Résolution de PDMF^3 : processus décisionnels de Markov à transitions, récompenses et politiques stochastiques factorisées*

Julia Radoszycki, Nathalie Peyrard, Régis Sabbadin

INRA-MIAT (UR 875), F-31326, Castanet-Tolosan, France. prenom.nom@toulouse.inra.fr

Abstract : Les processus décisionnels de Markov à espaces d'état et d'action factorisés, généralement appelés PDMF-AF, fournissent un cadre riche pour modéliser les problèmes de décision séquentielle sous incertitude dont l'espace d'état et d'action sont de grande dimension et hautement structurés, comme en robotique, en biologie de la conservation, ou en épidémiologie du paysage. Cependant, même les algorithmes de résolution dédiés (exacts ou approchés) ne s'appliquent pas quand les dimensions des espaces d'état et d'action excèdent toutes les deux 20 ou 30, sauf sous des hypothèses fortes sur les transitions des états ou sur la fonction de valeur. Dans cet article, nous introduisons le cadre PDMF^3 et des algorithmes de résolution approchée associés qui peuvent traiter des problèmes de beaucoup plus grande dimension. Un PDMF^3 est un PDMF-AF dont la fonction de récompense se décompose de manière additive et dont les politiques solutions sont contraintes à être factorisées et peuvent être stochastiques. Les algorithmes que nous proposons appartiennent à la famille des algorithmes d'itération de la politique et exploitent des outils d'optimisation continue. Nous les validons sur des expériences numériques approfondies. Sur des petits problèmes, pour lesquels la politique optimale est disponible, ils fournissent des politiques proches de la politique optimale. Sur des problèmes de plus grande taille de la sous-classe des PDMG, ils fournissent des résultats de qualité similaire à ceux obtenus avec des algorithmes de l'état de l'art. Enfin, nous montrons que nos algorithmes peuvent traiter des PDMF^3 de grande taille. Ils permettent en effet de résoudre des problèmes d'épidémiologie agricole dont l'espace d'état et d'action sont tous les deux de taille 2^{100}.

Mots-clés : processus décisionnels de Markov, méthodes de *policy gradient*, inférence dans les modèles graphiques.

1 Introduction

Les processus décisionnels de Markov (PDM) fournissent un outil intéressant pour modéliser et résoudre des problèmes de décision séquentielle sous incertitude (Puterman, 1994). Cependant, une application à des domaines comme la robotique, la gestion environnementale, etc. n'est pas directe quand les représentations de l'état et de l'action sont factorisées. Plusieurs algorithmes ont été proposés pour les PDM dont les espaces d'état et d'action sont factorisés (PDMF-AF, (Guestrin *et al.*, 2001; Kim & Dean, 2002)). Cependant, en général ils ne s'appliquent pas à des problèmes dont le nombre de variables d'état et d'action excèdent tous les deux 20 ou 30, sauf sous des hypothèses drastiques sur les transitions des états ou la fonction de valeur (voir section 2).

Dans cet article, nous proposons un nouveau cadre, appelé PDMF^3, pour la résolution approchée de PDMF-AF généraux à récompense additive. F^3 signifie triplement factorisé, puisque nous considérons des politiques stochastiques factorisées en plus d'une fonction de transition et d'une fonction de récompense factorisées. Pour résoudre les PDMF^3, nous proposons un algorithme générique de type itération de la politique. Nous montrons que l'évaluation de politiques stochastiques factorisées revient à calculer des probabilités marginales dans un réseau Bayésien, ce qui peut être fait de manière exacte ou approchée, en utilisant des outils de l'état de l'art. L'étape de mise à jour de la politique revient à effectuer de petits changements dans les paramètres des politiques stochastiques, soit selon une seule coordonnée, soit selon le

*Ce travail a été financé par le projet ANR-13-AGRO-0001-04.

gradient de la valeur de la politique. Ce dernier peut se calculer soit par différences finies, soit en calculant les marginales d'un réseau Bayésien modifié.

Dans la Section 2, nous donnons quelques repères sur les PDMF-AF et sur la résolution de PDM factorisés ou de Dec-POMDP factorisés, en lien avec notre approche. Dans la Section 3, nous présentons notre principale contribution. Nous introduisons le cadre PDMF^3 et l'algorithme générique de type itération de la politique, basé sur une méthode de gradient. Dans la Section 4, nous présentons des expériences numériques conséquentes sur des PDMF-AF, aléatoires ou inspirés par un problème d'épidémiologie agricole.

2 Contexte

2.1 PDMF-AF

La méthode que nous proposons dans cet article est dédiée à la résolution de PDM stationnaires à espaces d'état et d'action factorisés (PDMF-AF), à horizon fini ou infini, dont la fonction de transition est représentée par un réseau Bayésien dynamique (Murphy, 2002). Nous considérons le cas de PDMF-AF dont la fonction de récompense est une somme de fonctions de faible arité. Un tel PDMF-AF est un PDM $< \mathcal{S}, \mathcal{A}, P, R, T >$ avec :

- Espace d'état factorisé: $\mathcal{S} = \prod_{i=1}^{n} \mathcal{S}_i$, où chaque $\mathcal{S}_i$ est un ensemble fini. L'état du système au temps $t \in \{0, \ldots, T\}$ est noté: $s^t = (s_1^t, ..., s_n^t) \in \mathcal{S}$.
- Espace d'action factorisé: $\mathcal{A} = \prod_{j=1}^{m} \mathcal{A}_j$, où chaque $\mathcal{A}_j$ est un ensemble fini. L'action au temps $t \in \{0, \ldots, T-1\}$ est notée: $a^t = (a_1^t, ..., a_m^t) \in \mathcal{A}$.
- Fonction de transition factorisée: $\forall t \in \{0, \ldots, T-1\}$, $P(s^{t+1}|s^t, a^t) = \prod_{i=1}^{n} P_i(s_i^{t+1}|pa_P(s_i^{t+1}))$ où $pa_P(s_i^{t+1}) = pa_P^S(s_i^{t+1}) \bigcup pa_P^A(s_i^{t+1})$, $pa_P^S(s_i^{t+1}) \subset \{s_j^t, j = 1...n, s_{j'}^{t+1}, j' = 1...n, j' \neq i\}$ et $pa_P^A(s_i^{t+1}) \subset \{a_k^t, k = 1...m\}$. Les arcs synchrones sont autorisés, mais le graphe dirigé sous-jacent doit être acyclique.
- Fonction de récompense additive: $R(s^t, a^t) = \sum_{\alpha=1}^{r} R_\alpha\left(pa_R(R_\alpha^t)\right)$, où $pa_R(R_\alpha^t) \subset \{s_j^t, j = 1...n\} \cup \{a_k^t, k = 1...m\}$. $R(s^t, a^t)$ est supposée positive et bornée par $R_{\max}$.
- $0 < T \leq +\infty$ l'horizon du problème.

