Planification à base d'ordres topologiques pour la résolution des POMDPs

Jilles S. Dibangoye[1,2], Brahim Chaib-draa[1], Abdel-Illah Mouaddib[2]

[1] Laboratoire Damas
Université Laval - Québec Q.C., Canada
jilles-steeve.dibangoye.1@ulaval.ca

[2] Laboratoire Greyc – UMR 6072,
Université de Caen, France

Résumé : Bien que les processus décisionnels de Markov partiellement observables (POMDPs) aient reçu beaucoup d'attention au cours des dernières années, à ce jour, la résolution des problèmes réels reste un sérieux défi. Dans ce contexte, les techniques d'accélération des méthodes fondamentales ont été un des principaux axes de recherche. Parmi elles, les algorithmes ordonnés suggèrent des solutions au problème de l'ordre des *mises à jour* de la fonction de valeur. Ces techniques permettent notamment de réduire considérablement le nombre de mises à jour requises, mais en contrepartie, elles impliquent des coûts supplémentaires. Dans ce papier, nous présentons une nouvelle approche ordonnée : *la planification à base d'ordres topologiques* (TOP). Cette approche exploite les relations de causalité entre états afin de faire face à deux problématiques : (1) la détection de la structure d'un POMDP comme moyen de surmonter à la fois la malédiction de la dimension et celle de l'historique ; (2) la réduction des mises à jour inutiles et le déterminisme d'une fonction de valeur approximative sur la base d'ordres topologiques induits par la structure sous-jacente du POMDP. Les expérimentations prouvent que TOP est compétitif comparativement aux meilleurs algorithmes actuels. **Mots-clés** : POMDPs, Prise de décision séquentielle, Ordre topologique.

1 Introduction

Les POMDPs offrent un cadre robuste pour la planification dans les domaines impliquant états cachés et incertitude sur les effets des actions. Toutefois, malgré un effort soutenu dans le développement d'algorithmes de plus en plus efficaces, *e.g.,* (Roy *et al.*, 2005; Poupart & Boutilier, 2004; Pineau *et al.*, 2003; Cassandra *et al.*, 1997), les aspects de complexité et de calculabilité limitent l'utilisation des POMDPs dans les problèmes réels. Les limitations de ces techniques sont, dans une large mesure, une conséquence de deux problèmes étroitement liés : *la malédiction de la dimension* (Bellman, 1957), où la dimension du problème de planification est déterminée par le nombre d'états ; et *la malédiction de l'historique* (Pineau *et al.*, 2003), où le nombre d'historiques croît de façon exponentielle avec l'horizon de planification. Bon nombre de stratégies ont été proposées afin de réduire le nombre d'historiques considérés pour l'obtention d'une fonction de valeur exacte (Cassandra *et al.*, 1997) ou approximative (Pineau *et al.*, 2003). Les approches exactes de déterminisme d'une fonction de valeur ε-optimale sur l'espace entier des croyances, espace continu, utilisent des techniques d'élagage afin de réduire l'ensemble des historiques. Néanmoins, en raison de la continuité de l'espace des croyances, ces approches se sont révélées inefficaces en pratique. Les méthodes parmi les plus prometteuses dans le déterminisme d'une fonction de valeur approximative, pour des domaines de très grande dimensions sont basées sur l'opérateur de *mise à jour à base de points*. En particulier, Pineau *et al.* (2003) ont abandonné la mise à jour de la fonction de valeur sur l'ensemble des points de croyance en faveur de celle d'un seul point à la fois. L'algorithme résultant nommé *itération de valeurs à base de points* (PBVI) utilise alors un opérateur de mise à jour polynomial en lieu et place d'un opérateur exponentiel. Toutefois, la majorité des méthodes à base de points, y compris PBVI, restent limitées par la malédiction de la dimension. Comme nous l'avons déjà mentionné, certains algorithmes sont incapables de passer à l'échelle car ils calculent leur fonction de valeur exacte ou approximative sur l'ensemble entier des états de croyance. Cependant, dans les applications réelles, le déterminisme d'une fonction de valeur sur l'espace entier des états de croyance est bien souvent inutile. Selon Roy *et al.* (2005), l'espace des états de croyance accessible par simulation est souvent associé à un sous-espace structuré, de petite dimension, inclut dans un espace de plus grande dimension. Par conséquent, la recherche et l'exploitation de la structure

des POMDPs peuvent améliorer les algorithmes fondamentaux. Malheureusement, alors que des approches structurées, telles que celles qui sont proposées par Roy *et al.* (2005); Poupart & Boutilier (2004), peuvent conduire à une réduction exponentielle de la dimension de certains POMDPs, leurs mécanismes algorithmiques sont très délicats. Bien qu'utiles et intéressantes, ces approches n'ont aucun lien direct avec la nôtre, pour cette raison nous ne poursuivrons pas la comparaison.

La raison d'être de notre approche est de faire face aussi bien la malédiction de la dimension quà la malédiction de l'historique, et cela dans un même mécanisme algorithmique. TOP généralise la relation de causalité entre les états, relation qui a été utilisée avec succès en MDPs (Dai & Goldsmith, 2007; Bonet & Geffner, 2003; Abbad & Boustique, 2003). Il atténue la malédiction de la dimension en cloisonnant les états en groupe d'états mutuellement dépendants, *i.e,* couches. Le graphe de transition sur ces couches constitue la structure du problème. Puis, il n'effectue de planification que sur l'ensemble des couches dites *pertinentes*. Il surmonte la malédiction de l'historique en se limitant à un petit nombre de couches à chaque *simulation* (de trajectoires). TOP traverse simultanément l'espace des états de croyance et l'espace des états du MDP sous-jacent, et cela suivant des ordres topologiques. Il n'effectue de mises à jour que si celles-ci sont susceptibles de changer la fonction de valeur de façon non-triviale. Cela permet d'éviter les problèmes de mises à jour inutiles. Enfin, TOP est capable d'identifier les couches où la fonction de valeur a convergé et donc de re-diriger la recherche vers des couches non *ou* pas suffisamment explorées.

Le reste de ce papier est organisé comme suit : dans la Section 2, nous couvrons tout d'abord le contexte général des POMDPs et passons en revue les travaux parmi les plus récents y compris les techniques ordonnées ; Ensuite, nous présentons notre nouvelle approche ordonnée en Section 3, et discutons de sa mise en œuvre en Section 4. Enfin, nous présentons une évaluation empirique de TOP, et démontrons ses forces et faiblesses par rapport à d'autres algorithmes bien connus, et cela sur un grand nombre de bancs d'essai.

