Coordination distribuée et hors-ligne de planifications locales

Guillaume Desquesnes, Guillaume Lozenguez, Arnaud Doniec, Éric Duviella

IMT Lille Douai, Univ. Lille, URIA, F-59000 Lille, France
prénom.nom@mines-douai.fr

Résumé : Une modélisation centralisée n'est pas toujours une solution viable pour les problèmes de planification, à cause de la taille exponentielle de la représentation ou bien à cause de contraintes de vie privée entre les composants du système. Dans ce papier, un modèle distribué de système basé entités est utilisé pour résoudre ces problèmes. Le but est d'optimiser chaque entité en utilisant des actions pouvant affecter d'autres entités. N'ayant qu'une connaissance d'un sous-ensemble du système, les entités affectées par ses actions, un agent doit se coordonner avec d'autres agents pour atteindre une solution localement optimale du contrôle du système. Avec la coordination hors-ligne de planifications locales, chaque agent mettra à jour son propre plan en prenant en compte ceux des autres agents, en utilisant les principes de coévolution et de détection de la terminaison introduits par le Distributed Breakout Algorithm. Une approche générique est introduite pour coordonner la planification hors-ligne de tels problèmes. Cela résulte en chaque agent ayant sa propre politique, ne nécessitant pas de communiquer durant l'exécution. Des résultats expérimentaux sur un réseau de voies navigables utilisant cette approche donnent de bons résultats et montrent une capacité à passer à l'échelle.

Mots-clés : Processus décisionnels markoviens, Coordination et coopération, Planification multi-agent/distribuée

1 Introduction

Dans cet article, nous portons notre attention sur la planification d'un système observé et géré par de multiple agents, incapable d'avoir une vue globale du système pour des raisons, par exemple, de privacité ou de complexité. Chaque agent aura une vision locale, avec observabilité complète, limitée à sa partie du problème, avec des informations partagées avec d'autres agents. Nous nous plaçons dans le cas où les agents ne communiquent pas durant leurs exécutions.

L'approche proposée dans cet article s'intéresse en partie à résoudre des problèmes de gestion de l'eau dans des systèmes de voies navigable. Par ailleurs, en Europe, le réseau de voies navigables est réparti sur plusieurs pays, chaque pays gérant son propre sous-réseau. Ces sous-réseaux sont faiblement connectés. Un certain nombre de biefs, sur la frontière franco-belge, sont affectés par les déplacements d'eau des deux pays donc ces biefs devront être observés par les gestionnaires des deux pays. Dans chaque pays, l'obtention d'une planification sera dépendante des actions de ses voisins. Il sera donc nécessaire de se coordonner a priori pour atteindre des buts communs sans communication durant l'exécution.

Cet article introduit des mécanismes de coordination hors-ligne pour des systèmes basés sur les entités divisés en multiples agents, inspirés par l'algorithme LID-JESP introduit pour les ND-POMDP (Nair *et al.*, 2005) et le Distributed Breakout Algorithm (Yokoo & Hirayama, 1996). Il utilise une modélisation basée sur les processus de décision markovien (MDP (Puterman, 1994), Factored-MDP (Boutilier *et al.*, 1995), ...) pour la modélisation et planification locale. L'utilisation de mécanismes de communications décentralisés et de méthodes d'évolutions liées au problème permettra à la politique jointe de converger vers une solution localement optimale. Comme chaque agent possède son propre MDP, il obtiendra une politique qui ne nécessitera pas de communication lors de son utilisation.

Ce papier est organisé de la façon suivante : tout d'abord, nous introduisons notre problématique et rappelons les bases des processus de décision markovien (MDP) et du Distributed Breakout Algorithm. Puis, nous introduisons formellement le problème et présentons l'algorithme pour la coordination de plusieurs agents. Nous montrons qu'il est possible de prouver la convergence de l'algorithme. Après coup, un exemple illustrera l'algorithme proposé. Nous terminons en montrant des résultats expérimentaux de l'application de l'algorithme pour optimiser la gestion de l'eau dans de grands réseaux de voies navigables.

2 Présentation du problème

De nombreuses applications de taille importante sont naturellement distribuées et donc leur système peut être divisé en un ensemble (noté ω) de sous-systèmes semi-indépendant, appelé entités dans cet article. Une entité est définie comme un élément du système possédant un état s_e qui appartient à un ensemble fini d'états S_e. Ici, l'évolution de l'état d'une entité ne dépend pas directement de l'état des autres entités.

Chaque entité e est affectée par un ou plusieurs points d'action du système $\{A_{e_1}, \ldots, A_{e_k}\}$. Similairement, un point d'action A_i affectera une ou plusieurs entités. Les actions dans ces systèmes distribués ont des impacts locaux. Cependant, ces impacts locaux peuvent être chaînés empêchant donc de facilement distribuer le processus de planification. L'évolution de l'état de l'entité e ne dépend donc que de sa valeur courante s_e et de l'action jointe choisie (tuple d'actions réalisées simultanément sur chaque point d'action). Ainsi, il est possible d'exprimer la probabilité d'atteindre un état s' :

$$p(s'_e|s_e, a_{e_1}, \ldots, a_{e_k}) \tag{1}$$

Toute entité e possède un ensemble d'états optimaux ainsi qu'une fonction de distance δ_e donnant l'écart à l'état optimal le plus proche, voir équation 2. Le but est de choisir une politique jointe minimisant la distance de chaque entité relativement à ses états optimaux. À horizon 1, le but est de trouver pour chaque état s du système l'action jointe a minimisant l'équation 3.

$$\delta_e(s_e) = |s^*_e - s_e| \tag{2}$$

$$\sum_{e \in \omega} \sum_{s'_e \in S_e} \Big(p(s'_e|s_e, a_{e_1}, \ldots, a_{e_k}) \times \delta_e(s'_e) \Big) \tag{3}$$

À cause du nombre de combinaisons sur les actions jointes, les systèmes distribués et chaînés sont difficiles à contrôler optimalement. Le problème est d'autant plus complexe en considérant les incertitudes (l'état résultant d'une action dépend de probabilités).