Une politique stationnaire pour un PDM est définie comme une fonction $\delta : \mathcal{S} \mapsto \mathcal{A}$, qui décide de l'action globale a^t selon l'état global s^t. Sa valeur pour un état initial donné $s^0 \in \mathcal{S}$ est $V_\delta^{R,T}(s^0) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R^t \mid s_0, \delta\right]$, où $0 < \gamma \leq 1$ est un *facteur d'amortissement* donné[1]. Résoudre un PDM revient à trouver la politique δ^* de valeur maximale pour tout état initial : $\forall s^0 \in \mathcal{S}, \forall \delta : \mathcal{S} \mapsto \mathcal{A}, V_{\delta^*}(s^0) \geq V_\delta(s^0)$ (il existe toujours une telle politique déterministe (Puterman, 1994)).

Comme les solutions optimales des PDMF-AF sont hors de portée, nous n'essaierons pas de trouver une politique qui soit optimale pour tout état initial. Nous considérerons une distribution initiale donnée sur les états, $P^0(s^0)$, et rechercherons une politique maximisant la valeur espérée pour la distribution initiale P^0 : $V_\delta^{R,T}(P^0) = \sum_{s^0} P^0(s^0) V_\delta^{R,T}(s^0)$. Pour pouvoir être stockée en mémoire, la distribution initiale doit être représentée comme un réseau bayésien. De plus, nous limiterons notre recherche aux politiques stationnaires, même dans le cas d'un horizon fini où la politique optimale n'est pas forcément stationnaire, puisqu'elles peuvent être décrites de manière plus concise et sont plus faciles à interpréter et à calculer.

2.2 État de l'art

Dans les PDMF-AF, les politiques optimales, déterministes, sont décrites par une liste de fonctions δ_j^*: $\mathcal{S} \to \mathcal{A}_j$. Calculer, et même représenter, de telles politiques globales peut rapidement devenir difficile quand n augmente. C'est pourquoi la plupart des approches de résolution de PDMF-AF de grande taille ont tenté de remédier à cela en calculant des politiques approchées. Ces politiques sont soit (i) décrites par des arbres de décision ou des diagrammes de décision, (ii) des politiques factorisées, $\left(\prod_{i \in I_j} \mathcal{S}_i\right) \to \mathcal{A}_j$, où

[1] Dans le cas d'un horizon fini, on peut avoir $\gamma = 1$.

tous les $I_j \subset \{1, \ldots, n\}$ sont de faible cardinal, ou (iii) des politiques factorisées paramétrées, dont chaque politique locale est en plus paramétrée.

Dans la première famille d'approches, des algorithmes efficaces ont été proposés pour les PDMs à espace d'état factorisé (SPUDD (Hoey *et al.*, 1999) et APRICODD (St-aubin *et al.*, 2000)) mais ils ne passent pas à l'échelle quand l'espace d'action est aussi factorisé et de grande dimension. Moins d'attention a été portée au cas plus difficile où à la fois l'espace d'état et l'espace d'action sont factorisés. (Raghavan *et al.*, 2012) a proposé un algorithme de programmation dynamique symbolique pour PDMF-AF, qui a le même inconvénient : retourner une politique globale, représentée sous forme d'arbre, qui peut potentiellement prendre un espace exponentiel pour être représentée.

Dans la seconde famille d'approches, le cadre des POMDP décentralisés (Dec-POMDPs, (Bernstein *et al.*, 2002)) considère un espace d'action factorisé et des observations locales, conduisant à des politiques factorisées. Cependant, l'espace d'état est le plus souvent *non factorisé* et de petite taille, comparé à l'espace d'action (Dibangoye *et al.*, 2014). Il y a quelques exceptions, cependant, comme Dibangoye *et al.* (2012), qui considère des observations et des transitions indépendantes. Kumar *et al.* (2011) suppose que $V_\delta^{R,T}(s^0)$ s'écrit comme une somme de fonctions de faible arité par rapport à l'état initial, une propriété qui est vraie pour certaines sous-classes de Dec-POMDPs, principalement quand la fonction de transition présente une propriété d'indépendance. Ils proposent un algorithme EM qui passe à l'échelle, et résout des problèmes où $|\mathcal{S}| \approx 2^{58}$, $|\mathcal{A}| \approx 2^{40}$, $|\Omega| \approx 2^{47}$.

Toujours dans la seconde famille, Oliehoek *et al.* (2013) propose une approche basée sur les jeux Bayésiens collaboratifs graphiques. Le facteur limitant de cette approche est l'horizon T (l'horizon le plus grand considéré est $T = 6$, pour 100 variables d'action). Les PDM sur graphe (PDMG, (Sabbadin *et al.*, 2012; Cheng *et al.*, 2013)) forment une sous-classe de PDMF-AF qui appartient aussi à la seconde famille. Dans le cadre des PDMG, une seule variable d'action et une seule fonction de récompense sont associées à chaque variable d'état, et récompense, transition et politique sont supposées avoir la même structure de factorisation. Les algorithmes proposés dans le cadre PDMG font soit une approximation de la fonction de valeur comme une somme de fonctions de faible arité par rapport à l'état initial (Forsell & Sabbadin, 2006; Peyrard & Sabbadin, 2006), soit une approximation multiplicative (Cheng *et al.*, 2013).

Dans la troisième famille d'approches, Sallans & Hinton (2004) et Buffet & Aberdeen (2009) proposent des algorithmes d'apprentissage par renforcement qui optimisent des politiques paramétrées pour les PDMF-AF. L'approche de Sallans & Hinton (2004) a été appliquée à des problèmes avec au plus 40 variables d'action binaires (et moins de variables d'état). L'algorithme de gradient basé sur des simulations proposé par Buffet & Aberdeen (2009) peut s'appliquer à des problèmes de plus grande taille. Cependant, il ne permet pas de résoudre des problèmes avec des milliers de paramètres, ce qui est le cas quand on considère des politiques stochastiques factorisées comme dans notre approche.