2 Contexte général et travaux connexes

Nous commençons avec une vue d'ensemble des MDPs, POMDPs puis des espaces d'états de croyance. Nous offrons ensuite une courte introduction à l'état de l'art des algorithmes de résolution des POMDPs.

Un processus décisionnel de Markov (MDP) est un n-uplet (S, A, T, R, γ) où : S est un ensemble discret et fini d'états ; A est un ensemble discret et fini d'actions ; $T(s'|s, a)$ est une fonction de transition, *i.e.,* la probabilité de transition d'un état s à un état s' après l'exécution de l'action a ; $R(s, a)$ est une valeur réelle décrivant la récompense reçue après avoir exécuté l'action a dans l'état s ; γ est le facteur de décompte. Le cadre des MDPs a ensuite été généralisé aux processus décisionnels de Markov partiellement observables (POMDPs) pour la planification dans des domaines impliquant états cachés et incertitude sur les effets des actions. Plus formellement, un POMDP est un n-uplet $(S, A, T, R, \Omega, O, \gamma, b_0)$, où S, A, T, R et γ sont identiques au cadre des MDPs ; Ω est une ensemble discret et fini d'observations ; $O(o|s', a)$ est la probabilité d'observer o après exécution de l'action a et l'accès à l'état s' ; Un des aspects importants en POMDPs est la notion d'états de croyance noté b décrivant une distribution de probabilité sur l'ensemble des état du système, et constituant la statistique suffisante pour un historique donné ; b_0 désigne ainsi l'état de croyance initial, *i.e.,* la croyance de l'agent quant à son état initial dans le système. L'état de croyance suivant b, noté $b' = \tau(b, a, o)$, est mise à jour comme suit : $b'(s') = \frac{O(o|s',a)\sum_s b(s)T(s'|s,a)}{pr(o|b,a)}$, avec $pr(o|b, a) = \sum_s b(s) \sum_{s'} T(s'|s, a)O(o|s', a)$. étant donné un POMDP, l'objectif est de trouver une fonction de valeur V qui associe à tout état de croyance un hyperplan. En outre, comme la fonction de valeur V a été prouvée linéaire par morceaux et convexe (Sondik, 1978), elle peut être représentée comme un ensemble d'hyperplans $V = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ dit α-vecteurs, de sorte que $V(b) = \max_i (\alpha_i \cdot b)$. Enfin, la fonction de valeur V peut-être iterativement calculée en utilisant l'opérateur de mise à jour à base de points (Pineau *et al.*, 2003; Spaan & Vlassis, 2005; Smith & Simmons, 2004, 2005) :

$$\begin{array}{lcl} g^{\alpha}_{a,o}(s) & = & \sum_{s'} O(o|s', a)T(s'|s, a)\alpha_i(s') \\ g^{b}_{a} & = & r_a + \gamma \sum_o \arg\max_{g^{\alpha}_{a,o}:\, \alpha \in V} (b \cdot g^{\alpha}_{a,o}) \\ backup(b) & = & \arg\max_{g^{b}_{a}:\, a \in A} (b \cdot g^{b}_{a}) \end{array} \quad (1)$$

où $r_a(s) = R(s, a)$ est un vecteur de réprésentation de la fonction de récompense.

2.1 Algorithmes approximatifs

L'obstacle dû aux calculs résultant du nombre de vecteurs, qui sont générés pour le déterminisme d'une fonction de valeur optimale sur l'espace entier des états de croyance, a conduit au développement d'une vaste variété de méthodes approximatives. Au lieu de calculer une fonction de valeur optimale sur l'ensemble de l'espace des états de croyance, une approximation possible est de le faire uniquement sur un nombre fini d'états de croyance (Lovejoy, 1991). En conséquence l'opérateur de mise à jour est polynomial et la taille de la fonction de valeur est bornée par le nombre d'états de croyance. Néanmoins, une fonction de valeur optimale sur un ensemble discret d'états de croyance B est difficilement généralisable aux états de croyance hors de B. En revanche, les algorithmes de mise à jour à base de points mettent à jour uniquement les états de croyance qui sont *accessibles* en partant de b_0 (Pineau *et al.*, 2003; Spaan & Vlassis, 2005; Smith & Simmons, 2004, 2005). En mettant à jour ces états de croyance accessibles, il est empiriquement et théoriquement prouvé que la fonction de valeur se généralisera aux états de croyance hors de B.

En particulier, PBVI calcule alternativement les deux, un ensemble discret B d'états de croyance accessibles et la fonction de valeur optimale correspondante (Pineau *et al.*, 2003). L'algorithme débute par la construction de l'ensemble initial $B_0 = \{b_0\}$ des états de croyance, puis étend B_0 à B_1 avec l'ensemble des états de croyance les plus distants parmi les successeurs immédiats des états de croyance de B_0 et calcule une fois de plus la fonction de valeur optimale sur B_1. Ce processus se poursuit jusqu'à ce qu'un critère d'arrêt ait atteint. Un certain nombre d'inconvénients limitent cependant les capacités de l'algorithme PBVI à passer à l'échelle. Tout d'abord, l'étape d'expansion des états de croyance engage un grand nombre de ressources. En effet, cette étape consiste à comparer $|B||A||O|$ successeurs immédiats selon leur distance, *i.e.,* métrique L_2. Par conséquent, dans le pire des cas, cette sous-routine requiert $O(|B|^2|A||O||S|)$ opérations. Pour faire face à ce point faible, Pineau *et al.* (2003) ont proposé l'échantillonnage d'un successeur pour chaque paire, état de croyance – action, cela conduit à la réduction de la complexité de la phase d'expansion à $O(|B|^2|A||S|)$. Le second point de faiblesse réside dans la phase de mise à jour. Cela est principalement dû au fait que PBVI met à jour tous les états de croyance de B à chaque itération, même si cela est inutile ou redondant pour certains d'entre eux.