2.1 Modélisation à horizon infini

Le contrôle optimal d'un système complexe à entités (ω) se ramène à la résolution d'un processus décisionnel de Markov (MDP) (Puterman, 1994) (Markov Decision Process - MDP). Un MDP est un framework générique modélisant les possibilités de contrôle d'un système stochastique dynamique sous forme d'un automate probabiliste. Il est défini par un tuple $\langle S, A, T, C \rangle$ avec S et A respectivement l'ensemble d'états et d'actions qui définissent le système et ses capacités de contrôle. T est la fonction de transition définie par $T : S \times A \times S \to [0, 1]$. $T(s, a, s')$ est la probabilité d'atteindre l'état s' après avoir fait l'action a dans l'état s. La fonction de coût C est définie par $C : S \times A \times S \to \mathbb{R}$. $C(s, a, s')$ donne le coût pour avoir atteint l'état s' en faisant l'action a depuis l'état s.

Ici, l'ensemble des états du système S_ω correspond au produit cartésien des états locaux de chaque entité et les actions A_ω au produit cartésien des actions pour les points d'actions. Les fonctions de transition et de coût s'appuient sur les probabilités de transition et l'état optimal pour chaque entité.

$$T_\omega(s, a, s') = \prod_{e \in \omega} p(s'_e|s_e, a_{e_1}, \ldots, a_{e_k}) \qquad C_\omega(s, a, s') = C_\omega(s') = \sum_{e \in \omega} \delta_e(s'_e) \tag{4}$$

Résoudre un MDP consiste à construire une politique d'actions optimales. Une politique est une fonction $\pi : S_\omega \to A_\omega$ qui assigne une action à chaque état du système. Une politique optimale π^* est une politique qui minimise le coût espéré, en minimisant l'équation de Bellman (Bellman, 1957) :

$$V^\pi_\omega(s) = \sum_{s' \in S} T_\omega(s, \pi(s), s') \times \big(C_\omega(s, \pi(s), s') + \gamma \times V^\pi_\omega(s') \big) \tag{5}$$

$$\pi^*(s) = \underset{a \in A}{\arg\min} \left(\sum_{s' \in S} T_\omega(s, a, s') \times \Big(C_\omega(s, a, s') + \gamma \times V^{\pi^*}_\omega(s') \Big) \right) \tag{6}$$

2.2 Une approche multi-agent

En utilisant la modélisation basée agent, un agent sera responsable d'un sous-ensemble de points d'action du système modélisé. Il observera uniquement les entités affectées par ses actions possibles pour prendre une décision. Sa politique sera calculée pour être coordonnée avec les politiques de ces voisins. Ce framework promet de résoudre heuristiquement des problèmes intraitables en utilisant des algorithmes distribués, basés sur des processus décisionnels de Markov pour la planification locale.

Le problème de coordination de plan n'est pas nouveau. Le Distributed Breakout Algorithm (Yokoo & Hirayama, 1996) (DBA) est un algorithme pour résoudre des problèmes de satisfaction de contraintes (CSP) distribués. Pour le DBA, chaque agent essaye d'optimiser son évaluation (le nombre de contraintes violées) en échangeant son évaluation actuelle et son amélioration par rapport à l'évaluation précédente à son voisinage. Deux agents voisins n'ont pas le droit de se mettre à jour en même temps afin d'éviter les changements conflictuels. Un mécanisme de communication est utilisé pour détecter la convergence globale. Cette solution a été adaptée aux Network Distributed Partially Observable MDP (Nair *et al.*, 2005) qui se basent sur des transitions ou observations indépendantes entre les agents pour résoudre des problèmes avec un nombre d'agents important. Un algorithme similaire (Chades *et al.*, 2002) a aussi été proposé pour faciliter la résolution de Dec-POMDP avec une forte contrainte d'un ensemble d'états et d'une fonction de transition globaux.

Pour des systèmes complexes à base d'entité, la somme totale des distances aux états optimaux des entités est distribuable. Les agents ont pour but de minimiser la somme des fonctions de distance de chaque entité observée vers leurs valeurs optimales. L'agent α aura à minimiser sa fonction objective locale V_α, basée sur la somme des distances des entités gérées par α.

La difficulté est d'estimer l'état d'une entité dont l'agent n'a qu'un contrôle partiel des actions qui l'affectent. Deux agents observant une même entité sont définis comme étant voisins. Cependant, les agents n'ont pas forcément une connaissance complète de l'évolution des entités partagées avec leurs voisins. En effet, les politiques exactes des voisins ne sont pas exprimables sur la base des seules entités observées par l'agent. En minimisant chaque fonction objective locale, le but est d'atteindre un optimum local de la fonction objective globale.

3 Présentation de l'algorithme local

L'algorithme local proposé, appelé OCLP (Offline Coordination of Local Planning - Coordination hors-ligne de planifications locales), a pour but de trouver une politique jointe localement optimale de manière distribuée. Il est initialement inspiré par l'algorithme LID-JESP (Nair *et al.*, 2005) pour les ND-POMDP et du DBA (Yokoo & Hirayama, 1996) pour optimiser la gestion de grands réseaux de voies navigables en utilisant des MDP discrétisés. Cet algorithme se base sur des agents avec une vision locale qui peuvent être affectés par le choix d'autres agents.

