciées, ainsi que les transitions ou les récompenses définies sur cet état.

Nous présentons ici les deux principaux modèles intensionnels de PDM : les réseaux bayésiens dynamiques [Dean and Kanazawa, 1990, Boutilier et al., 2000, 1998, Hoey et al., 1999, St-Aubin et al., 2000, Feng and Hansen, 2002, Feng et al., 2003] et les modèles STRIPS probabilistes [Younes et al., 2005, Yoon et al., 2007, 2008, Teichteil-Königsbuch et al., 2010, Buffet and Aberdeen, 2009, Kuter and Nau, 2005, Kolobov et al., 2010a, Joshi et al., 2010].

Réseaux bayésiens dynamiques

Un réseau bayésien dynamique (cf. figure 1, chapitre 11, volume 1 et [Dean and Kanazawa, 1990]) est un graphe orienté dont les nœuds sont des variables d'état rangées par niveaux temporels, et les arcs indiquent les dépendances bayésiennes entre les variables d'état. Chaque niveau t contient toutes les variables d'état à l'instant t, et il ne peut y avoir d'arcs qu'entre variables de deux niveaux successifs ou d'un même niveau : par conséquent, les transitions entre deux états successifs du système — représentés par leurs variables d'état aux instants t et $t+1$ — sont nécessairement markoviennes. Si les transitions ne dépendent pas du temps, le réseau bayésien ne contient que deux niveaux t et $t+1$. Dans ce cas particulier dit *homogène*, il convient de noter les variables à l'instant t sous la forme $(X_1, \cdots, X_n)$ et les variables à l'instant $t+1$ sous la forme primée $(X'_1, \cdots, X'_n)$.

Les arcs du réseau bayésien dynamique renseignent sur la dépendance événementielle des variables d'état entre deux instants successifs, mais pas sur la probabilité de ces événements. Une table de probabilité est donc associée à tout réseau bayésien dynamique pour définir les probabilités de chaque variable X'_m à l'instant $t+1$. Formellement, si une variable m à l'instant $t+1$ dépend de variables $X_{j_1}, \cdots, X_{j_k}$ à l'instant t et de variables $X'_{i_1}, \cdots, X'_{i_l}$ à l'instant $t+1$, alors cela se traduit par des arcs des variables $X_{j_1}, \cdots, X_{j_k}$ et $X'_{i_1}, \cdots, X'_{i_l}$ vers la variable X'_m, et l'entrée $P(X'_m \mid X'_{i_1}, \cdots, X'_{i_l}, X_{j_1}, \cdots, X_{j_k})$ dans la table de probabilité.

Toutefois, la dépendance entre les variables dépend de leurs valeurs : suivant la valeur de X'_m, l'événement probabiliste associé à cette valeur ne dépend pas des mêmes variables.

Ainsi, le réseau bayésien n'indique pour chaque variable primée qu'une dépendance maximale parmi toutes les valeurs de la variable primée. Il en est de même de la probabilité $P(X'_m \mid X'_{i_1}, \cdots, X'_{i_l}, X_{j_1}, \cdots, X_{j_k})$, dont les variables de dépendance pour une valeur donnée de X'_m appartiennent à un sous-ensemble de $\{X'_{i_1}, \cdots, X'_{i_l}, X_{j_1}, \cdots, X_{j_k}\}$. La structure de cette probabilité peut être assez compliquée, si bien qu'elle est elle-même représentée par un arbre de décision ou, plus généralement, par un diagramme de décision algébrique [Bahar et al., 1997].

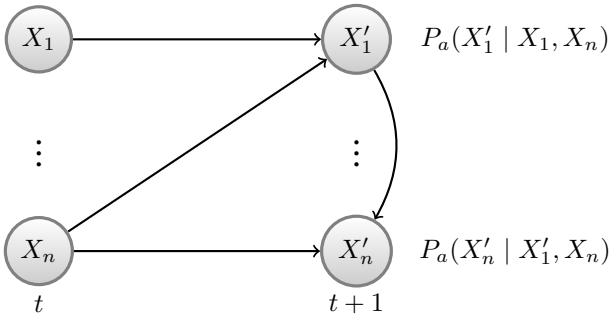

FIGURE 1: Exemple de réseau bayésien dynamique homogène

L'extension des réseaux bayésiens dynamiques aux PDM intensionnels est relativement simple. À chaque action du PDM est associé un réseau bayésien dynamique qui représente les probabilités de transition entre les variables d'état à deux instants successifs. La structure des récompenses, qui est généralement différente de celle des probabilités de transition, est définie par un autre réseau bayésien dynamique dont les arcs représentent les récompenses gagnées ou perdues par l'agent autonome pour chaque valeur des variables d'état à l'instant $t+1$ en fonction des variables d'état à l'instant t ou $t+1$. Le modèle suppose donc que la récompense gagnée ou perdue en arrivant dans un état s' est égale à la somme des récompenses gagnées ou perdues pour chaque valeur des variables d'état de s'.

Modèle STRIPS probabiliste

Depuis 2004, sous l'impulsion de la première compétition internationale de planification probabiliste [Younes et al., 2005], les communautés de planification classique et de processus décisionnels de Markov (PDM) se sont considérablement rapprochées. Partant du langage PDDL développé pour les compétitions de planification classique [Fox and Long, 2003], des effets probabilistes ont été rajoutés au langage afin de modéliser une sous-classe des PDM dans ce langage, nommé PPDDL (« *Probabilistic PDDL* »). Il en résulte une modélisation intensionnelle des PDM sous forme d'opérateurs STRIPS [Fikes and Nilsson, 1971] probabilistes. De nombreux planificateurs basés sur ce langage ont été développés depuis, permettant en l'espace de quelques années d'augmenter de plusieurs ordres de grandeur la taille des problèmes résolus et de diminuer tout autant les temps de calcul et la mémoire nécessaire.

Le langage PPDDL, bâti sur son prédécesseur PDDL, reprend et étend les éléments suivants :

— domaine défini en intension à l'aide de prédicats logiques ;
— définition séparée du domaine, décrivant la dynamique des actions, et du problème, définissant les objets du monde qui instancient les prédicats du domaine ;
— variables d'état représentées sous forme de prédicats logiques ;
— actions représentées sous forme de prédicats logiques ;
— domaines d'application des actions définis à l'aide d'une formule logique du premier ordre à valeur booléenne ;
— effets probabilistes des actions définis à l'aide de formules logiques du premier ordre, décrivant pour chaque effet probabiliste les variables devenant vraies ou fausses, pondérées par les probabilités d'occurrence des effets correspondants ;
— ensemble d'états initiaux possibles représentés par un ensemble de prédicats instanciés initialement vrais ;
— récompenses associées à chaque effet probabiliste, si le problème à résoudre consiste à maximiser le critère gamma-pondéré ;
— ensemble d'états buts définis sous la forme d'une formule logique du premier ordre, si le problème à résoudre consiste à atteindre le plus rapidement possible un ensemble de buts avec la plus grande probabilité ou le plus faible coût moyen.