Perseus proposé par Spaan & Vlassis (2005), est un autre algorithme à base de points. Il est dans l'esprit assez proche de PBVI. À la différence que l'étape d'expansion de PBVI est remplacée par une marche aléatoire à partir de l'état de croyance initial b_0, et au lieu de mettre à jour tous les états de croyance B à chaque itération Perseus le fait seulement pour un sous-ensemble de B. Ce sous-ensemble correspond aux états de croyance dont les valeurs n'ont pas été améliorées à la dernière mise à jour. Comme Perseus met souvent à jour un sous-ensemble d'états de croyance, il peut soit converger plus rapidement ou utiliser un plus grand nombre d'états de croyance afin d'améliorer la qualité de sa fonction de valeur. Le principal problème de Perseus réside dans la lenteur de sa convergence dans certains domaines. Bien que la marche aléatoire dirige la recherche vers des espaces les plus probablement accessibles, il n'existe aucune garantie que ces domaines offriront les récompenses utiles pour l'amélioration de la fonction de valeur. En outre, en traversant l'espace des états de croyance suivant une marche aléatoire sont comportement est aléatoire tout comme ses performances dans certains des domaines que nous avons testés.

De cette famille d'algorithmes, *heuristic search value iteration* (HSVI) a montré les meilleures performances et cela sur les bancs d'essai parmi les plus grands (Smith & Simmons, 2004, 2005). HSVI procède comme suit : il crée une séquence d'états de croyance, *i.e.,* trajectoire d'états de croyance, en se fondant à la fois sur la borne supérieure $\bar{V}$ et la borne inférieure $\underline{V}$ de la fonction de valeur V ; En partant avec l'état de croyance initial b_0, le prochain état de croyance qu'il sélection est le successeur du précédant maximisant l'écart entre la borne supérieure et la borne inférieure. L'expansion des états de croyance prend fin quand un critère d'arrêt est atteint, s'ensuit la mise à jour de la fonction de valeur dans le sens inverse de la génération des états de croyance. Malheureusement, aussi bien le maintien de la borne supérieure et la sélection de l'état de croyance suivant engendrent des coûts additionnels non négligeables. Par conséquent, le gain résultant de la réduction du nombre de mises à jour ne se manifeste pas pleinement dans le temps d'exécution total. Tout au long de l'exécution de l'algorithme, l'écart entre la borne inférieure et la borne supérieure à l'état de croyance initial b_0 décroît jusqu'à atteindre une valeur inférieure ou égale à un nombre réel ε. L'algorithme HSVI s'arrête alors. Néanmoins d'un point de vue pratique la clôture de cet écart est quasiment impossible à moins que l'écart initial soit relativement petit, comme nous le constatons dans les bancs d'essai de type Rocksample.

2.2 Algorithmes à base de simulation de trajectoires

Afin de surmonter le problème de la mise à jour systématique de tous les états de croyance à chaque itération, les chercheurs ont tourné leur attention vers des techniques à base de simulations de trajectoires. Il s'agit de méthodes où l'agent simule des interactions avec l'environnement et crée des trajectoires d'états de croyance. Parmi d'autres méthodes de cette classe, l'algorithme *forward search value iteration* (FSVI), récemment proposé par Shani *et al.* (2007), a démontré de très bonnes performances. FSVI utilise la Q-valeur du MDP sous-jacent comme heuristique pour l'exploration de l'espace des états de croyance. Dans ce dessein, il maintient à la fois un état s du MDP sous-jacent et un état de croyance b, ce qui permet la sélection d'une action a fondé sur l'état sous-jacent s et la Q-valeur du MDP sous-jacent. FSVI procède par la suite à la sélection de l'état suivant s' et l'observation o, en les échantillonnant de $T(\cdot|s,a)$ et $O(\cdot|a,s')$, respectivement. L'avantage de FSVI réside dans le faible coût résultant du calcul de la Q-valeur du MDP sous-jacent et donc la sélection des actions. Toutefois, un des inconvénients majeurs de FSVI repose sur le fait qu'il n'est pas en mesure de reconnaître les trajectoires qui permettront d'améliorer l'information sur l'état dans lequel il se trouve. En effet, en faisant usage de la Q-valeur optimale du MDP sous-jacent, FSVI limite sa capacité à créer des trajectoires qui visiteront des états susceptibles de fournir des observations utiles. Encore plus important, il est incapable de réorienter la recherche vers les états qui sont probablement inaccessibles suivant la Q-valeur du MDP sous-jacent, et cela même si ces derniers peuvent fournir des récompenses utiles pour la qualité de la fonction de valeur. En conséquence, FSVI peut conduire à une fonction de valeur sous-optimale dans certains domaines. Néanmoins, dans les domaines qui ne nécessitent pas la collecte d'information sur le long terme et/ou une exploration plus large de l'espace de croyance, FSVI est très compétitif. Enfin, même une simple politique d'exploration très conservative, *e.g.,* ε-gourmand, peut compenser le manque d'exploration.

2.3 Algorithmes ordonnés

Les techniques ordonnées visent à résoudre le problème du déterminisme d'une fonction de valeur approximative à la fois rapidement et avec précision. L'idée d'organiser les mises à jour dans un bon ordre a déjà été examinée auparavant par Shani *et al.* (2006); Virin *et al.* (2007) et dans une certaine mesure par Shani *et al.* (2007); Smith & Simmons (2004, 2005).

Shani *et al.* (2006) ont introduit l'algorithme *prioritized value iteration* (PVI). Cet algorithme utilise un ensemble fixe d'états de croyance et met à jour ces états de croyance suivant l'erreur de Bellman, défini comme suit : $e(b) = \max_a [r_a \cdot b + \sum_o pr(o|b,a)V(\tau(b,a,o))] - V(b)$. PVI met à jour à chaque étape l'état de croyance $b \in B$ qui améliorera plus probablement la fonction de valeur. Afin d'y parvenir, Shani *et al.* (2006) ont suggéré, dans la version naïve de PVI, de choisir l'état de croyance dont l'erreur de Bellman est la plus grande. L'algorithme s'arrête lorsqu'il ne parvient pas à trouver un état de croyance dont l'erreur de Bellman soit supérieure à zéro. En mettant à jour les états de croyance suivant l'ordre décroissant de leur erreur de Bellman, il est probable que certains états de croyance soient plus souvent mise à jour que d'autres. Cela peut conduire à des gains significatifs en termes de performances. Toutefois, il est un point qui limite l'usage de PVI. En effet, le calcul de l'erreur de Bellman suite à chacune des mises à jour induit des efforts considérables. La complexité de la mise à jour de l'erreur de Bellman est non négligeable en comparaison à la complexité de l'opérateur de mise à jour à base de points : dans le pire des cas, ils requièrent $O(|B||A||O||S|)$ et $O(|S|^2|V||A||O|)$ respectivement. Pour surmonter ce problème, Shani *et al.* (2006) calculent l'erreur de Bellman uniquement pour un sous-ensemble $\tilde{B}$ des états de croyance échantillonnés, réduisant ainsi la complexité de mise à jour de l'erreur de Bellman à $O(|\tilde{B}||A||O||S|)$.