Algorithme Coordination hors-ligne de planifications locales pour un agent α

1: Créer une politique local initiale $\pi_{\alpha_0} : S_\alpha \rightarrow A_\alpha$
2: $d \leftarrow$ distance maximale entre deux agents du système
3: $it \leftarrow 0$
4: **répéter :**
5: Adapter et échanger $\pi_{\alpha_{it}}$ avec chaque voisin (voir équation 7)
6: Mettre à jour MDP$_{\alpha_{it}} =< S_\alpha, A_\alpha, T_{\alpha_{it}}, R_\alpha >$ en MDP$_{\alpha_{it+1}}$ grâce aux politiques reçues
7: Construire $\pi'_{\alpha_{it}}$ la politique optimale de MDP$_{\alpha_{it+1}}$
8: $g_{\alpha_{it}} \leftarrow gain(\pi'_{\alpha_{it}}, \pi_{\alpha_{it}})$ de MDP$_{\alpha_{it+1}}$
9: Échanger $g_{\alpha_{it}}$ avec chaque voisin ; $G_{\alpha_{it}} \leftarrow$ gains du voisinage
10: $\pi_{\alpha_{it+1}} \leftarrow \pi'_{\alpha_{it}}$ si $g_{\alpha_{it}} = \max \text{disponible}(G_{\alpha_{it}})$ et $\pi_{\alpha_{it}}$ sinon
11: $counter \leftarrow d$ si $g_{\alpha_{it}} > 0$ sinon $counter - 1$
12: Échanger $counter$ avec chaque voisin ; $Counters_{it} \leftarrow$ compteurs du voisinage
13: $counter \leftarrow \max(Counters_{it})$
14: $it \leftarrow it + 1$
15: **jusqu'à :** $counter = 0$

À chaque itération it, les agents échangeront des adaptations de leur politique actuelle avec chacun de leurs voisins, adaptées à la connaissance commune des deux agents (ligne 5). Une politique de β adaptée à α, voir l'équation 7, correspond à une assignation d'un ensemble de couple action - probabilité à chaque état partagé par α et β.

$$\pi_{\alpha}^{\beta}(s_{\alpha\beta}) = \{(\pi_{\beta}(s_{\beta}), p(s_{\beta}|s_{\alpha\beta})),\ \forall s_{\beta} \in S_{\beta}\} \tag{7}$$

où $s_{\alpha\beta}$ est une assignation d'états aux entités supervisées à la fois par α et β.

Lorsqu'un agent obtient toutes les estimations de politique de son voisinage, il sera capable de changer son modèle MDP_{it}. Ainsi, il prendra en compte le nouvel impact du voisinage sur ses entités partagées via la mise à jour de sa fonction de transition de façon à refléter les actions probables du voisinage (ligne 6).

Toute itération pourra mettre à jour les parties du modèle affectant les entités partagées de chaque agent. L'utilisation de ce nouveau modèle, MDP_{it+1}, permettra aux agents d'obtenir une nouvelle politique optimale π'_{it} (ligne 7). L'amélioration de la politique optimale (π'_{it}) par rapport à la politique actuelle (π_{it}) dans le nouveau modèle (MDP_{it+1}) est évaluée grâce à une fonction heuristique (ligne 8). Cette fonction heuristique a pour but de guider l'exploration de la résolution et de détecter la convergence. Seul l'agent avec la plus grande amélioration par rapport à son voisinage gardera sa nouvelle politique, tandis que les autres agents l'ignoreront (ligne 10). Afin d'éviter de n'avoir qu'un seul agent gardant sa politique à chaque itération, la valeur d'amélioration d'agents bloqués par d'autres voisinages est ignorée. Ne garder qu'une seule nouvelle politique par voisinage permet d'éviter l'adoption des changements conflictuels lors d'une même itération.

Proposition
L'algorithme terminera, au plus en $d = \max_{\alpha,\beta \in Ag^2} pathSize(\alpha, \beta)$ *itérations si et seulement si tous les agents sont dans un optimum local.* $pathSize$ *est défini comme :*

$$pathSize(\alpha, \beta) = \begin{cases} 1 & \text{si } \beta \in N(\alpha) \\ 1 + \min_{\gamma \in N(\alpha)} pathSize(\gamma, \beta) & sinon \end{cases} \tag{8}$$

avec $N(\alpha)$ *l'ensemble des agents voisins de* α *et* Ag *l'ensemble des agents.*

Pour garantir que tous les agents s'arrêtent de calculer en même temps dès qu'ils atteignent un optimum local, un compteur est utilisé. Chaque agent initialisera son compteur à la même valeur : $d > 0$. La valeur de d est assignée une fois pour toutes au début de l'algorithme par un agent externe et correspond à la taille du chemin maximal entre deux agents du système. À la fin d'une itération, si une nouvelle politique, différente de la politique actuelle est disponible, alors le compteur de l'agent sera réinitialisé à d. Autrement, la compteur sera diminué de 1 (ligne 11). Ensuite, les agents échangeront leur compteur avec leurs voisins et ne garderont que le plus grand (ligne 13). Un agent aura terminé sa résolution lorsque son compteur atteindra 0 (ligne 15).

Pour cet algorithme, les communications sont synchronisées puisque la continuation de l'algorithme après l'échange des politiques, compteurs et gains dépend des réponses du voisinage.