L'approche que nous proposons est similaire dans l'esprit aux deux dernières approches, puisque nous traitons des PDMF-AF généraux, et que nous utilisons des approximations des probabilités marginales du réseau Bayésien dynamique sous-jacent pour évaluer les politiques. Mais, tandis que l'algorithme EM de Kumar *et al.* (2011) est de type *planification par inférence*, puisqu'il modélise directement le problème d'optimisation de la politique comme un problème d'inférence, nous utilisons seulement les méthodes d'inférence pour évaluer les politiques et maintenons des phases de mise à jour des paramètres. Autant que nous le sachions, aucune approche de planification par inférence n'a été proposée qui permette de résoudre des PDMF-AF généraux avec autant de variables que ce que nous faisons dans nos expérimentations numériques.

Dans la section suivante, nous proposons une nouvelle approche, appartenant à la seconde famille, où on considère des *politiques stochastiques factorisées* avec une structure définie *a priori*.

3 PDMF^3

3.1 Politiques stochastiques factorisées et PDMF^3

Un PDMF^3 est un PDMF-AF à fonction de récompense additive et politiques stochastiques factorisées (PSF). Les PSF généralisent les politiques factorisées déterministes. Elles sont définies par : $\delta(a^t|s^t) = \prod_{j=1}^m \delta_j(a_j^t|pa_\delta(a_j^t))$ où les $\delta_j(\cdot|pa_\delta(a_j^t))$ sont des distributions conditionnelles de probabilité et $pa_\delta(a_j^t) \subset \{s_i^t, i = 1...n\} \cup \{a_k^t, k = 1...m, k \neq j\}$. Nous considérons uniquement les politiques factorisées pour lesquelles le graphe des dépendances correspondant à la fois à la transition et à la PSF est acyclique (voir

Figure 1). Le problème que nous cherchons à résoudre est celui de l'optimisation de PSF de structure donnée. Ce type de problème émerge naturellement quand les agents ont une observabilité partielle de l'état joint. Dans le cas d'une observabilité totale, choisir la meilleure structure de politique, *i.e.* définir $\{pa_\delta(a_j^t), j = 1...m\}$ est une question importante et complexe que nous laissons pour de futures recherches.

La Figure 1 donne un exemple de structure de transition, de politique et de récompense pour un PDMF[3] à deux variables d'état et d'action où $pa_\delta(a_1^t) = \{s_1^t\}$ et $pa_\delta(a_2^t) = \{a_1^t, s_2^t\}$.

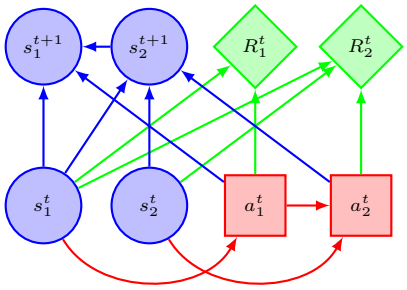

Figure 1: Exemple de structure de transition (en bleu), de récompense (en vert) et de politique (en rouge) pour un PDMF[3] à deux variables d'état et deux variables d'action. $pa_P(s_1^{t+1}) = \{s_1^t, a_1^t, s_2^{t+1}\}$, $pa_P(s_2^{t+1}) = \{s_1^t, s_2^t, a_2^t\}$, $pa_R(R_1^t) = \{s_1^t, a_1^t\}$, $pa_R(R_2^t) = \{s_1^t, s_2^t, a_2^t\}$, $pa_\delta(a_1^t) = \{s_1^t\}$ et $pa_\delta(a_2^t) = \{a_1^t, s_2^t\}$.

Une PSF donnée δ, une fonction de transition factorisée et une distribution initiale P^0 données, définissent conjointement un réseau Bayésien dynamique dans lequel la probabilité d'une trajectoire $(s,a)^{0:T} =< s^0, a^0, ...a^{T-1}, s^T >$ est:

$$P_\delta^T((s,a)^{0:T}) = P^0(s^0) \times \prod_{t=0}^{T-1} \left(\prod_{i=1}^{n} P_i(s_i^{t+1}|pa_P(s_i^{t+1})) \prod_{j=1}^{m} \delta_j(a_j^t|pa_\delta(a_j^t)) \right).$$

La valeur d'une PSF donnée δ pour une distribution initiale donnée P^0, dans le cas d'un horizon fini T[2] est définie par:

$$V_\delta^{R,T}(P^0) = \sum_{(s,a)^{0:T}} P_\delta^T((s,a)^{0:T}) \left(\sum_{t=0}^{T} \gamma^t R(s^t, a^t) \right).$$

Calculer cette espérance nécessite d'évaluer seulement certaines marginales de $P_\delta^T((s,a)^{0:T})$:

$$V_\delta^{R,T}(P^0) = \sum_{t=0}^{T} \gamma^t \sum_{\alpha=1}^{r} \sum_{pa_R(R_\alpha^t)} b_\alpha^t(pa_R(R_\alpha^t)) R_\alpha(pa_R(R_\alpha^t)),$$

où $b_\alpha^t(\cdot)$ est la distribution marginale des variables influençant la fonction de récompense α au temps t. Elle se calcule en marginalisant $P_\delta^T((s,a)^{0:T})$ par rapport à toutes les autres variables d'état et d'action dans la trajectoire $(s,a)^{0:T}$. Remarquons que la valeur s'écrit comme une somme de fonctions de faible arité de portées $\{pa_R(R_\alpha^t)\}_{\alpha \in \{1...r\}, t \in \{0,...,T\}}$. Cette décomposition est exacte et découle directement des hypothèses du cadre.

Il exitse un certain nombre d'algorithmes implémentés pour calculer de manière exacte ou approchée les marginales $b_\alpha^t(\cdot)$, comme l'algorithme exact *Junction Tree* (JT, voir par exemple (Yedidia *et al.*, 2005)), pour des problèmes de faible largeur d'arbre, ou l'algorithme approché *Loopy Belief Propagation* (LBP, (Frey & MacKay, 1998)). Des algorithmes de type Monte-Carlo (MC) peuvent aussi être utilisés pour approcher directement l'espérance.

[2]Dans le cas d'un horizon infini avec facteur d'amortissement, la somme infinie peut être approchée par une somme sur un horizon fini T suffisamment grand.