Dans le même ordre d'idées, Virin *et al.* (2007) ont proposé un algorithme ordonné, nommé *soft cluster based value iteration* (SCVI). Ce dernier est basé sur la méthode *generalized prioritized sweeping* introduite en MDPs par Wingate & Seppi (2005). Afin d'alléger la complexité relative au déterminisme d'une bonne séquence de mise à jour, de façon similaire à FSVI, ils utilisent un algorithme de *clustering* basé sur la fonction de valeur optimale du MDP sous-jacent. SCVI met ensuite à jour les états de croyance dans l'ordre décroissant d'association au *cluster* courant. Cette méthode a l'avantage d'éviter le tri systématique de la file d'attente des états de croyance requis dans le cas de PVI. Cependant, une fois de plus il est très probable qu'un état de croyance soit associé à plusieurs clusters, chaque état de croyance peut alors être mise à jour à chaque itération, et cela même si c'est inutile.

Enfin, on peut voir les algorithmes HSVI et FSVI comme des méthodes ordonnées. En effet, tous deux procèdent par simulation de trajectoires d'états de croyance suivant des heuristiques afin d'explorer l'espace des états de croyance et effectuent des mises à jour par chaînage arrière. La réalisation des mises à jour dans

l'ordre inverse des trajectoires a une importance cruciale. Comme nous le verrons plus tard, les fonctions de valeur et les états de croyance sont liés par une dépendance causale. À ce titre, la fonction de valeur d'un état de croyance successeur peut contribuer dans la fonction de valeur de l'état de croyance courant. Ainsi, la mise à jour prioritaire des états de croyance successeurs peut se traduire par un gain substantiel en performance.

3 Motivation

Tous les algorithmes ci-dessus font usage de l'opérateur de mise à jour à base de points sur un ensemble spécifique d'états de croyance afin de surmonter avec succès à la malédiction de l'historique. Jusqu'à présent cependant, toutes ces approches ont été victimes de la malédiction de la dimension lorsqu'ils sont confrontés aux domaines à très grands espaces d'états. Aucun des algorithmes énumérés ci-dessus n'utilise les caractéristiques intrinsèques des POMDPs. Ils ne considèrent guère les dépendances causales entre états notamment par le biais de la représentation graphique d'un POMDP, représentation qui selon Littman *et al.* (1995) est une propriété intrinsèque des MDPs donc des POMDPs, et potentiellement décide la complexité de leur résolution. Par exemple, la Figure 1 décrit un simple exemple de POMDP proposé par Pineau (2004), ainsi que la représentation graphique G associée. Examinons le problème à 5 états illustré en Figure 1. L'agent débute dans un état quelconque avec égale probabilité. Une fois dans ces états, la fonction d'observation (bruitée) fournit l'état actuel. En exécutant l'action a_1, l'agent se déplace aléatoirement entre les états s_1 et s_2, tandis qu'en exécutant l'action a_2 l'agent se déplace de vers les états s_3 et s_4. L'état s_5 est un état puits. La récompense est une fonction telle qu'il est bon ($+100$) de transiter vers l'état (s_3) et mauvais (-100) de transiter vers l'état s_4. La récompense est nulle partout ailleurs.

En observant ce POMDP, il nous apparaît que seul deux des composantes dans tout α-vecteur seront améliorées lors de l'application itérative des mises à jour, *i.e.,* les composantes associées aux états s_1 et s_2. Cela repose sur le fait que les composantes associées aux états s_3, s_4 et s_5 ont obtenu leurs valeurs optimales (respectivement $+100$, -100 et 0), à la première mise à jour. Par conséquent, certaines composantes des α-vecteurs peuvent avoir convergé tandis que d'autres n'y parviendront qu'après plusieurs mises à jour supplémentaires. Donc, ni les algorithmes exactes ni les méthodes approximatives cités plutôt n'échappent aux innombrables mises à jour parfois redondantes et/ou inutiles. De plus, ces algorithmes font l'usage abusif de toutes les composantes des α-vecteurs à chaque itération, bien qu'en pratiques certaines de ces composantes puissent avoir déjà atteint leurs valeurs optimales. Néanmoins, très peu d'intérêt a été accordé à ce point. À ce jour, toutes les approches tentant de résoudre ce POMDP s'y prendraient comme pour la résolution d'un quelconque POMDP à 5 états. Ils appliqueraient essentiellement la même stratégie que celle qu'ils appliqueraient pour un POMDP dont la représentation graphique correspond à une 5-clique. Ainsi, les stratégies de base de ces algorithmes n'ont aucune sous-routine intelligente, capable de distinguer différents POMDPs et d'utiliser différentes stratégies pour les résoudre. En s'inspirant de ces observations, nous voulons concevoir un algorithme général qui soit capable de découvrir la complexité intrinsèque des différents POMDPs par le biais de leur structure, afin d'utiliser différentes stratégies de mise à jour pour des POMDPs disposant de différentes propriétés structurelles.

4 Planification à base d'ordres topologiques

Dans cette section, nous proposons une description naïve de l'approche proposée. Avant toute chose, nous introduisons quelques définitions fortes utiles pour la suite. Soit M un POMDP défini par $(S, A, O, T, b_0, \gamma)$, $\{M_k\}_{k=0}^{K}$ un ensemble de sous-POMDPs tel que chaque POMDP M_k soit défini par un n-uplet $(S_k, A, O, T, \Omega, \gamma)$ où : pour tout $k = 0, 1, \cdots, K$, S_k est un sous-espace d'états de S où *couche*. Soit B_k l'ensemble des états de croyance accessibles par simulation dans M_k. Cela est possible en mémorisant à la fois un état $s \in S_k$ où l'agent est effectivement durant la simulation et l'état de croyance $b \in B_k$ de l'agent. À noter que les états de croyance $b \in B_k$ sont des distributions de probabilité sur S et pas exclusivement sur S_k. Ainsi, les trajectoires générées, seront à l'instar de FSVI, des trajectoires de paires, état de l'environnement – état de croyance (s, b). Dans les sous-sections suivantes, nous introduisons deux relations d'ordre des mises à jour : $\prec_1$ défini l'ordre suivant lequel les sous-ensembles $\{B_k\}_{k=0}^{K}$ sont traités ; et $\prec_2$ défini quant à lui l'ordre de mise à jour des états de croyance $b \in B_k$. Nous argumentons que le déterminisme de ces relations d'ordre peut nous permettre d'éviter ou d'atténuer le problème des mises à jour redondantes et/ou inutiles. Encore plus important, nous montrons que leur déterminisme ne requiert pas une grande quantité de ressources.