De plus haut niveau que les réseaux bayésiens dynamiques, le langage PPDDL est moins expressif, mais il peut être exponentiellement plus compact. En effet, chaque variable d'un réseau bayésien dynamique correspond à une instanciation d'un prédicat du langage PPDDL pour un ensemble d'objets du problème. Considérons par exemple un problème consistant à empiler des cubes dans un ordre particulier. Le prédicat `on(?b1,?b2)` indique si un bloc `b1` quelconque est sur un bloc `b2` quelconque. En supposant qu'il y ait n blocs dans le problème, ce seul prédicat représente à lui seul $n(n-1)$ variables d'état du problème. Il est clair que le nombre de variables instanciées (telles que définies dans un réseau bayésien dynamique) augmente exponentiellement avec le nombre d'objets du problème. Toutefois, ce gain en compacité suppose que les variables du problème sont binaires et indépendantes sémantiquement, ce qui n'est pas le cas de nombreux problèmes : dans le « monde des grilles » par exemple, il est impossible que l'agent se trouve indépendamment dans plusieurs cases au même instant.

Le langage PPDDL suppose que les transitions sont indépendantes du temps et, contrairement aux réseaux bayésiens dynamiques, les probabilités de transition portent sur plusieurs variables d'état primées (à l'instant suivant) à la fois. Reprenons l'exemple du « monde des blocs », où il est possible qu'un bloc `b1` explose ou qu'un bloc `b2` soit détruit quand l'agent tente d'empiler `b2` sur `b1` en appliquant l'action ci-dessous :

```
(:action put-on-block
   :parameters (?b1 ?b2 - block)
   :precondition (and (holding ?b1) (clear ?b2) (no-destroyed ?b2))
   :effect (and (emptyhand) (on ?b1 ?b2) (not (holding ?b1)) (not (clear ?b2))
                (probabilistic 1/10 (when (no-detonated ?b1)
                     (and (not (no-destroyed ?b2)) (not (no-detonated ?b1)))))))
```

Le code PPDDL précédent indique que l'action `put-on-block`, quels que soient les blocs `b1` et `b2`, n'est applicable que si l'agent tient `b1` dans sa main, qu'il n'y a aucun bloc sur `b2` et que `b2` n'est pas détruit. Cette action a pour effet de libérer la main de l'agent, de mettre `b1` sur `b2` et, avec une probabilité de 0.1, et si le bloc `b1` n'avait pas explosé avant le début de l'action, à la fois de ne pas détruire le bloc `b2` et de ne pas faire exploser le bloc `b1`.

Des traducteurs automatiques de PPDDL vers des réseaux bayésiens dynamiques ont été développés [Younes et al., 2005], mais ils se heurtent à une combinatoire exponentielle de la transformation dans le pire cas, ainsi qu'à un calcul difficile des probabilités de transition isolées pour chaque variable à l'instant suivant. Ce dernier point nécessite d'ailleurs, dans de nombreux cas, de rajouter au réseau bayésien dynamique des variables d'état fictives qui représentent le couplage synchrone entre les variables réelles du problème, augmentant d'autant plus la complexité de la résolution du PDM.

Avant de clore ce paragraphe, notons l'existence d'un nouveau langage de modélisation des PDM, utilisé depuis la compétition internationale de planification probabiliste en 2011. Il s'agit du langage RDDL (« *Relational Dynamic influence Diagram Language* ») [Sanner, 2010], basé sur la logique du premier ordre comme PPDDL, mais non centré sur les actions contrairement à ce dernier. Ainsi, il est plus aisé de modéliser en RDDL des effets communs à toutes les actions – ou indépendants des actions. D'un point de vue sémantique, les expressions de RDDL se rapprochent des réseaux bayésiens dynamiques, dans la mesure où les variables d'état sont mises à jour de manière individuelle suite aux effets des actions (contrairement à PPDDL, où les variables d'état sont mises à jour par groupes).

9.3.3 Algorithmes et planificateurs

De nombreux algorithmes ont été développés pour résoudre des PDM intensionnels. Ils se rangent en deux catégories fondamentalement différentes au niveau des méthodes utilisées : les approches probabilistes [Boutilier et al., 2000, Hoey et al., 1999, St-Aubin et al., 2000, Feng and Hansen, 2002, Feng et al., 2003, Joshi et al., 2010, Kuter and Nau, 2005, Buffet and Aberdeen, 2009], qui étendent les méthodes de résolution des PDM classiques aux PDM intensionnels, et les approches déterministes [Yoon et al., 2007, Teichteil-Königsbuch et al., 2010, Yoon et al., 2008, Kolobov et al., 2010a], qui réduisent la résolution du PDM intensionnel à plusieurs résolutions de problèmes de planification classique déterministe.

Approches probabilistes

Cette catégorie d'algorithmes repose essentiellement sur l'utilisation de diagrammes de décision algébriques [Bahar et al., 1997], qui sont des structures de données compactes pour représenter des fonctions de variables booléennes à valeurs réelles. Ils permettent de représenter efficacement les tables de probabilité ou de récompense des réseaux bayésiens dynamiques, ainsi que la fonction de valeur optimisée. Ils sont souvent utilisés conjointement aux diagrammes de décision binaires [Bryant, 1986], qui représentent des fonctions booléennes à valeurs booléennes, et qui sont donc particulièrement indiqués pour les préconditions des actions ou la politique d'action optimisée.