3.2 Optimisation de politiques stochastiques factorisées dans les PDMF[3]

Quand on cherche des PSF optimales, on est face à un problème d'optimisation continue sous contraintes:
Problème d'optimisation dans un PDMF[3].

$$\begin{array}{ll} \underset{\bar{\delta}\in(\mathbb{R}^+)^N}{\text{maximiser}} & V_{\bar{\delta}}^{R,T}(P^0) \\ \text{s. c.} & \sum_{a_j\in\mathcal{A}_j} \delta_j(a_j|pa_\delta(a_j)) = 1,\ \forall j, \forall pa_\delta(a_j) \end{array}$$

où $\bar{\delta}$ est le vecteur de coordonnées$\{\delta_j(a_j|pa_\delta(a_j)),\ \forall j, \forall pa_\delta(a_j)\}$.
Le nombre de coordonnées de $\bar{\delta}$ est:

$$N = \sum_{j=1}^{m} |\mathcal{A}_j| \prod_{k/A_k\in pa_\delta^A(a_j)} |\mathcal{A}_k| \prod_{k/S_k\in pa_\delta^S(a_j)} |\mathcal{S}_k| \tag{1}$$

Quand $\forall j = 1...m, |\mathcal{A}_j|= |\mathcal{A}_1|, |pa_\delta^A(a_j)|= |pa_\delta^A(a_1)|= z$ et $\forall i = 1...n, |\mathcal{S}_i|= |\mathcal{S}_1|$, $|pa_\delta^S(a_j)|= |pa_\delta^S(a_1)|= y$, on a

$$N = m|\mathcal{A}_1|^{z+1}|\mathcal{S}_1|^y$$

ce qui est beaucoup plus petit que le nombre de paramètres d'une politique globale déterministe ($m|\mathcal{S}|= m|\mathcal{S}_1|^n$) quand $z << m$ et $y << n$.

Théoreme. *Le problème de décision associé au problème d'optimisation dans un PDMF*[3] *est* NP^{PP}*-difficile.*
La preuve est omise pour des raisons de place. Elle utilise une réduction du problème $EMAJSAT$, qui est connu pour être NP^{PP}-difficile (Littman *et al.*, 1998). $EMAJSAT$ est le problème qui consiste, à partir d'une formule SAT sur les variables $\{X_i\}$ et $\{Y_j\}$, à décider si il existe une instanciation x^* pour laquelle la majorité des instaciations de Y satisfait la formule.

3.3 Algorithmes d'itération de la politique approchée

Le problème d'optimisation dans un PDMF[3] ne pourra pas être résolu de manière exacte en pratique puisque (i) nous aurons seulement accès la plupart du temps à une approximation de $V_{\bar{\delta}}^{R,T}(P^0)$ et (ii) le problème peut avoir plusieurs maxima locaux puisqu'il n'est *a priori* pas convexe (la plupart des algorithmes d'optimisation continue sont des algorithmes d'optimisation locale qui n'ont la garantie de converger vers un optimum global que quand la fonction objectif est convexe). Nous proposons donc une famille d'algorithmes de type itération de la politique, alternant des *évaluations approchées* et des *améliorations locales* des politiques:

- Etape d'évaluation: étant donnée la PSF courante δ^q, $V_{\bar{\delta}^q}^{R,T}(P^0)$ est évaluée de manière exacte ou approchée, en utlisant par exemple l'algorithme JT, LBP ou MC.
- Etape de mise à jour: δ^q est améliorée en δ^{q+1}, en utilisant soit une montée de gradient (GA pour *gradient ascent* (Luenberger & Ye, 2008)) soit une montée par coordonnées (CA pour *coordinate ascent* (Luenberger & Ye, 2008)).

3.3.1 Algorithmes de montée de gradient (GA)

Les algorithmes de montée de gradient fournissent un argument maximum local $\bar{\delta}$ de la valeur (éventuellement approchée) $V_{\bar{\delta}}^{R,T}(P^0)$[3]. Une simple reparamétrisation permet de se débarasser des contraintes:

$$\forall j, \forall a_j, \forall pa_\delta(a_j)\ \delta_j(a_j|pa_\delta(a_j)) = \frac{e^{\theta_j(a_j|pa_\delta(a_j))}}{\sum_{a'_j\in\mathcal{A}_j} e^{\theta_j(a'_j|pa_\delta(a_j))}},$$

[3]Ils fournissent en réalité un point critique, *i. e.* un point d'annulation du gradient, ce qui est une condition nécessaire mais pas suffisante pour avoir un maximum local.

où les coordonnées $\theta_j(a_j|pa_\delta(a_j))$ prennent des valeurs réelles. θ a la même structure que δ, et nous définissions le vecteur de coordonnées réelles $\bar{\theta}$ de manière analogue à $\bar{\delta}$ (tous les deux ont le même nombre de coordonnées).

Une fois reparamétré, le problème d'optimisation devient:

$$\underset{\bar{\theta}\in\mathbb{R}^N}{\text{maximiser}} V_{\bar{\theta}}^{R,T}(P^0) \tag{2}$$

où N est défini par l'Equation (1).

Utiliser une approche de montée de gradient pour résodure le problème de maximisation de l'Equation (2) nécessite d'être capable de calculer le gradient $\nabla_{\bar{\theta}} V_{\bar{\theta}}^{R,T}(P^0)$, soit analytiquement, soit numériquement. On peut montrer que les composantes $\frac{\partial V_{\bar{\theta}}^{R,T}(P^0)}{\partial \bar{\theta}_k}$ peuvent s'exprimer à partir de marginales d'un réseau bayésien dynamique, dérivé de celui définissant $V_{\bar{\theta}}^{R,T}(P^0)$. En théorie, le gradient peut donc être calculé de la même manière que la valeur $V_{\bar{\theta}}^{R,T}(P^0)$ elle même, en utilisant par exemple l'algorithme JT. Cependant, cela est souvent trop coûteux en pratique. Nous présentons donc une approximation du gradient par différences finies, que nous avons utilisée dans les expériences numériques: $\forall k = 1...N$,

$$\frac{\partial V_{\bar{\theta}}^{R,T}(P^0)}{\partial \bar{\theta}_k} \approx \frac{V_{\bar{\theta}^+}^{R,T}(P^0) - V_{\bar{\theta}}^{R,T}(P^0)}{\epsilon}, \tag{3}$$

où $\bar{\theta}_k^+ = \bar{\theta}_k + \epsilon$ et $\forall l \neq k\ \bar{\theta}_l^+ = \bar{\theta}_l$. $V_{\bar{\theta}}^{R,T}(P^0)$ et $V_{\bar{\theta}^+}^{R,T}(P^0)$ sont calculées de manière exacte ou approchée, en utlisant JT, LBP ou MC. ϵ est un pas petit et positif. Remarquons qu'un changement d'une seule coordonnée $\bar{\theta}_k$ de $\bar{\theta}$ induit un changement dans plusieurs coordonnées de $\bar{\delta}$. En effet, la politique $\bar{\delta}^+$ associée à $\bar{\theta}^+$ est donnée par:

$$\begin{aligned}
\bar{\delta}_k^+ &= \frac{\bar{\delta}_k e^\epsilon}{1 + (e^\epsilon - 1)\bar{\delta}_k} \\
\bar{\delta}_g^+ &= \frac{\bar{\delta}_g}{1 + (e^\epsilon - 1)\bar{\delta}_k}, \forall g \in G(k) \\
\bar{\delta}_g^+ &= \bar{\delta}_g, \forall g \notin G(k),
\end{aligned} \tag{4}$$

où $G(k)$ est défini comme suit: si $\bar{\delta}_k = \delta_j(a_j|pa_\delta(a_j))$, alors $g \in G(k)$ *si et seulement si* $\exists a'_j \neq a_j$, $\bar{\delta}_g = \delta_j(a'_j|pa_\delta(a_j))$. Une fois calculée l'approximation du gradient par différences finies, la mise à jour du vecteur de paramètres $\bar{\theta}$ est donnée par:

$$\bar{\theta}^{q+1} = \bar{\theta}^q + \eta_q \nabla_{\bar{\theta}^q} V_{\bar{\theta}^q}^{R,T}(P^0) \tag{5}$$

Il existe plusieurs façons de choisir le pas η_q utilisé dans l'Equation (5) pour mettre à jour $\bar{\theta}^q$ dans la direction du gradient à l'itération q (Luenberger & Ye, 2008). Il peut être *fixé* une fois pour toutes. Il peut aussi être choisi *optimal*, *i.e.* conduisant à une augmentation maximale de $V_{\bar{\theta}}^{R,T}(P^0)$. La méthode des *conditions de Wolfe* peut être vue comme une méthode intermédiaire de choix du pas qui fonctionne bien en pratique. Dans nos expériences numériques, nous avons comparé ces trois méthodes de recherche linéaire.

3.3.2 Algorithmes de montée par coordonnées (CA)

Dans le cas de variables d'action binaires, en utilisant le fait que $\forall j, \forall pa_\delta(a_j),\ \delta_j(a_j = 2|pa_\delta(a_j)) = 1 - \delta_j(a_j = 1|pa_\delta(a_j))$, nous pouvons construire directement un problème d'optimisation non contraint, dont le vecteur de paramètres est

$$\tilde{\delta} = \{\delta_j(a_j = 1|pa_\delta(a_j)), \forall j, \forall pa_\delta(a_j)\}.$$

Cela permet aussi de réduire le nombre de paramètres à optimiser à $N' = \frac{N}{2}$ et le problème d'optimisation devient:

$$\underset{\tilde{\delta}\in[0,1]^{\frac{N}{2}}}{\text{maximiser}} V_{\tilde{\delta}}^{R,T}(P^0) \tag{6}$$

L'algorithme de montée par coordonnées est un algorithme sans gradient. Il consiste à parcourir de manière cyclique les $\frac{N}{2}$ coordonnées $\delta_j(a_j = 1|pa_\delta(a_j))$ et essayer d'améliorer localement la valeur en

modifiant la coordonnée courante $\tilde{\delta}_k$. Les modifications peuvent consister à utiliser un *pas fixe* ou optimiser la valeur en modifiant uniquement la coordonnée courante (en utilisant l'algorithme *golden section search*, voir (Luenberger & Ye, 2008)). Si toutes les coordonnées sont restées inchangées après un cycle complet, un maximum local a été trouvé.

Les méthodes sans gradient sont généralement inapplicables à des problèmes avec plus de 1000 paramètres (Rios & Sahinidis, 2013) (nous dépassons largement ce nombre dans les problèmes que nous considérons). Cependant, l'algorithme de montée par coordonnées cyclique décrit ci-dessus a donné de bons résultats en pratique. Puisque les variables d'action sont binaires, les contraintes sont des contraintes 'de boîte' et l'algorithme de montée par coordonnées a la garantie de converger vers un maximum local de $V_{\tilde{\delta}}^{R,T}(P^0)$.

Si l'algorithme de montée de gradient permet de calculer les coordonnées du gradient en parallèle, l'algorithme de montée par coordonnées est forcément séquentiel.

4 Expériences numériques

Nous avons réalisé un certain nombre d'expériences numériques sur des problèmes de type PDMF-AF à horizon infini, ce qui est le cas le plus difficile pour notre algorithme. Dans toutes les expériences, nous avons utilisé un facteur d'amortissement de $\gamma = 0.9$.

Les différents algorithmes de résolution que nous avons testés empiriquement sont dénommés sous la forme *Optim-Eval*, où *Optim* est la méthode d'optimisation (GA ou CA) et *Eval* est la méthode d'évaluation, exacte (JT) ou approchée (LBP ou MC).

Notre code est en Matlab et utilise des appels à des fonctions de la librairie libDAI (Mooij, 2010) pour les algorithmes JT et LBP. Nous avons utilisé une implémentation Scilab de l'algortihme *Mean-Field Approximate Policy Iteration* (MF-API) pour les PDMG (Sabbadin *et al.*, 2012). Dans les PDMG, la transition, la récompense et la politique sont supposées avoir la même structure, et l'objectif est de trouver la meilleure politique déterministe factorisée. MF-API est basé sur une évaluation en champ moyen des politiques déterministes factorisées, *i.e.* une méthode 'plus approchée' que LBP. De plus, il n'y a pas de garantie avec cet algorithme d'améliorer la valeur de la politique à chaque itération. Les expériences ont été réalisées sur un serveur de 16 coeurs, et l'algorithme de montée de gradient a été parallélisé, en utilisant la toolobox Matlab de parallélisation.

Dans la Section 4.1, une comparaison des différents algorithmes d'évaluation est effectuée. Dans la Section 4.2, nous comparons les différents algorithmes d'itération de la politique approchée sur des petits problèmes ($n = 6$ variables d'état binaires, $m = 6$ variables d'action binaires, et $r = 6$ fonctions de récompense), pour lesquels une politique *globale* optimale peut être calculée. Nous avons utilisé la MDP toolobox[4] pour calculer les politiques globales optimales. Dans la Section 4.3, nous comparons les différents algorithmes d'itération de la politique approchée sur des problèmes de grande taille et de structure aléatoire (15 variables de chaque type, et des variables d'action binaires ou ternaires). Finalement, dans la Section 4.4, nous illustrons l'utilisation de nos algorithmes de type itération de la politique sur des problèmes d'épidémiologie agricole de grande taille (25 et 100 variables d'action binaires de chaque type). Nous n'avons trouvé aucune autre implémentation d'un algorithme dédié à la résolution de PDMF-AF permettant de traiter d'aussi grands problèmes. Dans la Section 4.2, nous avons utilisé une *structure de politique naturelle*, dans laquelle $pa_\delta(a_j)$ est l'union des variables d'état qui (i) soit apparaissent dans une même fonction de récompense R_α que a_j (ii) soit influencent une variable d'état conjointement avec a_j (*i. e.* apparaissent conjointement avec a_j dans un ensemble $pa_P(s_i^{t+1})$).