FIG. 1 – Un simple exemple de POMDP, sa représentation graphique G et son graphe de couches $G_{/\sim}$.

Dès lors, le gain dû à l'ordre des mises à jour se fera entièrement sentir dans le temps d'exécution global de notre algorithme lors des expérimentations.

4.1 Ordonner les sous-ensembles d'états de croyance $\{B_k\}_{k=0}^K$

Le problème du déterminisme de l'ordre $\prec_1$ du traitement des sous-ensembles $\{B_k\}_{k=0}^K$, peut être vu comme le problème de l'identification des dépendances causales entre ces sous-ensembles. Notons tout d'abord que les états de croyance et leurs fonctions de valeur sont reliés par une dépendance causale. Afin d'expliquer cette idée, considérons le POMDP M, les sous-ensembles d'états de croyance B_{k+1} et B_k. Si un état de croyance $b' \in B_{k+1}$ est un état de croyance successeur de l'état de croyance $b \in B_k$ pour une paire action – observation (a, o), alors $V(b)$ dépend de $V(b')$. En effet, la fonction de valeur sur b peut être itérativement mise à jour comme suit : $V(b) = \max_a (b \cdot r_a + \gamma \sum_o pr(o|a, b)V(\tau(a, o, b)))$. Pour cette raison, nous voulons mettre à jour b' avant b, *i.e.,* $B_{k+1} \prec_1 B_k$. Il est utile de noter que cette relation est transitive. Toutefois, les POMDPs sont en général cycliques, donc de telles relations sont fréquentes entre états de croyance. Dans ce contexte, nous tournons notre attention vers le problème du déterminisme d'une relation d'ordre $\prec_1$ pour un POMDP cyclique. Une idée employée avec succès en MDPs (Dai & Goldsmith, 2007; Bonet & Geffner, 2003; Abbad & Boustique, 2003) et adaptée ici, consiste à : grouper tous les états qui ont une dépendance causale mutuelle dans une même couche S_k ; ces couches $\{S_k\}_{k=0}^K$ constituent l'espace d'états d'un nouveau POMDP M', acyclique. Dès lors, l'ordre $\prec_1$ des mises à jour des sous-ensembles $\{B_k\}_{k=0}^K$ est donné par l'ordre topologique inverse des états $\{S_k\}_{k=0}^K$ de M'. De façon plus précise, soient deux paires (s, b) et (s', b') associées aux paires (S_k, B_k) et (S_{k+1}, B_{k+1}), respectivement, on a alors $(B_{k+1} \prec_1 B_k)$ si et seulement si $(S_{k+1} \prec_1 S_k)$ et en particulier $(b' \prec_1 b)$ si et seulement si $(s' \prec_1 s)$. Reste cependant à trouver un algorithme du déterminisme des couches $\{S_k\}_{k=0}^K$ qui induisent les sous-ensembles $\{B_k\}_{k=0}^K$. Des travaux en MDPs ont montré que le problème du calcul des couches correspond au problème du calcul des composantes fortement connexes de la représentation graphique du MDP, donc de celle du POMDP où les arrêtes liées aux observations sont omises (Dai & Goldsmith, 2007; Bonet & Geffner, 2003; Abbad & Boustique, 2003). Heureusement, il existe de nombreux algorithmes, y compris celui de Tarjan (1972), qui permettent de trouver en temps linéaire ces couches. La Figure 1 illustre le graphe des couches $G_{/\sim}$, qui est le graphe dont les nœuds sont des couches de G et les arcs $S_k \rightarrow S_{k+1}$ indiquent qu'il existe un état $s \in S_k$ capable de transiter vers un état $s' \in S_{k+1}$. Dans le graphe $G_{/\sim}$ (voir Figure 1), les couches sont $S_0 = \langle s_1 s_2 \rangle$, $S_1 = \langle s_3 \rangle$, $S_2 = \langle s_4 \rangle$, and $S_3 = \langle s_5 \rangle$. Ainsi, l'ordre topologique $\prec_1$ des mises à jour des sous-ensembles $\{B_3, B_2, B_1, B_0\}$ est alors tel que $B_3 \prec_1 B_2 \prec_1 B_1 \prec_1 B_0$.

4.2 Ordonner les mises à jour dans B_k

L'ordre topologique $\prec_1$ présenté ci-dessus est malheureusement incapable d'ordonner les mises à jour des états de croyance d'un même sous-ensemble B_k. En particulier, dans les problèmes de POMDPs où tous les états $s \in S$ ont une dépendance causale les uns avec les autres, il n'y a qu'une seule couche. C'est notamment le cas du banc d'essai `Hallway`. Dans ces situations l'ordre topologique $\prec_1$ est inutile. Pour traiter cette question nous avons recours au principe de chaînage avant ou *projections successives*. Il s'agit d'une procédure qui consiste à : (1) créer par simulation des trajectoires de paires, état de l'environnement – état de croyance, noté (s, b) ; (2) ordonner l'ordre selon lequel les états de croyance $b \in B_k$ sont générés

et mises à jour. Ce dernier point fournira l'ordre topologique $\prec_2$ recherché. Les heuristiques que nous utilisons afin de traverser l'espace des états de croyance sont basées sur la structure du POMDP et l'état réel de l'environnement dans lequel l'agent se trouve lors de la simulation. Pour déterminer un bon ordre de mise à jour des états de croyance dans un même sous-ensemble B_k, nous mémorisons certaines informations supplémentaires. Chaque état s de l'environnement doit se souvenir de ses *prédécesseurs*, *i.e.,* les états qui ont une probabilité non-nulle de transiter vers s. De plus, nous associons à chaque état s une *priorité* $pri(s)$, initialement fixée à la valeur optimale $V^{\star}_{\text{MDP}}(s)$ du MDP sous-jacent. Cette estimation sera utilisée comme une heuristique qui dirige la recherche vers les régions de l'espace d'états qui ont une forte priorité. De plus, l'état de croyance b de la paire (s, b) est mise à jour si et seulement si la priorité associée à l'état s est supérieure à une petite valeur ε préalablement définie, *i.e.,* $pri(s) > \varepsilon$. Notre intuition derrière ce critère est qu'il nous mène vers les états de croyance dont la mise à jour entraînerait une amélioration significative de la fonction de valeur. En effet, la priorité d'un état nous donne une estimation du gain escompté si l'état de croyance associé était mise à jour. À noter qu'un état sous-jacent peut-être associé à plusieurs états de croyance différents. À partir d'une paire, état sous-jacent – état de croyance (s, b), la mise à jour de V fonctionne comme décrit dans l'Algorithme 1. Soit $\hat{p}(s, s')$ la plus grande probabilité d'accéder à s'