De nombreux algorithmes de résolution de PDM classiques ont été adaptés aux PDM intensionnels modélisés sous forme de réseaux bayésiens dynamiques, en redéfinissant les opérateurs algébriques de l'équation de Bellman sur la base des opérateurs fournis par les diagrammes de décision algébriques ou binaires. Contrairement aux PDM classiques, les PDM intensionnels basés sur les diagrammes de décision travaillent non pas explicitement sur des états individuels, mais en intension sur plusieurs états à la fois à l'aide de formules logiques portant sur les variables d'état du problème. La principale difficulté de l'adaptation des algorithmes classiques aux PDM intensionnels basés sur les diagrammes de décision réside donc

dans la nécessité de repenser globalement l'équation de Bellman (sur des ensembles d'états) et non localement (sur chaque état individuel). Les principaux algorithmes développés dans ce cadre sont : SPUDD [Hoey et al., 1999] (adaptation de l'itération de la valeur), APRICODD [St-Aubin et al., 2000] (approximation de SPUDD), sLAO* [Feng and Hansen, 2002], sRTDP [Feng et al., 2003] et FODD-PLANNER [Joshi et al., 2010]. Ce dernier repose sur les diagrammes de décision du premier ordre [Wang et al., 2008], qui sont une extension des diagrammes de décision pour représenter des fonctions à valeurs booléennes dont les variables sont des formules logiques portant sur les variables d'état du problème. Cette nouvelle structure, plus informée que les diagrammes de décision sur la structure du PDM à résoudre, permet de trouver des solutions optimales en utilisant moins de ressources de calcul.

Néanmoins, les approches reposant sur les diagrammes de décision souffrent souvent de quelques inconvénients qui peuvent devenir rédhibitoires en pratique. Premièrement, ces structures de données nécessitent d'ordonner les variables dans la structure pour maintenir la structure compacte et garantir des temps d'accès convenables. Or, l'ordre des variables influe grandement sur la taille de la structure et donc sur la performance des opérations sous-jacentes ; en pratique, l'utilisateur d'un algorithme de résolution de PDM intensionnel basé sur ces structures doit fournir un ordre de variables garantissant de bonnes performances pour chaque problème, ce qui n'est pas toujours aisé. Deuxièmement, les algorithmes de résolution de PDM basés sur ces structures de données, et donc sur le modèle des réseaux bayésiens dynamiques, nécessitent de connaître les probabilités marginales, c'est-à-dire les probabilités de chaque variable isolée de l'instant suivant (cf. figure 1). Ceci présente deux inconvénients majeurs indépendants : d'abord, ce modèle introduit des cycles potentiels dans les variables de l'instant $t+1$ qu'il est nécessaire d'analyser afin de calculer correctement la probabilité de l'état suivant [Boutilier et al., 1998] ; ensuite, une traduction depuis le langage PPDDL, où les probabilités à l'instant $t + 1$ portent sur plusieurs variables à la fois, requiert de rajouter des variables d'état artificielles pour modéliser la dépendance entre les variables à l'instant $t + 1$ [Younes et al., 2005]. Troisièmement, une traduction depuis le langage PPDDL peut créer également de nombreuses variables d'état inutiles supplémentaires, dues à l'instanciation a priori de tous les prédicats du domaine pour tous les objets du problème, qu'ils soient atteignables ou non en pratique. Par exemple, dans le « monde des blocs », le prédicat `on(b1,b1)`, qui signifie que le bloc `b1` est empilé sur lui-même, ne devient jamais vrai mais il est pourtant construit et rajouté au réseau bayésien dynamique.

Par ailleurs, d'autres algorithmes de résolution des PDM intensionnels basés sur les réseaux bayésiens dynamiques définissent la politique comme une fonction paramétrée, qui représente la probabilité de choisir une action en fonction des valeurs des variables d'état. Ces algorithmes reposent sur la simulation de Monte-Carlo de la politique courante en partant d'un état initial connu, et apprennent les paramètres de la politique en fonction des observations reçues à l'aide de méthodes de descente de gradient. L'algorithme FPG [Buffet and Aberdeen, 2009], qui a remporté la compétition internationale de planification probabiliste en 2006, se base sur des réseaux de neurones pour représenter la politique et apprendre ses paramètres. Une version améliorée [Buffet and Aberdeen, 2007] utilise le planificateur déterministe FF comme une heuristique pour guider les trajectoires des simulations de Monte-Carlo vers les buts du problème à résoudre. Cette idée est au cœur des approches déterministes, que nous présentons dans le paragraphe suivant.

Enfin, récemment, certaines approches proposent de générer à la volée une partie du graphe

des états atteignables, à partir d'une description intensionnelle d'un PDM. Bien que certaines méthodes reposent sur des techniques de simulation stochastique [Keller and Eyerich, 2012], la majorité de ces approches est basée sur des algorithmes de recherche heuristique [Bonet and Geffner, 2005, Teichteil-Königsbuch et al., 2011, Teichteil-Königsbuch, 2012, Kolobov et al., 2012]. L'heuristique, qui est calculée en analysant la structure des opérateurs du langage de modélisation (PPDDL ou RDDL), permet de guider l'expansion du graphe vers les états les plus prometteurs.

Approches déterministes

Cette deuxième catégorie de méthodes comprend les approches de type « diviser pour régner » en divisant le problème probabiliste en plusieurs problèmes déterministes. Chaque action du PDM est transformée soit en une action déterministe dont l'effet est l'effet probabiliste le plus probable, soit en autant d'actions déterministes que d'effets probabilistes. Ces méthodes se sont développées depuis la première compétition internationale de planification probabiliste en 2004, car l'utilisation du langage PPDDL, basé sur PDDL, permet une interface très simple avec des planificateurs déterministes existants. Plus précisément, un domaine (probabiliste) PPDDL est transformé en plusieurs domaines (déterministes) PDDL, qui sont résolus par plusieurs appels à un même planificateur déterministe. Toutefois, les algorithmes diffèrent par la façon d'exécuter ou d'agréger les plans produits dans un contexte probabiliste.

Le premier algorithme de ce type, qui a remporté la première compétition internationale de planification probabiliste, est FF-REPLAN [Yoon et al., 2007]. Cette approche est purement réactive : après avoir généré un domaine (déterministe) PDDL, l'algorithme appelle le planificateur déterministe FF [Hoffmann and Nebel, 2001] depuis l'état initial et exécute le plan solution ; si le plan échoue à cause des effets probabilistes de l'action courante, l'algorithme relance FF depuis l'état d'échec courant et exécute le nouveau plan depuis cet état. Cet algorithme, pourtant aveugle quant aux récompenses et aux états d'échec – dont les états puits ou « culs-de-sac » – a obtenu de très bons résultats sur l'ensemble des problèmes, tant en qualité qu'en temps de calcul. En fait, les problèmes de cette première compétition probabiliste avaient été simplement adaptés de la compétition déterministe, sans que les effets probabilistes n'influent sensiblement sur l'optimisation de la solution du PDM : n'importe quel plan relativement court avait une chance élevée d'atteindre le but du problème, quitte à replanifier souvent en cas d'échec. Afin de remédier à ce problème, des problèmes avec des états puits atteignables avec une probabilité élevée ont été rajoutés dans les compétitions suivantes.