4.1 Méthodes d'évaluation des politiques

Dans le but de choisir une méthode d'évaluation approchée, la méthode d'évaluation des marginales est importante, mais aussi l'horizon d'approximation, $\hat{T}$. Les deux peuvent avoir un impact sur l'erreur d'évaluation et le temps de calcul. Nous avons testé l'impact de l'horizon d'approximation sur de petits PDMF[3], pour lesquels nous pouvons aller jusqu'à $\hat{T} = 100$ (Table 1).

L'erreur obtenue pour $\hat{T} = 20$ est significative (10%), mais du même ordre avec LBP qu'avec JT. MC donne de bons résultats pour $\hat{T} = 40$ (erreur de 1%), et toutes les méthodes sont assez précises pour $\hat{T} = 100$, mais trop lentes pour être incorporées dans un algorithme d'optimisation.

[4] `http://www.inra.fr/mia/T/MDPtoolbox`

Méthode d'évaluation	ERM	temps moyen (sec)
MC $\hat{T} = 40$	0.01	23.45
JT $\hat{T} = 20$	0.11	0.085
LBP $\hat{T} = 20$	0.11	0.049
MC $\hat{T} = 100$	6.8×10^{-4}	66.05
JT $\hat{T} = 100$	2.4×10^{-5}	1.1
LBP $\hat{T} = 100$	0.009	0.23

Table 1: Erreur relative moyenne et temps d'évaluation moyen de PSF sur 100 petits PDMF^3 aléatoires ($n = 6$ variables d'état binaires, $m = 6$ variables d'action binaires, $r = 6$ fonctions de récompense). Pour l'évaluation MC, la valeur moyenne est calculée sur 4000 trajectoires.

C'est pourquoi nous avons choisi d'utiliser l'algorithme LBP avec $\hat{T} = 20$ dans l'étape d'évaluation de nos algorithmes de type itération de la politique, puisqu'il offre un bon compromis entre temps de calcul et erreur d'approximation. L'approche MC avec $\hat{T} = 40$ (et 4000 trajectoires) sera utilisée pour comparer les valeurs des politiques retournées par ces algorithmes, quand une évaluation exacte est impossible.

4.2 PDMF^3 aléatoires de petite taille

Nous avons réalisé une série d'expériences sur de petits PDMF^3 pour lesquels la politique *globale* optimale peut être calculée, dans le but d'évaluer les politiques retournées par différentes instanciations *Optim-Eval* de notre algorithme (Table 2). La politique uniforme est utilisée pour initialiser les algorithmes. La valeur de η_q dans les algorithmes de montée par coordonnées est de 0.1, car les paramètres de la politique appartiennent à $[0, 1]$, tandis qu'elle est de 10 ou 20 dans les algorithmes de montée de gradient où les paramètres peuvent prendre n'importe quelle valeur réelle.

L'erreur relative moyenne (ERM) entre la politique optimale et les politiques approchées est donnée, ainsi que le temps de calcul moyen des politiques. Comme la politique optimale est globale, elle peut ne pas avoir la structure naturelle supposée dans nos algorithmes. L'ERM donnée est donc une borne supérieure de l'ERM par rapport à la politique factorisée optimale.

En utilisant les algorithmes CA-JT ou GA-JT, on obtient des politiques dont l'ERM par rapport à la politique globale optimale est proche de 2%. En utilisant les algorithmes CA-LBP ou GA-LBP avec différentes méthodes de recherche linéaire, on obtient une ERM de l'ordre de 12%. Nous avons aussi essayé l'algorithme GA-JT pour une structure de politique supposée générée aléatoirement, et de même taille que la structure naturelle. Nous avons alors obtenu une ERM de 0.048 (0.02 avec la structure naturelle) et un temps d'exécution moyen de 3.66 minutes (3.25 minutes avec la structure naturelle). Nous pouvons donc raisonnablement penser que, quand aucune structure de politique n'est donnée dans les contraintes du problème, la structure naturelle que nous avons proposée est un bon choix. Ici GA n'est pas parallélisé, sauf pour le cas de variables d'action ternaires où il commence à y avoir un gain en temps de calcul.

Plus généralement, plus un algorithme *Optim-Eval* donné est lent, meilleur il est en termes de qualité. La résolution exacte est plus rapide mais elle ne permet pas de trouver la meilleure politique factorisée avec la structure contrainte, tandis que CA-JT ou GA-JT fournissent une politique factorisée localement optimale présentant cette structure donnée.

4.3 PDMF^3 aléatoires de grande taille

Sur des PDMF^3 et des PDMG aléatoires, avec 15 variables de chaque type et 15 fonctions de récompense, la politique globale optimale ne peut plus être calculée. Nous avons comparé (i) CA-LBP, GA-LBP et MF-API sur des PDMG à variables binaires et (ii) CA-LBP, GA-LBP et une politique uniforme sur des PDMF^3 (Table 3). Pour les PDMF^3, nous nous sommes donné une structure de politique aléatoire, avec au plus 9 variables parent par variable d'action. Les algorithmes ont été initialisés avec la politique gloutonne dans le cas des PDMG, et avec la politique uniforme dans les autres cas. La politique gloutonne est la politique factorisée déterministe qui choisit les actions qui maximisent la récompense immédiate. Pour ces problèmes, la parallélisation de GA-LBP le rend beaucoup plus rapide que CA-LBP, au prix d'une légère perte de qualité.