Algorithm 1: Sous-routine de MISE À JOUR.

```
UPDATE(s, b, V)
begin
    if pri(s) > psilon then
        Mettre-à-jour l'état de croyance b : V' ← ajout(V, backup(b))
        Réinitialiser la priorité de s pri(s) = 0.
        Calculer la variation Δ = V'(b) − V(b)
        Utiliser Δ pour modifier les priorités des prédécesseurs de s
end
```

en partant de s, alors on a $\hat{p}(s, s') = \max_{(a',o')\in A\times\Omega} T(s'|s, a') \cdot O(o'|s', a')$. Si nous mettons à jour la fonction de valeur V pour une paire, état de l'environnement – état de croyance (s', b), et qu'elle change d'une quantité Δ, les prédécesseurs immédiats de s' en sont informés. Tout état s pour lequel il existe une paire action-observation (a, o) tel que $\hat{p}(s, s') \neq 0$ a sa priorité $pri(s)$ promue à $\Delta \cdot \hat{p}(s, s')$ à moins que cette dernière ait une valeur précédante supérieure.

Afin de déterminer l'état de croyance b^* suivant la paire (s, b), nous déterminons dans un premier temps le triplet état-action-observation (s^*, a^*, o^*) puis en déduisons l'état de croyance b^* comme suit : $b^* = \tau(b, a^*, o^*)$. Pour réaliser ce projet, nous devons déterminer l'état suivant s^*. Cela requiert la distinction entre deux cas de figure :

1. Si (s, b) est associée à une couche qui ne soit pas *résoluble*[1], nous cherchons à générer un chemin aussi court que possible de l'état s vers un état d'une couche dite résoluble. Étant donné G, le problème du déterminisme du plus court chemin entre toutes paires d'états est équivalent au problème du déterminisme du plus court chemin de toutes paires de nœuds $(s, s') \in S^2$ de G. Un graphe qui est tel que chaque arc est étiqueté par un coût défini comme suit :

$$cost(s, s') = \begin{cases} 1 - \hat{p}(s, s') & \text{Si } \hat{p}(s, s') > \varepsilon \\ +\infty & \text{autrement.} \end{cases}$$

   Afin de résoudre un tel problème, nous utilisons l'algorithme de Floyd-Warshall (Cormen *et al.*, 2001) qui requiert une complexité en temps de $O(|S|^3)$ et retourne à la fois la distance la plus courte entre deux états $dist(s, s')$ et l'état suivant le plus court chemin entre ces deux états $next(s, s')$. En conséquence, sachant l'état courant s et un état s_G d'une couche résoluble, l'état suivant est donné par $next(s, s_G)$. Le coût additif dû à l'exécution de l'algorithme de Floyd-Warshall est négligeable comparativement à la complexité de la résolution des POMDPs.

2. Si (s, b) est associé à une couche résoluble S_k, les trajectoires sont générées de façon ordonnée. Nous déterminons dans un premier temps un état s_G qui dispose de la plus grande priorité parmi tous les états de S_k. Puis, nous utilisons les résultats de l'algorithme de Floyd-Warshall et retournons l'état suivant $next(s, s_G)$ comme décrit dans l'algorithme 2. Dans tous les cas, nous utilisons l'état suivant

[1] Une couche est dite résoluble si elle est soit un nœud feuille de la structure du POMDP $G_{/\sim}$ ou une couche qui ne soit dépendante que de couches déjà résolues.

s^* retourné par la routine PICKNEXTSTATE pour déterminer la paire action-observation (a, o). Pour cela, nous avons à notre disposition deux stratégies : (1) la plus simple consiste à sélectionner la paire (a, o) par échantillonnage de $T(s^*|s, \cdot)$ et $O(\cdot|s^*, a)$, respectivement ; (2) la seconde stratégie consiste en la sélection de la paire (a, o) qui maximise la probabilité d'accéder à l'état s^* partant de l'état s, $(a^*, o^*) = \arg\max_{a,o} \tau(b, a, o)[s^*] \cdot pri(s^*)$.

Les trajectoires sont arrêtées lorsqu'elles aboutissent à des états dont les couches ont déjà été résolues, *i.e.,* s^*.TERMINAL $= true$. L'algorithme quant à lui se termine lorsque toutes les couches pertinentes ont été résolues ou alors que la valeur prédéfinie de la *récompense moyenne escomptée* (ADR) est atteinte. Une couche est dite résolue si tous les états relatifs ont leur priorité réduite à zéro. Cela est motivé par l'observation selon laquelle la priorité des états décrit les changements dans la fonction de valeur, donc la résolution d'une couche indique que les états de croyance associés ont convergé pour cette couche.

Algorithm 2: PICKNEXTSTATE sub-routine.

```
PICKNEXTSTATE(s, b)
begin
    pick any state s_G that belongs to a solvable layer
    if ¬s.SOLVABLE then forall s' ∈ S: s'.SOLVABLE do
        if dist(s, s_G) > dist(s, s') then s_G ← s'
        else if dist(s, s_G) = dist(s, s') then
            if pri(s_G) < pri(s') then s_G ← s'
    else find state s_G with the highest priority
    s* ← next(s, s_G)
end
```

Algorithm 3: TOP for solving POMDPs.

```
TOP(s_0, b_0)
begin
    repeat
        TOPTRIAL(s_0, b_0)
    until V has not converged
end
TOPTRIAL(s, b)
begin
    if ¬s.SOLVED then
        s* ← PICKNEXTSTATE(s, a)
        a* ← PICKACTION(s, s*)
        o* ← PICKNEXTOBSERVATION(s*, a)
        if s*.TERMINAL then break
        TOPTRIAL(s*, τ(b, a*, o*))
        UPDATE(s, b, V)
end
```