4 Preuve

La coordination hors-ligne de planifications locales possède certaines garanties notamment sur la convergence vers une solution localement optimale ainsi que sur la terminaison de l'algorithme après avoir fait $\max_{\alpha,\beta \in Ag^2} pathSize(\alpha, \beta)$ itérations une fois qu'une solution localement optimale est atteinte.

4.1 Preuve de terminaison

Supposons que l'agent α ne commence pas l'itération it car il a terminé sa résolution à l'itération $i - 1$ ($counter_{\alpha}(it - 1) = 0$), mais que d'autres agents ne sont pas dans un optimum local. Cela implique qu'à l'itération $i - d$, il existe au moins un agent β qui peut améliorer sa politique et ainsi réinitialiser son compteur ($counter_{\beta}(it - d) = d$). Les compteurs étant décrémentés d'au plus 1 à chaque itération et étant propagés à travers le voisinage, alors à l'itération $it - d + pathSize(\alpha, \beta)$ la borne minimale du compteur de l'agent α peut être définie par $it - d + pathSize(\alpha, \beta)$. De ce fait, $counter_{\alpha}(it - 1) \geq$

$d - pathSize(\alpha, \beta) + 1 - d + pathSize(\alpha, \beta) = 1$. Donc l'agent α aura besoin d'effectuer l'itération i. Par contradiction, si l'algorithme termine pour un agent, alors tous les agents sont dans un optimum local.

À l'inverse, si tous les agents ont atteint un optimum local, les compteurs ne seront jamais réinitialisés à d et diminueront de 1 à chaque itération. Donc après d itérations, $\forall \alpha, counter_\alpha = 0$ et les agents s'arrêtent.

4.2 Preuve de convergence

Pour prouver la convergence de l'algorithme, il va être montré par récurrence que la somme des coûts espérés pour chaque état du système diminue à chaque itération :

$$\sum_{s \in S_\omega} V_\omega^{\pi_\omega^{it}}(s) \geq \sum_{s \in S_\omega} V_\omega^{\pi_\omega^{it+1}}(s) \tag{9}$$

4.2.1 Cas mono-agent

Si le système est modélisé par un seul agent, alors $\alpha = \omega$ et le MDP local de l'agent α est complet donc par définition :

$$\sum_{s \in S_\omega} V_\omega^{\pi_\omega^{it}}(s) \geq \sum_{s \in S_\omega} V_\omega^{\pi_\omega^{it+1}}(s) \tag{10}$$

En l'occurrence, l'agent trouvera sa politique optimale globale lors de la première et unique itération de l'OCLP puisque la coordination n'est pas requise.

4.2.2 Cas multi-agent

En supposant qu'un ensemble de n agents soit garantie de converger, il est possible de prouver que l'ajout d'un agent supplémentaire, avec un certain ensemble d'entités à contrôler, gardera cette garantie. Dans la preuve suivante, le $(n+1)^{\text{ème}}$ agent est dénoté α tandis que l'ensemble de n agents est représenté par un méta-agent β. ω correspond donc à l'ensemble des entités supervisées par α et β.

Si α met à jour sa politique

Dans le cas ou α met à jour sa politique, lorsque son gain est le maximum de son voisinage (ligne 10 de l'OCLP), à l'itération it ($\pi_\alpha^{it+1} = \pi_\alpha'^{it}$), alors comme aucun agent voisin avec α ne changera de politique durant cette itération :

$$\sum_{s \in S_\alpha} V_\alpha^{\pi_\alpha^{it}}(s) \geq \sum_{s \in S_\alpha} V_\alpha^{\pi_\alpha^{it+1}}(s) \tag{11}$$

Le méta-agent β possède des agents qui seront influencés par le changement de politique de α, il est donc impossible d'établir une relation entre la somme des coûts espérés aux itérations it et $it+1$. Néanmoins, grâce à l'hypothèse de convergence du méta-agent β, il est possible d'affirmer que le nouvel ensemble de politiques du méta-agent diminue la somme des coûts espérés à l'itération it :

$$\sum_{s \in S_\beta} V_\beta^{\pi_\beta^{it}}(s) \geq \sum_{s \in S_\beta} V_\beta'^{\pi_\beta^{it}}(s) \tag{12}$$

L'évolution d'une entité ne dépendant que de son état et du choix d'action, cela rend la fonction de valeur décomposable, d'où :

$$V_\mu^{\pi_\mu^{it}}(s) = \sum_{e \in \mu} V_e^{\pi_\mu^{it}}(s) \qquad \forall \mu \in Ag,\ \forall s \in S_\mu \tag{13}$$

Soit $\alpha \cap \beta$ l'ensemble des entités partagées par α et β. Puisque l'agent α garde sa politique à l'itération it alors tout agent du méta-agent β supervisant au moins une des entités de $\alpha \cap \beta$ ne changera pas de plan durant cette itération. Ceci implique :

$$\sum_{e\in\alpha\cap\beta} V_e^{\pi'_{it}}(s) = \sum_{e\in\alpha\cap\beta} V_e^{\pi_{it}}(s) \quad \text{et} \quad \sum_{e\in\beta\setminus\alpha} V_e^{\pi'_{it}}(s) = \sum_{e\in\beta\setminus\alpha} V_e^{\pi_{it+1}}(s) \quad ,\forall s \in S_\beta \tag{14}$$

d'où

$$V_\beta^{\prime\pi_\beta^{it}}(s) = \sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it+1}} + \sum_{e\in\alpha\cap\beta} V_e^{\pi_\beta^{it}} \quad , \forall s \in S_\beta \tag{15}$$

donc en combinant l'équation 15 à l'équation 12 :

$$\sum_{s\in S_\beta} V_\beta^{\pi_\beta^{it}}(s) \geq \sum_{s\in S_\beta} V_\beta^{\prime\pi_\beta^{it}}(s) \tag{16}$$

$$\sum_{s\in S_\beta}\left(\sum_{e\in\alpha\cap\beta} V_e^{\pi_\beta^{it}}(s) + \sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it}}(s)\right) \geq \sum_{s\in S_\beta}\left(\sum_{e\in\alpha\cap\beta} V_e^{\pi_\beta^{it}}(s) + \sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it+1}}(s)\right) \tag{17}$$

$$\sum_{s\in S_\beta}\left(\sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it}}(s)\right) \geq \sum_{s\in S_\beta}\left(\sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it+1}}(s)\right) \tag{18}$$