Plutôt que d'appeler le planificateur déterministe à chaque échec du plan indépendamment des risques futurs, le planificateur RFF [Teichteil-Königsbuch et al., 2010] agrège au sein d'une politique des plans produits par FF depuis plusieurs états qui peuvent être atteints en partant de l'état initial. Un faisceau de plans convergeant vers l'état but est ainsi construit de manière incrémentale jusqu'à ce que la probabilité d'échec à l'exécution des plans agrégés soit inférieure à un seuil donné. Cette probabilité d'échec est calculée par simulation de Monte-Carlo en partant de l'état initial et en suivant la politique agrégée courante. À chaque fois que la simulation de la politique courante atteint avec une probabilité suffisante des états pour lesquels aucune action n'est planifiée, le planificateur FF est appelé et le plan produit depuis cet état est rajouté dans la politique courante. Dans certaines configurations de l'algorithme RFF, la politique agrégée courante peut être optimisée de sorte à maximiser la probabilité d'atteindre le but en évitant les états puits. RFF a remporté la compétition internationale de planification

probabiliste en 2008, en ayant de bonnes performances tant en qualité qu'en temps de calcul pour la majorité des problèmes, y compris ceux comprenant des états puits atteignables avec une forte probabilité.

Enfin, d'autres algorithmes entremêlent une résolution classique de l'équation de Bellman avec une génération de plans déterministes à l'aide de FF. Ces approches ont l'avantage de synthétiser une politique optimale, car le planificateur déterministe est utilisé en quelque sorte comme heuristique de guidage vers la politique optimale. Le planificateur FF-HINDSIGHT [Yoon et al., 2008, 2010] optimise la fonction de valeur par améliorations successives de la politique, qui est une agrégation de plans déterministes produits par FF, et qui est évaluée par simulation de Monte-Carlo en partant de l'état initial. Cet algorithme peut être considéré comme l'intégration de RFF dans un schéma d'itération de la politique. La planificateur RETRASE [Kolobov et al., 2009, 2010a,b] approche la fonction de valeur par une combinaison linéaire de fonctions de base, qui sont automatiquement générées et paramétrées en utilisant des évaluations fournies par le planificateur déterministe FF. Ces deux algorithmes ont démontré de très bonnes performances en qualité sur les problèmes des dernières compétitions internationales de planification probabiliste.

9.4 Autres extensions du cadre des PDM en intelligence artificielle

En dehors de l'hypothèse de représentation des états et décisions *en extension*, d'autres hypothèses pesant sur le cadre des processus décisionnels de Markov ont dû être levées pour le rendre utilisable en planification dans l'incertain :

— L'hypothèse d'observabilité complète de l'état du monde à chaque instant.
— L'hypothèse (différente de la précédente) de connaissance parfaite du modèle (transitions, récompenses). En effet, parfois ce modèle n'est accessible qu'indirectement, par simulation ou expérimentation.
— Parfois également, seules des évaluations « qualitatives » des préférences et des connaissances sont disponibles.

Pour pallier ces différentes limitations, plusieurs pistes ont été suivies, que nous allons décrire succinctement. Pour plus de détails, le lecteur est invité à se référer aux deux ouvrages de synthèse [Buffet and Sigaud, 2008a,b]

9.4.1 Processus décisionnels de Markov partiellement observables

Dans de nombreux problèmes de décision séquentielle dans l'incertain, l'agent n'a pas une connaissance complète de l'état réel de son environnement, contrairement à ce qui est supposé dans le cadre des PDM. Ce sont ses actions qui, en plus de modifier l'état du système, lui apportent des informations permettant de préciser sa connaissance du monde. Ainsi, on peut distinguer deux types d'effets des actions, en environnement partiellement observable :

— Effet *physique* des actions : une action a_t, appliquée par l'agent à l'instant t, transforme l'état du monde x_t en un état x_{t+1} éventuellement de manière stochastique, suivant une probabilité de transition $p_t(x_{t+1}|x_t, a_t)$.

— Effet *épistémique* des actions (voir aussi volume 1, chapitre 11,« Révision des croyances et fusion d'informations multi-sources ») : l'état résultant x_{t+1} n'est pas observé directement, mais une *observation indirecte* o, élément d'un ensemble d'observations $\mathcal{O}$ est retournée par le système. Cette observation peut être « déterministe », dans le cas où il existe une fonction $O : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{O}$ telle que $o = O(x_{t+1}, a_t)$. Cette observation peut, plus généralement, être stochastique. Dans ce cas, $O(x_{t+1}, a_t, o) = p(o|x_{t+1}, a_t)$ est la probabilité d'observer o lorsque l'action a_t est appliquée et résulte en x_{t+1}.

La prise en compte de ces deux types d'effet conduit à étendre le cadre des processus décisionnels de Markov [Akström, 1965, Sondik, 1978]. La connaissance imparfaite de l'état du monde à l'instant t est modélisée par un *état de croyance* $b_t \in \mathcal{B}$, où $\mathcal{B}$ est l'ensemble des distributions de probabilité sur $\mathcal{X}$. b_0 est l'état de croyance initial et pour $t \geq 0$, $b_t(x)$ représente la probabilité associée par l'état de croyance à l'état x. Cet état de croyance évolue sous l'effet combiné des actions et observations successives.

Une première approche pour la résolution d'un PDMPO consiste à associer directement à chaque observation possible de l'état du monde une action. Mais cette méthode associe la même action à tous les états du monde conduisant à la même observation. Or, en général, de telles politiques sont sous-optimales [Littman, 1994]. Afin de déterminer une politique tirant au mieux parti des observations imparfaites, il est nécessaire de différencier des états conduisant à la même observation en prenant en compte les états rencontrés précédemment. Ceci peut être fait (dans le cas où l'horizon est fini) en définissant une politique δ_t déterminant l'action à effectuer à l'instant t, non pas simplement en fonction de la dernière observation o_{t-1}, mais en fonction de *l'historique* des actions et observations, $\{a_1, o_1, \ldots, a_{t-1}, o_{t-1}\}$.

Dans le cas où l'horizon est infini, bien qu'un PDMPO puisse être traduit en un PDM classique, la nature du problème obtenu (espace d'états continu) rend impossible sa résolution par les méthodes classiques de programmation dynamique. Cette complexité a été à la source de plusieurs voies de recherche en intelligence artificielle, et plusieurs familles d'algorithmes de résolution de PDMPO ont été proposées :

— Une première famille de méthodes met à profit la structure (linéaire par morceaux et convexe) de la fonction de valeur $V : \mathcal{B} \rightarrow \mathbb{R}$ pour proposer des algorithmes de type *itération de la valeur*. Ces algorithmes utilisent une structure d'arbre pour représenter une politique et sa fonction de valeur associée et définir une méthode itérative de calcul de la fonction de valeur (approchée) [Kaelbling et al., 1998].