4.3 Analyse de pertinence

Une optimisation peut-être rajoutée afin de restreindre la recherche sur des parties pertinentes du POMDP. En particulier, sachant des informations additionnelles telles que l'état initial et l'état terminal (ceci peut-être aisément étendu au cas de plusieurs états initiaux et terminaux), notre algorithme TOP est capable d'élaguer les parties non pertinentes du problème comme suit :

Théorème 1
Soit M un POMDP, $\{M_k\}_k$ un ensemble de sous-POMDPs induit par les couches $\{S_k\}_k$ de M, où S_K and S_0 sont les couches associées à l'état terminal et l'état initial, respectivement. Ainsi, les affirmations suivantes sont valides :

- *(C1) : Il existe une solution au problème M si et seulement si $S_K \preceq_1 S_0$;*
- *(C2) : Une fonction de valeur de M à ε près de l'optimum peut-être obtenue en résolvant uniquement les sous-POMDPs M_k tels que : $S_K \preceq_1 S_k \preceq_1 S_0$.*

Le théorème ci-dessus défini des conditions de restriction de l'espace de recherche. La preuve de ce théorème s'appuie sur l'observation selon laquelle si la couche S_K succède selon l'ordre topologique $\prec_1$ à la couche S_0, alors il n'existe aucune trajectoire d'états de croyance partant de l'état initial à l'état terminal. En d'autres termes, il n'existe pas de solutions au POMDP M. Le critère de restriction C2 permet la sélection des couches pertinentes pour les informations initiales dont nous disposons. L'explication de ce critère réside dans l'observation selon laquelle les couches S_k en dehors de l'intervalle $S_K \preceq_1 S_k \preceq_1 S_0$ ne peuvent influencer la fonction de valeur retournée.

5 Évaluations empiriques

Nous allons maintenant évaluer la performance de l'algorithme TOP en comparaison aux autres méthodes parmi les plus récentes incluant, HSVI (Smith & Simmons, 2005) ; FSVI ; PVI et PBVI. Nous avons testé ces algorithmes sur plusieurs bancs d'essai de la communauté des POMDPs, y compris : TigerGrid (Cassandra *et al.*, 1994) ; des problèmes de navigations , *e.g.,* Hallway, Hallway2 and Mit (Cassandra *et al.*, 1994) ; ou encore les domaines de Rock Sample (Smith & Simmons, 2004). Nous utilisons le code source de Guy Shani qui implémente tous ces algorithmes, code auquel nous avons ajouté notre algorithme. Les expérimentations ont été effectuées sur la même plateforme Intel Core Duo 1.83GHz CPU avec un processeur 1Gb de mémoire principale.

Nous évaluons ces algorithmes selon les critères suivants : **évaluation de la fonction de valeur–** récompense moyenne escomptée (ADR) : $\frac{\sum_{i=0}^{\#trials}\sum_{j=0}^{\#steps}\gamma^j r_j}{\#trials}$. L'ADR est filtré à l'aide du filtre du premier ordre pour réduire les bruits dans cet estimé. Comme nous nous intéressons à la vitesse de convergence, chaque algorithme a été exécuté jusqu'à ce que sa fonction de valeur atteigne un ADR prédéfini. Nous avons également reporté la valeur de l'état de croyance initial b_0 afin de souligner les effets des différentes stratégies sur la structure de la fonction de valeur. **Temps d'exécution –** nous avons reporté le temps CPU, cependant comme il s'agit de la mise en œuvre d'une implémentation spécifique, nous avons également reporté la quantité d'opérations de mise à jour effectuées. **Mémoire –** nous reportons également le nombre d'hyperplans de la fonction de valeur finale.

De façon similaire à TOP, les algorithmes HSVI et FSVI utilisent tous deux des trajectoires d'états de croyance et ne différent que par la manière dont ces trajectoires sont générées et l'ordre des mises à jour sur les états de croyance. Comme le montre la Table 1, TOP est plus rapide que HSVI, FSVI sur la majeure partie des bancs d'essai et il est très compétitif sur les autres. Bien que nos résultats ne soient que préliminaires à ce jour et ils ne doivent pas être utilisé comme point de comparaison, ils sont pour le moins très encourageants. En comparaison à HSVI et FSVI, TOP réalise le plus petit nombre de mises à jour sur la majeure partie des bancs d'essai. Plus important encore, en n'effectuant des mises à jour que lorsque la priorité des états du MDP sous-jacent est supérieure à ε, TOP réduit considérablement le nombre de mise à jour nécessaires. On remarque, cependant, que si l'ADR de tous les algorithmes était à peu près égal pour tous les bancs d'essai, la valeur de l'état de croyance initial $V(b_0)$ varie considérablement d'un algorithme à un autre, voir par exemple le domaine RockSample $(5, 7)$ à la Table 1. Cela s'explique par la définition des modèles des bancs d'essai, modèles où bien souvent l'état terminal est inexistant. Nous avons également noté que FSVI souffrait du manque d'exploration. En effet, FSVI utilise la Q-valeur du MDP sous-jacent afin de traverser l'espace des états de croyance. Malheureusement, comme la Q-valeur est fixe, FSVI n'explore qu'un petit ensemble des trajectoires possibles, donc il lui est difficile d'améliorer sa fonction de valeur notamment lorsque certaines des trajectoires non explorées peuvent contribuer pour beaucoup à la fonction de valeur. TOP est également plus rapide que les algorithmes PVI et PBVI. Comme nous nous intéressons à la vitesse de convergence, chaque algorithme a été exécuté jusqu'à ce que sa fonction de valeur atteigne l'ADR prédéfini, peu importe le nombre d'états de croyance nécessaires. Tandis que PBVI procède en mettant à jour un ensemble d'états de croyance puis en étendant cet ensemble et ainsi de suite, PVI essaie, 'a coût négligeable, de classer par ordre de priorité ses séquences de mise à jour. Si l'on examine leurs performances par rapport à TOP, nous observons que ces deux algorithmes requièrent plus de temps afin d'atteindre l'ADR prédéfini. Bien que PVI peut effectuer un petit nombre de mises à jour, *e.g.,* Table TigerGrid 1, l'effort requis pour ordonner les séquences d'états de croyance est prohibitif. Les performances de TOP sur l'ensemble des bancs d'essai sont très encourageantes, même si nous notons que les calculs dus à l'algorithme de Floyd-Warshall peuvent entraîner des coûts additifs excessifs. Pour cette raison, nous suggérons l'usage de l'algorithme de Drijkstra, afin de déterminer les plus courts chemins d'un petit nombre de paires d'états. Actuellement, nous examinons cette question, en espérant réduire l'effort

requis par cette sous-routine notamment pour des problèmes encore plus larges.