L'addition des équations 11 et 12 peut alors se simplifier en :

$$\sum_{s\in S_\alpha} V_\alpha^{\pi_\alpha^{it}}(s) + \sum_{s\in S_\beta} V_\beta^{\pi_\beta^{it}}(s) \geq \sum_{s\in S_\alpha} V_\alpha^{\pi_\alpha^{it+1}}(s) + \sum_{s\in S_\beta} V_\beta^{\prime\pi_\beta^{it}}(s) \tag{19}$$

$$\sum_{s\in S_\alpha}\left(\sum_{e\in\alpha} V_e^{\pi_\alpha^{it}}(s)\right) + \sum_{s\in S_\beta}\left(\sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it}}(s)\right) \geq \sum_{s\in S_\alpha}\left(\sum_{e\in\alpha} V_e^{\pi_\alpha^{it+1}}(s)\right) + \sum_{s\in S_\beta}\left(\sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it+1}}(s)\right) \tag{20}$$

et donc, grâce à l'indépendance entre les entités :

$$\sum_{s\in S_\omega}\left(\sum_{e\in\alpha} V_e^{\pi_\alpha^{it}}(s) + \sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it}}(s)\right) \geq \sum_{s\in S_\omega}\left(\sum_{e\in\alpha} V_e^{\pi_\alpha^{it+1}}(s) + \sum_{e\in\beta\setminus\alpha} V_e^{\pi_\beta^{it+1}}(s)\right) \tag{21}$$

$$\sum_{s\in S_\omega}\left(\sum_{e\in\omega} V_e^{\pi_\omega^{it}}(s)\right) \geq \sum_{s\in S_\omega}\left(\sum_{e\in\omega} V_e^{\pi_\omega^{it+1}}(s)\right) \tag{22}$$

$$\sum_{s\in S_\omega} V_\omega^{\pi_\omega^{it}}(s) \geq \sum_{s\in S_\omega} V_\omega^{\pi_\omega^{it+1}}(s) \tag{23}$$

□

Si α ne met pas à jour sa politique

Lorsque l'agent α ne change pas sa politique lors de l'itération it, alors par définition :

$$\sum_{s\in S_\beta} V_\beta^{\pi_\beta^{it}}(s) \geq \sum_{s\in S_\beta} V_\beta^{\pi_\beta^{it+1}}(s) \qquad \text{et} \qquad \sum_{s\in S_\alpha} V_\alpha^{\pi_\alpha^{it}}(s) = \sum_{s\in S_\alpha} V_\alpha^{\prime\pi_\alpha^{it}}(s) \tag{24}$$

d'où

$$V_e^{\pi_\alpha^{it}}(s) = V_e^{\pi_\alpha^{it+1}}(s) \qquad \forall e \in \alpha \setminus \beta,\ \forall s \in S_\alpha \tag{25}$$

donc en suivant un raisonnement similaire à α changeant de politique :

$$\sum_{s\in S_\alpha}\left(\sum_{e\in\alpha\setminus\beta} V_e^{\pi_\alpha^{it}}(s)\right)+\sum_{s\in S_\beta}\left(\sum_{e\in\beta} V_e^{\pi_\beta^{it}}(s)\right)\geq\sum_{s\in S_\alpha}\left(\sum_{e\in\alpha\setminus\beta} V_e^{\pi_\alpha^{it+1}}(s)\right)+\sum_{s\in S_\beta}\left(\sum_{e\in\beta} V_e^{\pi_\beta^{it+1}}(s)\right) \tag{26}$$

$$\sum_{s\in S_\omega}\left(\sum_{e\in\omega} V_e^{\pi_\omega^{it}}(s)\right)\geq\sum_{s\in S_\omega}\left(\sum_{e\in\omega} V_e^{\pi_\omega^{it+1}}(s)\right) \tag{27}$$

$$\sum_{s\in S_\omega} V_\omega^{\pi_\omega^{it}}(s)\geq\sum_{s\in S_\omega} V_\omega^{\pi_\omega^{it+1}}(s) \tag{28}$$

□

Lorsque n agents arrivent à converger entre eux, l'ajout d'un agent supplémentaire maintiendra cette propriété que ce dernier change ou garde sa politique.

4.2.3 Conclusion de la preuve de convergence

Il a été prouvé qu'un agent seul convergera toujours vers l'optimum global. De plus ajouter un agent à un groupe d'agents pouvant converger vers un optimum local maintiendra cette propriété, alors par récurrence l'algorithme OCLP convergera toujours vers un optimum local.