— Les méthodes de résolution les plus récentes sont basées sur une discrétisation de l'espace d'états, ou combinent les deux types d'approches [Pineau et al., 2003], [Smith and Simmons, 2005].

9.4.2 Processus décisionnels de Markov et apprentissage

Le cadre des processus décisionnels de Markov permet de représenter et résoudre des problèmes de planification dans l'incertain. Grâce à la programmation dynamique, le surcroît de complexité de la résolution de ces problèmes lié à leur aspect séquentiel est limité. Nous venons d'indiquer qu'il est possible d'étendre le cadre des PDM à la planification en environnement « partiellement observable ». On parle parfois dans le cadre des PDM d'observabilité partielle dans un sens différent alors que l'état du monde est parfaitement connu à chaque instant. Il s'agit du cas où le *modèle* du PDM est imparfaitement connu, i.e. lorsque les fonctions p et r du modèle $< \mathcal{X}, A, p, r >$ sont inconnues a priori mais accessibles par expérimentation, soit

parce qu'on peut *simuler* la dynamique du système, soit parce qu'on peut l'expérimenter en temps réel. Les méthodes de type *apprentissage par renforcement* (voir par exemple volume 1, chapitre 9, « modèles d'apprentissage », ou volume 3, chapitre 8, « IA et robotique ») visent à résoudre de tels problèmes dans lesquels le « modèle » du PDM est appris en même temps que sa solution optimale. Pour ce faire, il existe deux types de méthodes : les méthodes *indirectes* et les méthodes *directes*.

Les *méthodes indirectes* [Kumar and Varaiya, 1986, Sutton, 1991, Peng and Williams, 1993, Moore and Atkeson, 1993] supposent d'apprendre dans un premier temps (par simulation ou expérimentation) le modèle $(p,\ r)$ du PDM, puis de le résoudre par un algorithme de Programmation Dynamique. De manière un peu plus évoluée, on peut focaliser l'effort lié à l'apprentissage du modèle sur des zones de l'espace d'états-actions ($\mathcal{X} \times A$) prometteuses, tout en ne négligeant pas totalement le reste de l'espace d'états-actions, afin de garantir qu'on ne passe pas à côté d'une politique optimale. Les méthodes indirectes permettent de résoudre des PDM dont on ne connaît pas le modèle, à la condition de pouvoir expérimenter ou simuler ce modèle. Ces méthodes présentent toutefois un inconvénient : elles nécessitent de stocker au moins partiellement les fonctions $\hat{p}$ et $\hat{r}$, ce qui peut nécessiter jusqu'à $O(|\mathcal{X}|^2|A|)$ d'espace de stockage.

Les *méthodes directes* [Sutton, 1988, Watkins, 1989, Watkins and Dayan, 1992] permettent de se passer de stocker le modèle $(p,\ r)$ en entier et de ne garder que ce qui est nécessaire à l'évaluation de politiques ou au calcul de politiques optimales. En contrepartie, des expérimentations / simulations plus nombreuses peuvent être nécessaires. Le choix d'une méthode directe sera donc préféré lorsque les simulations ont un coût faible, et qu'un problème de taille mémoire peut se poser.

Les méthodes directes et indirectes entrelacent en général apprentissage et programmation dynamique, afin de gagner en efficacité.

9.4.3 Approches qualitatives pour les PDM

Un apport de l'IA à la théorie de la décision en général a été la proposition et l'étude de critères de décision alternatifs au critère traditionnel de l'utilité espérée. Parmi ces critères de décision alternatifs, des critères « qualitatifs », plus adaptés à certaines problématiques de l'IA (élicitation de connaissances / préférences, communication homme / machine), ont été utilisés dans le cadre de la planification dans l'incertain.

[da Costa Pereira et al., 1997] ont proposé une approche de la planification dans l'incertain basée sur la théorie des possibilités. Celle-ci utilise des distributions de possibilité pour représenter les effets des actions, mais se place dans un cadre *non observable* et avec des préférences non graduelles sur des états buts. En conséquence, des plans *inconditionnels* maximisant le degré de possibilité ou de nécessité d'atteindre un état but sont calculés. Les problèmes sont modélisés dans un langage logique de type STRIPS. Par la suite, la théorie des possibilités a été utilisée afin de définir une contrepartie qualitative des processus décisionnels de Markov [Sabbadin et al., 1998, Sabbadin, 2001]. Récemment, ces travaux ont été étendus pour offrir un langage structuré et des algorithmes de résolution adaptés pour la planification dans l'incertain [Garcia and Sabbadin, 2008]. D'autres extensions de ce cadre sont présentées dans [Mouaddib et al., 2008].

9.5 Conclusion

Dans ce chapitre nous avons proposé une revue, forcément sommaire et non exhaustive, des travaux de la communauté de l'IA autour de la planification classique et de la planification dans l'incertain. Nous avons tout d'abord présenté le cadre de la planification classique STRIPS propositionnelle, puis présenté ses extensions basées sur le langage de description de problèmes PDDL devenu un standard dans la communauté. Nous avons traité brièvement la thématique de l'analyse structurelle de problèmes, à l'origine du développement de planificateurs performants, et décrit les principaux algorithmes de planification classique et planificateurs associés. Nous avons ensuite mis l'accent sur le cadre des processus décisionnels markoviens (PDM) que la communauté de l'intelligence artificielle s'est approprié afin de l'utiliser en planification sous incertitude. A cet effet des algorithmes novateurs de résolution approchée ou non de PDM ont été proposés. De plus, certains des points forts de l'IA en termes de représentation des connaissances (logique, réseaux bayésiens) ont été mis à profit afin d'améliorer le faible pouvoir d'expression du modèle PDM traditionnel.

Nous avons tenté dans ce chapitre de présenter brièvement les aspects saillants de ces diverses approches et proposé des pointeurs vers des articles les traitant de manière plus approfondie. Les lecteurs intéressés par la planification classique, les processus décisionnels de markov et leurs utilisations en IA sont invités à se pencher sur l'abondante source bibliographique (souvent anglo-saxonne) existant sur le sujet.

Nous avons dû faire des choix sur les approches décrites dans ce chapitre. Nous avons en particulier laissé de côté un pan entier, très actif, de la recherche en planification en intelligence artificielle, celui de la *planification multi-acteurs*. Le lecteur intéressé par ce domaine pourra par exemple s'orienter vers des références générales sur la planification distribuée [Durfee, 1999] ou sur la planification multi-agents [Vlassis, 2009, Shoham and Leyton-Brown, 2009] et ses liens avec les MDP [Beynier et al., 2010, Canu and Mouaddib, 2011].