Méthode	ADR	$V(b_0)$	$\|V\|$	Time (sec)	#mise à jour
TigerGrid					
HSVI	0.68	1.75	1337	535	1695
FSVI	0.66	1.89	1608	380	1275
PBVI	0.66	2.15	472	3749	33101
PVI	0.60	0.69	357	> 3530	480
TOP	0.68	1.05	755	115	640
Hallway					
HSVI	0.51	0.88	258	391	886
FSVI	0.51	0.89	317	440	969
PBVI	0.51	0.75	63	59	1447
PVI	0.51	0.69	185	3272	500
TOP	0.52	0.60	213	56	352
Hallway 2					
HSVI	0.35	0.16	128	126	145
FSVI	0.36	0.24	252	76	204
PBVI	0.35	0.26	128	347	1355
PVI	0.35	0.27	307	> 9565	400
TOP	0.35	0.12	100	21	98
Rock Sample $(4, 4)$					
HSVI	18.036	17.92	173	11	239
FSVI	18.036	13.63	79	7	98
TOP	18.036	13.63	40	3	68
Rock Sample $(5, 5)$					
HSVI	19.237	19.24	50	22	60
FSVI	17.840	13.84	185	200	307
TOP	19.204	17.20	61	37	88
Rock Sample $(5, 7)$					
HSVI	24.202	22.96	208	2754	461
FSVI	22.198	14.97	353	4057	384
TOP	24.086	21.19	317	2791	436
Mit					
HSVI	0.88	0.77	1786	17344	2288
FSVI	0.86	0.69	339	1213	1275
TOP	0.88	0.44	520	820	385

TAB. 1 – Performance measurements.

6 Conclusion

Nous avons introduit un nouvel algorithme de résolution des POMDPs à horizon infini, nommé *planification à base d'ordres topologiques* (TOP). Cet algorithme étudie les propriétés structurelles des POMDPs afin de déterminer la meilleure séquence des mises à jour. Il est basé sur l'idée selon laquelle les différents POMDPs disposent de structures graphiques différentes, or ces dernières déterminent la complexité intrinsèque de la résolution d'un POMDP. Suivant cette observation, TOP parvient à déterminer deux relations d'ordre comme outils pour faire face à la fois à la malédiction de la dimension mais aussi à celle de l'historique. Ces ordres topologiques permettent non seulement d'organiser les mises à jour de la fonction de valeur mais aussi à orienter l'exploration de l'espace des états de croyance, évitant ainsi les mises à jour redondantes et les sous-espaces d'états de croyance non pertinents.

7 Remerciement

Nous remercions Guy Shani pour les discussions constructives sur ce document et pour la mise à notre disposition de son code source.

Références

ABBAD M. & BOUSTIQUE H. (2003). A decomposition algorithm for limiting average markov decision problems. *Oper. Res. Lett.*, **31**(3), 473–476.
BELLMAN R. E. (1957). *Dynamic Programming*. Dover Publications, Incorporated.
BONET B. & GEFFNER H. (2003). Faster heuristic search algorithms for planning with uncertainty and full feedback. In *IJCAI*, p. 1233–1238.
CASSANDRA A., LITTMAN M. L. & ZHANG N. L. (1997). Incremental Pruning : A simple, fast, exact method for partially observable Markov decision processes. In *UAI*, p. 54–61.
CASSANDRA A. R., KAELBLING L. P. & LITTMAN M. L. (1994). Acting optimally in partially observable stochastic domains. In *AAAI*, p. 1023–1028.
CORMEN T. H., LEISERSON C. E., RIVEST R. L. & STEIN C. (2001). *Introduction to Algorithms, Second Edition*. The MIT Press.
DAI P. & GOLDSMITH J. (2007). Topological value iteration algorithm for Markov decision processes. In *IJCAI*, p. 1860–1865.
LITTMAN M. L., DEAN T. & KAELBLING L. P. (1995). On the complexity of solving Markov decision problems. In *UAI*, p. 394–402.
LOVEJOY W. S. (1991). Computationally feasible bounds for partially observable markov decison processes. *Operations Research*, **39**, 175–192.
PINEAU J. (2004). *Tractable Planning Under Uncertainty : Exploiting Structure*. PhD thesis, Robotic Institute. Carnegie Mellon University., Pittsburgh.
PINEAU J., GORDON G. & THRUN S. (2003). Point-based value iteration : An anytime algorithm for POMDPs. In *IJCAI*.
POUPART P. & BOUTILIER C. (2004). VDCBPI : an approximate scalable algorithm for large POMDPs. In *NIPs*.
ROY N., GORDON G. & THRUN S. (2005). Finding approximate pomdp solutions through belief compression. *JAIR*, **23**, 1–40.
SHANI G., BRAFMAN R. I. & SHIMONY S. E. (2006). Prioritizing point-based pomdp solvers. In *ECML*, p. 389–400.
SHANI G., BRAFMAN R. I. & SHIMONY S. E. (2007). Forward search value iteration for pomdps. In *IJCAI*, p. 2619–2624.
SMITH T. & SIMMONS R. (2004). Heuristic search value iteration for pomdps. In *UAI*, p. 520–527, Arlington, Virginia, United States.
SMITH T. & SIMMONS R. G. (2005). Point-based POMDP algorithms : Improved analysis and implementation. In *UAI*.
SONDIK E. J. (1978). The optimal control of partially observable markov decision processes over the infinite horizon : Discounted cost. *Operations Research*, **12**, 282–304.
SPAAN M. T. J. & VLASSIS N. (2005). Perseus : Randomized point-based value iteration for POMDPs. *JAIR*, **24**, 195–220.
TARJAN R. (1972). Depth-first search and linear graph algorithms. *SIAM J. Comput.*, **1**(2).
VIRIN Y., SHANI G., SHIMONY S. E. & BRAFMAN R. I. (2007). Scaling up : Solving pomdps through value based clustering. In *AAAI*.
WINGATE D. & SEPPI K. D. (2005). Prioritization methods for accelerating mdp solvers. *J. Mach. Learn. Res.*, **6**, 851–881.