5 Exemple d'utilisation de l'algorithme basée sur un CSP

Cet algorithme a été introduit pour résoudre des problèmes de partage de ressources entre plusieurs entités utilisant des actions distribuées. Un exemple illustratif est ici présenté, modélisant une répartition d'objets identiques entre plusieurs chariots, les entités du système, représentés par des carrés sur la figure 1. Chaque chariot a une évaluation différente de sa charge, voir équation 29, définissant donc une valeur optimale. Chaque chariot a un remplissage initial et une capacité maximale (indiqués dans chaque carré par « initial/maximale »). Il n'est possible de déplacer les objets que de chariots en chariots, uniquement dans le sens indiqué par les flèches et dans les capacités définies par les intervalles à proximité. Dans cet exemple, les actions sont considérées comme instantanées et simultanées.

Trois agents, α, β, γ vont gérer les transferts, supposés simultanés d'objets entre les différents chariots qu'ils contrôlent. L'agent α contrôle le transfert $\{o\}$, ce qui lui donne la vision des chariots $\{0, 1\}$. β et γ contrôlent respectivement $\{p, q\}$ et $\{r\}$ avec vision respectivement de $\{1, 2, 3\}$ et de $\{2, 4\}$. Le chemin maximal entre deux agents dans ce système est de taille $d = 2$.

Un agent ne peut déplacer un objet d'un chariot à un autre que si il sait que le chariot source n'est pas vide et que le chariot destination n'est pas rempli. Chaque agent a donc un simple problème sous contraintes, visant à minimiser la somme des distances des entités qu'il supervise, voir équation 29, qui sera utilisé en combinaison de l'algorithme présenté comme outil de planification à la place d'un MDP, dans un souci de simplification.

$$\begin{array}{llll} s_0^* = 4 & s_2^* = 3 & s_3^* = 0 & s_4^* = 0 \\ \delta_0(s_0) = 4 - s_0 & & \delta_3(s_3) = |0 - s_3| & \\ \delta_1(s_1) = 0 & & \delta_4(s_4) = (0 - s_4)^2 & \\ \delta_2(s_2) = |3 - s_2| & & & \end{array} \tag{29}$$

Le but étant donc de minimiser la somme C_ω de ces distances. Dans ce scénario, le résultat des actions est déterministe.

L'état initial du système est $\langle 0, 0, 3, 1, 3\rangle$. Dans cet exemple, chaque agent choisit son plan initial en supposant la non-existence de ses voisins. Cela donne les plans suivants :

$$\alpha\ :\ (o = 0) \qquad \beta\ :\ (p = 1, q = 0) \qquad \gamma\ :\ (r = 2)$$

Ces plans initiaux rendent le système dans l'état suivant :

$\langle 0, 1, 5, 0, 1\rangle$ avec comme valeur globale ($C_\omega = 7$).

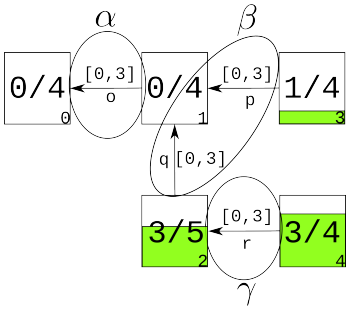

FIGURE 1 – Réseau de l'exemple composé de 5 entités (carrés), 4 actionneurs (arcs) et 3 agents ($\{\alpha, \beta, \gamma\}$) dans son état initial

Dans cet exemple, les plans des agents donnent les actions qui seront faites peu importe l'état du système. L'adaptation pour le voisinage consistera donc uniquement à envoyer les actions affectant les entités partagées.

L'agent α enverra son action $(o = 0)$ à β. Similairement γ enverra $(r = 2)$ à β. Ce dernier communiquera $(p = 1, q = 0)$ à α et $(q = 0)$ à γ. Ainsi, chaque agent pourra modifier sa vision de son environnement en prenant en compte les choix annoncés de leurs voisins. Après un nouveau calcul de plan, les nouveaux plans des agents sont :

$$\alpha \ : \ (o = 1) \qquad \beta \ : \ (p = 1, q = 2) \qquad \gamma \ : \ (r = 2)$$

Ce qui donne les gains suivant, ici la différence d'évaluation du monde après l'application du plan actuel et du nouveau plan que les agents viennent de calculer :

$$g_\alpha = 1 \qquad \text{et} \qquad g_\beta = 2 \qquad \text{et} \qquad g_\gamma = 0$$

Le gain de γ est nul, signifiant qu'il possède déjà un plan localement optimal, aucun changement ne sera donc appliqué à cet agent. L'agent β possède le gain maximum de son voisinage (2) ce qui implique qu'il gardera son nouveau plan. Un agent du voisinage d'α garde déjà son plan (β), α ne gardera donc pas le plan qu'il vient de calculer. Le nouveau plan joint résultant de cette itération amènera au système suivant :

$$\alpha \ : \ (o = 0) \qquad \beta \ : \ (p = 1, q = 2) \qquad \gamma \ : \ (r = 2)$$

$\langle 0, 3, 3, 0, 1 \rangle$ avec comme valeur globale $(C_\omega = 5)$.

Les agents ayant trouvé un plan meilleur que leur plan actuel ($gain > 0$) réinitialisent leur compteur ($d_\alpha = 2$ et $d_\beta = 2$) tandis que les autres agents le décrémentent de 1 ($d_\gamma = 1$). Puis les agents échangent leur compteur avec leur voisinage et ne gardent que le maximum ($d_\gamma = 2$).