Références

K. J. Akström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10 :174–205, 1965.

F. Bacchus. The 2000 AI planning systems competition. *AI Magazine*, 22(3) :47–56, 2001.

C. Bäckström and B. Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4) :625–655, 1995.

R. Iris Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10 (2-3) :171–206, 1997.

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

D. P. Bertsekas. *Dynamic Programming : Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1987.

A. Beynier, F. Charpillet, D. Szer, and A.I. Mouaddib. *Markov Decision Processes and Artificial Intelligence*, chapter DEC-MDP/POMDP, pages 321–359. Wiley, 2010.

J. Bibai, P. Savéant, M. Schoenauer, and V. Vidal. An evolutionary metaheuristic based on state

decomposition for domain-independent satisficing planning. In *Proc. ICAPS*, pages 18–25, 2010.

A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2) :279–298, 1997.

J. Blythe. An overview of planning under uncertainty. *AI magazine*, 20(2) :37–54, 1999.

B. Bonet and H. Geffner. Planning as heuristic search : New results. In *Proc. ECP*, pages 359–371, 1999.

B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2) :5–33, 2001.

B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proc. AAAI*, pages 714–719, 1997.

Blai Bonet and Héctor Geffner. mGPT : A probabilistic planner based on heuristic search. *JAIR*, 24 :933–944, 2005.

A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-FF : Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24 : 581–621, 2005.

C. Boutilier, R. I. Brafman, and C. W. Geib. Structured reachability analysis for Markov decision processes. In *Proc. UAI*, pages 24–32, 1998.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2) :49–107, 2000.

R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, 1986.

O. Buffet and D. Aberdeen. FF + FPG : Guiding a policy-gradient planner. In *Proc. ICAPS*, pages 42–48, 2007.

O. Buffet and D. Aberdeen. The factored policy-gradient planner. *Artificial Intelligence*, 173 (5-6) :722–747, 2009.

O. Buffet and O. Sigaud, editors. *Processus décisionnels de Markov en intelligence artificielle - vol. 1*. Traité IC2 - Informatique et systèmes d'information. Hermes - Lavoisier, Cachan, France, 2008a.

O. Buffet and O. Sigaud, editors. *Processus décisionnels de Markov en intelligence artificielle - vol. 2*. Traité IC2 - Informatique et systèmes d'information. Hermes - Lavoisier, Cachan, France, 2008b.

E. Burns, S. Lemons, R. Zhou, and W. Ruml. Best-first heuristic search for multi-core machines. In *Proc. IJCAI*, pages 449–455, 2009.

T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2) :165–204, 1994.

D. Cai, J. Hoffmann, and M. Helmert. Enhancing the context-enhanced additive heuristic with precedence constraints. In *Proc. ICAPS*, 2009.

A. Canu and A.I. Mouaddib. Dynamic local interaction model : framework and algorithms. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11), Workshop of Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domains (MSDM)*, 2011.

Y. Chen, C. Hsu, and B. Wah. Temporal planning using subgoal partitioning and resolution in SGPlan. *Artificial Intelligence*, 26 :323–369, 2006.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. ASPEN - Automating space mission operations using automated planning and scheduling. In *Proceedings of the International Conference on Space Operations (SpaceOps)*, 2000.

A. Coles and K. A. Smith. Marvin : A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*, 28 :119–156, 2007.

A. Coles, M. Fox, D. Long, and A. Smith. Planning with problems requiring temporal coordination. In *Proc. AAAI*, pages 892–897, 2008.

A. Coles, A. Coles, M. Fox, and D. Long. Temporal planning in domains with linear processes. In *Proc. IJCAI*, pages 1671–1676, 2009.

W. Cushing, S. Kambhampati, Mausam, and D. S. Weld. When is temporal planning really temporal ? In *Proc. IJCAI*, pages 1852–1859, 2007a.

W. Cushing, D. S. Weld, S. Kambhampati, Mausam, and K. Talamadupula. Evaluating temporal planning domains. In *Proc. ICAPS*, pages 105–112, 2007b.

C. da Costa Pereira, F. Garcia, J. Lang, and R. Martin-Clouaire. Planning with graded non deterministic actions : A possibilistic approach. *International Journal of Intelligent Systems*, 12 :935–962, 1997.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3) :142–150, 1990.

M. B. Do and S. Kambhampati. Planning as constraint satisfaction : Solving the planning graph by compiling it into CSP. *Artificial Intelligence*, 132(2), 2001.

M.B. Do and S. Kambhampati. Sapa : A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, 20 :155–194, 2003.

E. H. Durfee. Distributed problem solving and planning. In *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, 1999.

S. Edelkamp and P. Kissmann. GAMER : Bridging planning and general game playing with symbolic search. In *Booklet of the 6th International Planning Competition*, 2008.

E. Fabre, L. Jezequel, P. Haslum, and S. Thiébaux. Cost-optimal factored planning : Promises and pitfalls. In *Proc. ICAPS*, pages 65–72, 2010.

Z. Feng and E. A. Hansen. Symbolic heuristic search for factored Markov decision processes. In *Proc. AAAI*, pages 455–460, 2002.

Z. Feng, E. A. Hansen, and S. Zilberstein. Symbolic generalization for on-line planning. In *Proc. UAI*, pages 209–216, 2003.

R. Fikes and N. J. Nilsson. STRIPS : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4) :189–208, 1971.

M. Fox and D. Long. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20 :61–124, 2003.

M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27 :235–297, 2006.

L. Garcia and R. Sabbadin. Complexity results and algorithms for possibilistic influence dia-

grams. *Artificial Intelligence*, 172(8-9) :1018–1044, 2008.

B. Gazen and C. Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In *Proc. ECP*, pages 221–233, 1997.

H. Geffner. Functional STRIPS : A more flexible language for planning and problem solving. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 187–209. Kluwer, Alphen aan den Rijn, Netherlands, 2000.

A. Gerevini and D. Long. Plan constraints and preferences in PDDL3. Technical Report RT 2005-08-47, Dept. of Electronics for Automation, University of Brescia, Italy, 2005.

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

M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In *Proc. AIPS*, pages 61–67, 1994.

M. Ghallab, D. Nau, and P. Traverso. *Automated planning : Theory and practice*. Morgan Kaufmann, San Francisco, CA, USA, 2004.

R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2) :163–223, 2003.