Méthode - pas	ERM	temps moyen
Variables d'action binaires		
GA-JT - $\eta_q = 20$	0.0200	3.25 min
GA-LBP - $\eta_q = 20$	0.1228	1.88 min
GA-LBP - $\eta_q = 10$	0.1214	2.47 min
GA-LBP - pas de Wolfe	0.1216	2.48 min
GA-LBP - pas optimal	0.1217	2.63 min
CA-JT - $\eta_q = 0.1$	0.0222	5.14 min
CA-LBP - $\eta_q = 0.1$	0.1228	1.53 min
Exacte (politique globale)	-	1.95 min
Variables d'action ternaires		
GA-LBP - $\eta_q = 20$	0.1288	39.74s
Exacte (politique globale)	-	43.44s

Table 2: Erreur relative moyenne et temps de calcul moyen sur 100 petits PDMF^3 aléatoires ($n = 6$ variables d'état binaires, $m = 6$ variables d'action binaires, $r = 6$ fonctions de récompense). Les deux dernières lignes correspondent à 100 problèmes aléatoires avec $n = m = r = 5$ et des variables d'action ternaires.

Méthode - politique initiale - pas	Valeur (MC)	temps moyen
PDMG		
CA-LBP - gloutonne - $\eta_q = 0.1$	99.492	6.31 min
GA-LBP - gloutonne - pas de Wolfe	99.538	39.91 sec
CA-LBP - uniforme - $\eta_q = 0.1$	97.508	22.58 min
GA-LBP - uniforme - pas de Wolfe	99.472	1.25 min
MF-API - gloutonne	99.684	1.16 sec
PDMF^3, variables d'action binaires		
GA-LBP - uniforme - pas de Wolfe	85.65	9.18 min
CA-LBP - uniforme - $\eta_q = 0.1$	86.738	74.24 min
PDMF^3, variables d'action ternaires		
GA-LBP - uniforme - pas de Wolfe	76.47	14.97 min
CA-LBP	pas applicable	–
Uniform policy	65,542	–

Table 3: Evaluation Monte-Carlo des politiques solution obtenues pour de grand PDMF^3 et PDMG aléatoires (15 variables de chaque type). Les valeurs sont moyennées sur 5 problèmes aléatoires. GA est parallélisé (8 coeurs).

Sur les PDMG, nos algorithmes fournissent des PSF de bonne qualité si on compare à MF-API, mais ils sont plus lents que MF-API. Ces bons résultats en termes de qualité, tout comme ceux obtenus sur les problèmes de petite taille, nous encouragent à utiliser nos algorithmes sur des problèmes de grande taille n'appartenant pas à la sous-classe des PDMG.

4.4 Problèmes d'épidémiologie agricole

Nous considérons ici des PDMF^3 de très grande taille, inspirés par un problème d'épidémiologie agricole modélisé dans le cadre PDMG dans (Sabbadin *et al.*, 2012; Cheng *et al.*, 2013). Les parcelles sont distribuées sur une grille de taille 5×5 ou 10×10. Une variable d'état binaire (culture infectée ou saine) et une variable d'action binaire (système de culture normal ou parcelle traitée et laissée en jachère) sont attachées à chaque parcelle. La maladie peut se propager aux parcelles voisines uniquement. Dans le cas de la grille 10×10, le nombre total de paramètres décrivant une PSF est $N = 5184$.

Pour rendre le modèle PDMG initial plus réaliste, nous considérons que le traitement a lieu avant la propagation de la maladie et réduit la probabilité de contamination de parcelles voisines. Dans ce cas, le problème n'est plus un PDMG. La prbabilité d'infection devient:

$$P(\epsilon, p_1, n_1, p_2, n_2) = \epsilon + (1-\epsilon)(1-(1-p_1)^{n_1}(1-p_2)^{n_2}).$$

p_1 (resp. p_2) est la probabilité de contamination courte distance sans (resp. avec) traitement, n_1 (resp. n_2) est le nombre de parcelles voisines infectées et non traitées (resp. traitées), et ε est la probabilité de

Méthode - pas	Valeur (MC)	temps moyen
grille 5×5, départ uniforme		
Uniforme	10367	-
Gloutonne	13191	-
CA-LBP - $\eta_q = 0.1$	17264	1h16
GA-LBP - pas de Wolfe	13195	21min20
grille 10×10, départ uniforme		
Uniforme	40495	-
Gloutonne	52575	-
CA-LBP - $\eta_q = 0.1$	65068	43h42
GA-LBP - pas de Wolfe	52574	10h20

Table 4: Evaluation Monte-Carlo des politiques dans le problème d'épidémiologie agricole.

Méthode - pas - $\hat{T}$	Valeur (MC)	temps moyen
grille 5×5, $p = 0.2$, départ uniforme		
Uniforme	11393	-
Gloutonne	13953	-
Politique 1	18911	-
Politique 2	18349	-
CA-LBP - $\eta_q = 0.1$ - $\hat{T} = 20$	15055	47.25 min
GA-LBP - pas de Wolfe - $\hat{T} = 20$	14427	43.18 sec
GA-LBP - pas de Wolfe - $\hat{T} = 40$	14480	3.8 min

Table 5: Evaluation Monte-Carlo des politiques dans le problème d'épidémiologie agricole avec réseau de surveillance.

contamination longue distance. La forme de la fonction de récompense est la même que dans (Sabbadin *et al.*, 2012), avec une récompense maximale de $\rho = 100$.

La Table 4 donne les résultats obtenus pour $\epsilon = 0.01$, $p_1 = 0.6$, $p_2 = 0.4$ et $\hat{T} = 20$ (dans l'évaluation approché par LBP). En prenant $\hat{T} = 40$ nous avons obtenu des résultats similaires. Nous avons comparé les politiques issues de CA-LBP et GA-LBP avec la politique stochastique uniforme et la politique gloutonne. CA-LBP améliore la valeur de la politique gloutonne de 30% dans le cas de la grille 5×5, et de 23% dans le cas de la grille 10×10. GA-LBP converge quant à lui vers la politique gloutonne.

Enfin, nous avons considéré la situation d'un réseau de surveillance dans un problème d'épidémiologie agricole. Dans ce type de problèmes, les politiques expertes donnent de bons résultats. La transition et la récompense sont les mêmes que dans Sabbadin *et al.* (2012), mais les décisions des agriculteurs à l'échelle de la parcelle se prennent en fonction de la même information: 4 parcelles (la parcelle de coordonnées (2,2) et les trois parcelles symétriques) sont surveillées et leur état (sain ou infecté) au temps courant est connu. MF-API ne permet pas de résoudre ce problème, qui n'est pas un PDMG. Les politiques issues de CA-LBP et GA-LBP sont comparées à la politique uniforme, à la politique gloutonne et à des politiques expertes, consistant à traiter une parcelle (i) si une des parcelles monitorées parmi les plus proches est infectée (politique 1) ou (ii) si au moins trois des parcelles monitorées sont infectées (politique 2).