Comme les compteurs ne sont pas nuls, une nouvelle itération amène aux plans et gains suivant :

$$\alpha \ : \ (o = 3) \qquad \beta \ : \ (p = 1, q = 2) \qquad \gamma \ : \ (r = 3)$$

$$g_\alpha = 3 \qquad \text{et} \qquad g_\beta = 0 \qquad \text{et} \qquad g_\gamma = 1$$

À cette itération, le gain de β est nul, il ne garde donc pas son plan. α et γ ont les gains maximums de leur voisinage respectif, ils garderont donc leur nouveau plan. Le nouveau plan joint sera donc :

$$\alpha \ : \ (o = 3) \qquad \beta \ : \ (p = 1, q = 2) \qquad \gamma \ : \ (r = 3)$$

$\langle 3, 0, 4, 0, 0 \rangle$ avec comme valeur globale $(C_\omega = 2)$.

Après les réductions, les nouveaux compteurs sont : $d_\alpha = 2$, $d_\beta = 1$ et $d_\gamma = 2$. β a un voisin dont la valeur du compteur est supérieure à la sienne, il la prendra donc ($d_\beta = 2$).

Une nouvelle itération sera donc effectuée menant à :

$$\alpha \ : \ (o = 3) \qquad \beta \ : \ (p = 1, q = 3) \qquad \gamma \ : \ (r = 3)$$

$$g_\alpha = 0 \qquad \text{et} \qquad g_\beta = 1 \qquad \text{et} \qquad g_\gamma = 0$$

Seul β possède un gain non nul, il gardera donc son nouveau plan. Les deux autres agents resteront avec leur plan précédent. Le nouveau plan joint sera donc :

$$\alpha \ : \ (o = 3) \qquad \beta \ : \ (p = 1, q = 3) \qquad \gamma \ : \ (r = 3)$$

$\langle 3, 1, 3, 0, 0 \rangle$ avec comme valeur globale ($C_\omega = 1$).

Les compteurs après réductions et échanges seront tous égaux à 2, l'algorithme devra donc continuer. À ce point, l'algorithme a atteint un optimum local. Néanmoins, il faudra encore deux itérations pour que la convergence soit détectée par les agents. Durant ces deux itérations, aucun agent ne pourra améliorer son plan, résultant en une réduction progressive des compteurs jusqu'à ce qu'ils atteignent tous la valeur 0 de façon simultanée, ce qui signifiera l'arrêt de la résolution.

6 Expérimentation sur des réseaux de voies navigables

Une modélisation stochastique des réseaux de voies navigables en utilisant des MDPs a été initiée dans (Desquesnes *et al.*, 2016) et adaptée à l'OCLP dans (Desquesnes *et al.*, 2017) avec de bons résultats. Un réseau de voies navigables est un système à grande échelle composé de canaux artificiels et des rivières canalisées, divisés par des écluses. Toute partie du réseau entre deux écluses est appelée bief. La préoccupation principale des gestionnaires des voies navigables est de maintenir un certain niveau d'eau sur tous les biefs pour permettre la navigation. La navigation n'est autorisée sur un bief que si et seulement si le niveau d'eau sur tous les biefs est compris entre le niveau maximum de navigation (Highest Navigation Level – HNL) et le niveau minimum de navigation (Lowest Navigation Level – LNL). Le niveau d'eau optimal est le niveau normal de navigation (Normal Navigation Level – NNL). Le niveau d'eau est perturbé par l'ouverture des écluses imposée par la navigation, des échanges incontrôlés tels que des échanges avec les nappes phréatiques ou encore la météo et, par des déplacements contrôlés résultant de l'ouverture des points de transferts (pompes et portes) entre deux biefs, par les gestionnaires. Avec l'OCLP, les entités du modèle seront les biefs, avec une discrétisation des différents niveaux d'eau comme ensemble d'états de l'entité. Similairement, les transferts d'eau discrétisés sont les actions du réseau. Dans cette application, les agents contrôlent un ensemble de points de transfert et donc observent tous les biefs qu'ils affectent. Le but est de minimiser la distance quadratique du niveau des biefs à leur NNL. Le réseau expérimental, voir figure 2, est composé de 7 biefs, représentés par des carrés, et de 14 points de transfert, représentés par des arêtes, et est planifié sur une durée arbitraire de 8 pas de temps.

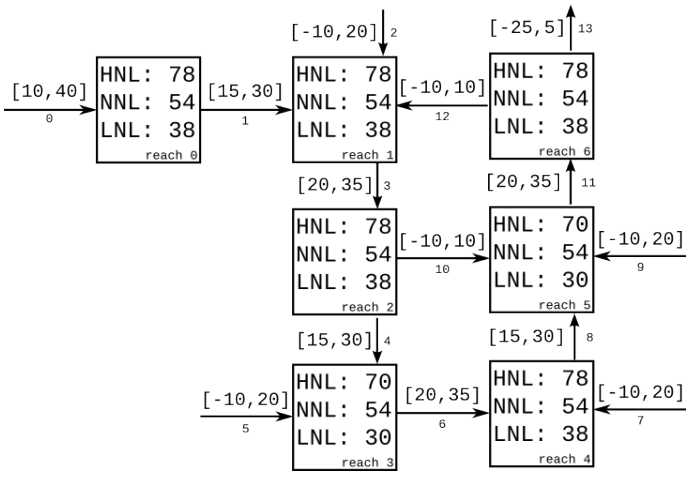

FIGURE 2 – Réseau de voies navigables expérimental composé de 7 biefs (carrés) et de 14 points de transfert (arcs)