S. Grandcolas and C. Pain-Barre. Filtering, decomposition and search space reduction for optimal sequential planning. In *Proc. AAAI*, pages 993–998, 2007.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19 :399–468, 2003.

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

P. Haslum. Improving heuristics through relaxed search - an analysis of TP4 and HSP*a in the 2004 planning competition. *Journal of Artificial Intelligence Research*, 25 :233–267, 2006.

P. Haslum and H. Geffner. Admissible heuristics for optimal planning. In *Proc. AIPS*, pages 70–82, 2000.

P. Haslum and H. Geffner. Heuristic planning with time and resources. In *Proc. ECP*, pages 121–132, 2001.

M. Helmert. Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, 143(2) :219–262, 2003.

M. Helmert. A planning heuristic based on causal graph analysis. In *Proc. ICAPS*, pages 161–170, 2004.

M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26 :191–246, 2006.

M. Helmert. *Understanding Planning Tasks*. Springer Verlag, Berlin, Germany, 2008.

M. Helmert and C. Domshlak. Landmarks, critical paths and abstractions : What's the difference anyway ? In *Proc. ICAPS*, 2009.

M. Helmert and H. Geffner. Unifying the causal graph and additive heuristics. In *Proc. ICAPS*, pages 140–147, 2008.

M. Helmert, P. Haslum, and J. Hoffmann. Flexible abstraction heuristics for optimal sequential

planning. In *Proc. ICAPS*, pages 176–183, 2007.

M. Helmert, M. B. Do, and I. Refanidis. IPC-2008, deterministic part : Changes in PDDL 3.1. http ://ipc08.icaps-conference.org/PddlExtension, 2008a.

M. Helmert, M. B. Do, and I. Refanidis. IPC-2008, deterministic part : Results. http ://ipc08.icaps-conference.org/Results, 2008b.

S. L. Hickmott, J. Rintanen, S. Thiébaux, and L. B. White. Planning via petri net unfolding. In *Proc. IJCAI*, pages 1904–1911, 2007.

J. Hoey, R. St-Aubin, A. J. Hu, and C. Boutilier. SPUDD : Stochastic planning using decision diagrams. In *Proc. UAI*, pages 279–288, 1999.

J. Hoffmann. Extending FF to numerical state variables. In *Proc. ECAI*, pages 571–575, 2002.

J. Hoffmann and S. Edelkamp. PDDL2.2 : The language for the classical part of IPC-4. In *Booklet of the 4th International Planning Competition*, 2004.

J. Hoffmann and S. Edelkamp. The deterministic part of IPC-4 : An overview. *Journal of Artificial Intelligence Research*, 24 :519–579, 2005.

J. Hoffmann and B. Nebel. The FF planning system : Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14 :253–302, 2001.

J. Hoffmann, J. Porteous, and L. Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22 :215–278, 2004.

R. Huang, Y. Chen, and W. Zhang. A novel transition based encoding scheme for planning as satisfiability. In *Proc. AAAI*, 2010.

F. V. Jensen. *Bayesian networks and decision graphs*. Springer Verlag, Berlin, Germany, 2001.

S. Joshi, K. Kersting, and R. Khardon. Self-taught decision theoretic planning with first order decision diagrams. In *Proc. ICAPS*, pages 89–96, 2010.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable domains. *Artificial Intelligence*, 101(1-2) :99–134, 1998.

E. Karpas and C. Domshlak. Cost-optimal planning with landmarks. In *Proc. IJCAI*, pages 1728–1733, 2009.

H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proc. IJCAI*, pages 318–325, 1999.

H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proc. KR*, pages 374–384, 1996.

Thomas Keller and Patrick Eyerich. Prost : Probabilistic planning based on uct. In *ICAPS*, 2012.

E. Keyder and H. Geffner. Heuristics for planning with action costs revisited. In *Proc. ECAI*, pages 588–592, 2008.

E. Keyder, S. Richter, and M. Helmert. Sound and complete landmarks for and/or graphs. In *Proc. ECAI*, pages 335–340, 2010.

A. Kishimoto, A. S. Fukunaga, and A. Botea. Scalable, parallel best-first search for optimal sequential planning. In *Proc. ICAPS*, pages 10–17, 2009.

J. Koehler and J. Hoffmann. On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm. *Journal of Artificial Intelligence Research*, 12 :338–386, 2000.

J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In *Proc. ECP*, pages 273–285, 1997.

A. Kolobov, Mausam, and D. S. Weld. ReTrASE : Integrating paradigms for approximate probabilistic planning. In *Proc. IJCAI*, pages 1746–1753, 2009.

A. Kolobov, Mausam, and D. S. Weld. Classical planning in MDP heuristics : With a little help from generalization. In *Proc. ICAPS*, pages 97–104, 2010a.

A. Kolobov, Mausam, and D. S. Weld. SixthSense : Fast and reliable recognition of dead ends in MDPs. In *Proc. AAAI*, pages 1108–1114, 2010b.

Andrey Kolobov, Peng Dai, Mausam, and Daniel S. Weld. Reverse iterative deepening for finite-horizon MDPs with large branching factors. In *ICAPS*, 2012.

R. Korf. Depth-first iterative-deepening : an optimal admissible tree search. *Artificial Intelligence*, 27(1) :97–109, 1985.

P. R. Kumar and P. P. Varaiya. *Stochastic Systems : Estimation, Identification and Adaptive Control.* Prentice Hall, Englewood Cliffs, NJ, USA, 1986.

U. Kuter and D. S. Nau. Using domain-configurable search control for probabilistic planning. In *Proc. AAAI*, pages 1169–1174, 2005.

P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling. *Artificial Intelligence*, 143(2) :151–188, 2003.

P. Laborie and M. Ghallab. Planning with sharable resources constraints. In *Proc. IJCAI*, pages 1643–1649, 1995.

M. L. Littman. Memoryless policies : Theoretical limitations and practical results. In *Proc. of the 3rd International Conference on Simulation and Adaptive Behavior*, pages 238–245, 1994.

D. Long and M. Fox. The efficient implementation of the plan-graph. *Journal of Artificial Intelligence Research*, 10 :85–115, 1999.

D. Long and M. Fox. The 3rd international planning competition : Results and analysis. *Journal of Artificial Intelligence Research*, 20 :1–59, 2003.

F. Maris and P. Régnier. TLP-GP : Solving temporally-expressive planning problems. In *Proc. TIME*, pages 137–144, 2008.

F. Maris, P. Régnier, and V. Vidal. Planification par satisfaction de bases de clauses. In Lakhdar Saïs, editor, *Problème SAT : défis et challenges*, chapter 11, pages 289–309. Hermes, 2008.