Les résultats montrent que CA-LBP et GA-LBP améliorent la politique uniforme ou gloutonne, GA-LBP étant plus rapide mais de qualité légèrement moins bonne que CA-LBP (Table 5). Cependant, les politiques expertes sont de meilleure valeur. Dans cet exemple, nous avons pu vérifier que les politiques expertes étaient des maxima locaux de $V_\delta^{R,T,LBP}(P^0)$ (ou des points critiques). Dans le cas général, quand des politiques expertes factorisées sont disponibles, initialiser CA-LBP ou GA-LBP avec elles est une bonne manière de les évaluer et de déterminer si elles peuvent être améliorées ou si elles sont des maxima locaux.

5 Conclusion

Nous avons proposé un cadre générique, appelé PDMF^3, et une famille d'algorithmes de résolution approchée pour des PDMF-AF de grande taille, basés sur le calcul de politiques stochastiques factorisées. Dans ce cadre, nous pouvons traiter des problèmes ayant jusqu'à 2^{100} états et actions.

Les expériences numériques ont montré que ces algorithmes retournent des politiques de valeur équivalente

ou meilleure que celles retournées par les algorithmes existants, quand ceux-ci sont disponibles. Quand les problèmes sont de trop grande taille pour être résolus avec les approches existantes, les politiques retournées par nos algorithmes améliorent des politiques arbitraires simples. Quand une connaissance experte est disponible à propos des politiques solutions, nos algorithmes peuvent utiliser ces politiques expertes comme initialisation, et soit les améliorer, soit montrer qu'elles ne peuvent être améliorées localement. Les politiques expertes peuvent également suggérer une structure de politique.

Une utilisation raisonnable de CA-LBP et GA-LBP est donc de les utiliser dans une boucle itérative Politique Experte - Optimisation - Interprétation Experte. La flexibilité de notre approche nous permet aussi d'utiliser successivement les différents algorithmes: (i) GA-LBP pour arriver rapidement à un optimum local puis (ii) CA-LBP (si les variables d'action sont binaires) pour éventuellement améliorer la valeur courante et (iii) CA-MC ou GA-MC pour optimiser une estimation asymptotiquement non biaisée de la valeur.

Le cadre que nous proposons ici admet certaines extensions naturelles. Par exemple, l'observabilité partielle est déjà traitée en partie, puisque les PSF n'ont accès qu'à une partie des variables d'état (voir la dernière expérience numérique par exemple). Il peut être étendu facilement au cas des Dec-POMDP à espace d'état factorisé, en considérant des observations pour lesquelles la probabilité d'observation est factorisée. Le principe de l'approche présentée ici serait inchangé, à condition que le graphe des dépendances formé par la transition, la PSF et la probabilité d'observation soit acyclique. Une extension plus difficile de ce travail serait de déterminer automatiquement une structure de politique satisfaisante. Déterminer la structure qui admet une PSF de valeur maximale revient à optimiser à la fois des variables discrètes (représentant la structure de la politique, avec des contraintes de parcimonie) et des variables continues (les paramètres de la PSF).

References

BERNSTEIN D., GIVAN R., IMMERMAN N. & ZILBERSTEIN S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, **27**(4), 819–840.

BUFFET O. & ABERDEEN D. (2009). The factored policy-gradient planner. *Artificial Intelligence*, **173**, 722–747.

CHENG Q., LIU Q., CHEN F. & IHLER A. (2013). Variational planning for graph-based MDPs. In *Advances in Neural Information Processing Systems 26*, p. 2976–2984.

DIBANGOYE J. S., AMATO C., BUFFET O. & CHARPILLET F. (2014). Exploiting separability in multiagent planning with continuous-state MDPs. In *Proceedings of the 13th international conference on Autonomous Agents and Multiagent Systems*.

DIBANGOYE J. S., AMATO C. & DONIEC A. (2012). Scaling up decentralized MDPs through heuristic search. In *Proceedings of the 28th Conference on Uncertainty and Artificial Intelligence*, p. 217–226.

FORSELL N. & SABBADIN R. (2006). Approximate linear-programming algorithms for graph-based markov decision processes. In *Proceedings of the 17th European Conference on Artificial Intelligence*, p. 590–594.

FREY B. & MACKAY D. (1998). A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, p. 479–485.

GUESTRIN C., KOLLER D. & PARR R. (2001). Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems*, p. 1523–1530.

HOEY J., ST-AUBIN R., HU A. & BOUTILIER C. (1999). SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, p. 279–288.

KIM K.-E. & DEAN T. (2002). Solving factored MDPs with large action space using algebraic decision diagrams. In *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence*, p. 80–89.

KUMAR A., ZILBERSTEIN S. & TOUSSAINT M. (2011). Scalable multiagent planning using probabilistic inference. In *Proceedings of the 22th International Joint Conference on Artificial intelligence*.

LITTMAN M., GOLDSMITH J. & MUNDHENK M. (1998). The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, **9**, 1–36.

LUENBERGER D. G. & YE Y. (2008). *Linear and Nonlinear Programming*. Springer, 3rd edition.

MOOIJ J. M. (2010). libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, **11**, 2169–2173.

MURPHY K. (2002). *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, School of Computer Science, University of California, Berkeley.

OLIEHOEK F. A., WHITESON S. & SPAAN M. T. J. (2013). Approximate solutions for factored Dec-POMDPs with many agents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*.

PEYRARD N. & SABBADIN R. (2006). Mean field approximation of the policy iteration algorithm for graph-based markov decision processes. In *ECAI*, p. 595–599.

PUTERMAN M. (1994). *Markov decision processes*. John Wiley and Sons.

RAGHAVAN A., JOSHI S., FERN A., TADEPALLI P. & KHARDON R. (2012). Planning in factored action spaces with symbolic dynamic programming. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

RIOS L. M. & SAHINIDIS N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, **56**, 1247–1293.

SABBADIN R., PEYRARD N. & FORSELL N. (2012). A framework and a mean-field algorithm for the local control of spatial processes. *International Journal of Approximate Reasoning*, **53**(1), 66–86.

SALLANS B. & HINTON G. E. (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, **5**, 1063–1088.

ST-AUBIN R., HOEY J. & BOUTILIER C. (2000). APRICODD: Approximate policy construction using decision diagrams. In *Advances in Neural Information Processing Systems*, p. 1089–1095.

YEDIDIA J. S., FREEMAN W. T. & WEISS Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, **51**(7), 2282–2312.