D. McDermott. A heuristic estimator for means-ends analysis in planning. In *Proc. AIPS*, pages 142–149, 1996.

D. McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2) :35–56, 2000.

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CI, USA, 1998.

P. Meseguer, F. Rossi, and T. Schiex. Soft constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 9, pages 281–328. Elsevier, Amsterdam, Netherlands, 2006.

A. W. Moore and C. G. Atkeson. Prioritized sweeping : Reinforcement learning with less data

and less real time. *Machine Learning*, 13 :103–130, 1993.

A. I. Mouaddib, C. Pralet, R. Sabbadin, and P. Weng. Processus décisionnels de Markov et critères non classiques. In O. Buffet and O. Sigaud, editors, *Processus décisionnels de Markov en intelligence artificielle - vol 1*, chapter 5. Hermes - Lavoisier, Cachan, France, 2008.

A. Newell and H. Simon. GPS : A program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. McGraw Hill, New-York, NY, USA, 1963.

X. Nguyen and S. Kambhampati. Reviving partial order planning. In *Proc. IJCAI*, pages 459–466, 2001.

J. Pearl. *Heuristics*. Addison Wesley, Reading, MA, USA, 1983.

E. P. D. Pednault. ADL : Exploring the middle ground between STRIPS and the situation calculus. In *Proc. IJCAI*, pages 324–332, 1989.

J. Penberthy and D. Weld. UCPOP : A sound, complete, partial order planner for ADL. In *Proc. KR*, pages 103–114, 1992.

J. Peng and R. J. Williams. Efficient learning and planning within the dyna framework. *Adaptive Behavior*, 1(4) :437–454, 1993.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration : An anytime algorithm for POMDPs. In *Proc. IJCAI*, pages 1025–1032, 2003.

Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3) : 193–204, 1970.

C. Pralet and G. Verfaillie. Réseaux de contraintes sur des chronogrammes pour la planification et l'ordonnancement. *Revue d'Intelligence Artificielle*, 24(4) :485–504, 2010.

M. L. Puterman. *Markov decision processes : Discrete stochastic dynamic programming*. John Wiley & Sons, New York, NY, USA, 1994.

S. Richter and M. Westphal. The LAMA planner : Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39 :127–177, 2010.

S. Richter, M. Helmert, and M. Westphal. Landmarks revisited. In *Proc. AAAI*, pages 975–982, 2008.

J. Rintanen. Complexity of concurrent temporal planning. In *Proc. ICAPS*, pages 280–287, 2007.

J. Rintanen. Heuristics for planning with SAT. In *Proc. CP*, pages 414–428, 2010.

R. Sabbadin. Possibilistic Markov decision processes. *Engineering Applications of Artificial Intelligence*, 14 :287–300, 2001.

R. Sabbadin, H. Fargier, and J. Lang. Towards qualitative approaches to multi-stage decision making. *International Journal of Approximate Reasoning*, 19(3-4) :441–471, 1998.

S. Sanner. Relational Dynamic Influence Diagram Language (RDDL) : Language Description. `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`, 2010.

Y. Shoham and K. Leyton-Brown. *Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York, 2009.

D. E. Smith and D. S. Weld. Temporal planning with mutual exclusion reasoning. In *Proc. IJCAI*, pages 326–337, 1999.

T. Smith and R. Simmons. Point-based POMDP algorithms : Improved analysis and implementation. In *Proc. UAI*, pages 542–547, 2005.

E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon : Discounted costs. *Operations Research*, 26(2) :282–304, 1978.

R. St-Aubin, J. Hoey, and C. Boutilier. APRICODD : Approximate policy construction using decision diagrams. In *Proc. NIPS*, pages 1089–1095, 2000.

R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3 (1) :9–44, 1988.

R. Sutton. Planning by incremental dynamic programming. In *Proc. of the 8th International Workshop on Machine Learning*, pages 353–357, 1991.

F. Teichteil-Königsbuch, U. Kuter, and G. Infantes. Incremental plan aggregation for generating policies in MDPs. In *Proc. AAMAS*, pages 1231–1238, 2010.

Florent Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *AAAI*, 2012.

Florent Teichteil-Königsbuch, Vincent Vidal, and Guillaume Infantes. Extending classical planning heuristics to probabilistic planning with dead-ends. In *Proc. AAAI'11*, 2011.

S. Thiébaux, J. Hoffmann, and B. Nebel. In defense of PDDL axioms. In *Proc. IJCAI*, pages 961–968, 2003.

R. Valenzano, N. Sturtevant, J. Schaeffer, K. Buro, and A. Kishimoto. Simultaneously searching with multiple settings : An alternative to parameter tuning for suboptimal single-agent search algorithms. In *Proc. ICAPS*, pages 177–184, 2010.

G. Verfaillie, C. Pralet, and M. Lemaître. How to model planning and scheduling problems using constraint networks on timelines. *Knowledge Eng. Review*, 25(3) :319–336, 2010.

V. Vidal. *Recherche dans les graphes de planification, satisfiabilité et stratégies de moindre engagement. Les systèmes LCGP et LCDPP*. PhD thesis, IRIT, Université Paul Sabatier, Toulouse, France, 2001.

V. Vidal. A lookahead strategy for heuristic search planning. In *Proc. ICAPS*, pages 150–160, 2004.

V. Vidal and H Geffner. Solving simple planning problems with more inference and no search. In *Proc. CP*, pages 682–696, 2005.

V. Vidal and H. Geffner. Branching and pruning : An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3) :298–335, 2006.

V. Vidal, L. Bordeaux, and Y. Hamadi. Adaptive K-parallel best-first search : A simple but efficient algorithm for multi-core domain-independent planning. In *Proc. 3rd Symposium on Combinatorial Search (SOCS)*, 2010.

N. Vlassis. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

C. Wang, S. Joshi, and R. Khardon. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research*, 31(1) :431–472, 2008.

C. J. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

C. J. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 3(8) :279–292, 1992.

D. S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4) :27–61, 1994.

S. W. Yoon, A. Fern, and R. Givan. FF-Replan : A baseline for probabilistic planning. In *Proc. ICAPS*, pages 352–359, 2007.

S. W. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *Proc. AAAI*, pages 1010–1016, 2008.

S. W. Yoon, W. Ruml, J. Benton, and M. B. Do. Improving determinization in hindsight for on-line probabilistic planning. In *Proc. ICAPS*, pages 209–217, 2010.

H. L. S. Younes and R. G. Simmons. VHPOP : Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, 20 :405–430, 2003.

H. L. S. Younes, M. L. Littman, D. Weissman, and J. Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24 : 851–887, 2005.