Des décompositions, de tailles variables, de ce réseau en agents ont été proposées par un algorithme heuristique spécifique à cette application, visant à réduire la taille des fonctions de transition de tous les agents. Pour chaque décomposition, la somme des tailles des fonctions de transition des agents (en nombre de valeurs), le temps pour converger, le pourcentage du temps qu'un bief soit hors de son rectangle de

navigation (*out*) et la distance relative moyenne du bief à son NNL au cours du temps (*avg*) sont observés. Les deux dernières valeurs (*out* et *avg*) correspondent à une moyenne sur 50000 simulations des politiques produites. Ceci est dû au fait que les actions, dans ces expérimentations, correspondent à des intervalles de volumes à transférer. Le simulateur a donc choisi une valeur aléatoirement dans l'intervalle obtenue par la politique. Dans toutes les simulations, les biefs commencent à leur NNL. L'exécution de l'algorithme s'est effectuée sur un cluster, avec un agent par machine, en utilisant le framework JADE (Bellifemine *et al.*, 1999).

	taille	*durée* (s)	*out* (%)	*avg* (%)
6 agents	5.8×10^6	591	0.000	17.13
7 agents	2.9×10^6	167	0.001	15.38
8 agents	4.8×10^5	68	0.000	16.45
9 agents	4.3×10^5	35	0.000	17.27
10 agents	3.5×10^5	32	0.021	17.16
11 agents	2.9×10^5	30	0.023	17.83
12 agents	2.3×10^5	31	0.016	17.27
13 agents	1.7×10^5	30	0.009	16.64
14 agents	1.1×10^5	26	0.007	15.57

TABLE 1 – Scénarios de sept biefs

Les résultats, voir table 1, montrent une réduction évidente de la taille des fonctions de transition et du temps requis pour converger. En effet, la taille du modèle d'un agent croît exponentiellement en fonction du nombre d'entités observées et du nombre de points de transfert contrôlés. L'évaluation produit de bons résultats, puisque le pourcentage du temps hors du rectangle de navigation est infime, tout en étant légèrement influencé par le facteur aléatoire lors de l'application des politiques. Les distances relatives à l'optimal sont aussi correctes au vu de la discrétisation choisie des volumes des biefs et des volumes transférés en intervalles. Pour comparaison, la taille d'un intervalle de volume d'un bief est équivalente à une distance relative de 20% au volume optimal. À cause de la taille du problème et de la limitation des ressources de calcul à notre disposition, trouver une solution optimale de façon centralisée pour cette modélisation n'était pas possible de même pour toute décomposition de taille inférieure à 6.

7 Conclusion et travaux futurs

7.1 Conclusion

Dans cet article, la coordination hors-ligne de planifications locales (Offline Coordination of Local Planning – OCLP) a été présentée pour des systèmes à base d'entités. L'algorithme proposé permet de résoudre des grands problèmes en distribuant le modèle sur plusieurs agents connectés. Ces agents possèdent des connaissances limitées du système et construisent leur propre politique locale grâce à une coordination inter-agents durant le calcul des politiques. Comme chaque agent est défini avec sa propre politique locale, il n'y a pas besoin de communication durant l'exécution des agents.

L'approche OCLP est prouvée de terminer lorsqu'une solution localement optimale est atteinte et de converger vers cet optimum local. Elle a été utilisée avec succès pour optimiser la gestion de l'eau dans un grand réseau de voies navigables sur plusieurs pas de temps. Ces résultats expérimentaux montrent une diminution significative de la taille du modèle joint et du temps de résolution selon la décomposition du réseau en agents.

Comme cette approche se base sur la notion d'entités partagées, il sera difficile de modéliser des systèmes fortement connectés. Chaque connexion entre deux agents implique des informations redondantes qui nécessiteront plus de modélisation et de communications. Néanmoins, cela pourrait améliorer la qualité des modèles locaux en augmentant leur connaissance du monde extérieur.

7.2 Travaux futurs

L'impact de la décomposition de la structure en agents sur la qualité des solutions obtenues pourra être étudiés. L'influence du degré d'interconnexion entre les entités et entre les agents sur la capacité de passage

à l'échelle et sur les résultats obtenus pourra aussi être examinée. Une fonction de coût multicritères correspondant à une somme de coût indépendantes sur les états et les actions semble posséder les mêmes garanties de convergence que celle utilisée dans cet article et pourrait donc être étudiée. Nous allons pouvoir étudier l'intérêt et la possibilité d'utiliser des agents avec des algorithmes de planification locale hétérogènes en utilisant un langage partagé pour échanger les politiques adaptés. Finalement, cet algorithme pourrait être appliqué à d'autres formalismes de modélisation et de planification avec pas ou peu de modifications et pourraient donc être explorés.

Références

BELLIFEMINE F., POGGI A. & RIMASSA G. (1999). Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, p. 33– : London.

BELLMAN R. (1957). A Markovian Decision Process. *Journal of Mathematics and Mechanics*, **6**(4), 679–684.

BOUTILIER C., DEARDEN R., GOLDSZMIDT M. & OTHERS (1995). Exploiting structure in policy construction. In *IJCAI*, volume 14, p. 1104–1113.

CHADES I., SCHERRER B. & CHARPILLET F. (2002). A Heuristic Approach for Solving Decentralized-POMDP : Assessment on the Pursuit Problem. In *SAC '02 : Proceedings of the 2002 ACM symposium on Applied computing*, p. 57–62, Madrid, Spain : ACM.

DESQUESNES G., LOZENGUEZ G., DONIEC A. & DUVIELLA E. (2016). Dealing with large mdps, case study of waterway networks supervision. In *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection*, p. 48–59. Springer.

DESQUESNES G., LOZENGUEZ G., DONIEC A. & DUVIELLA E. (2017). Distributed MDP for water resources planning and management in inland waterway. *IFAC 2017 World Congress, Toulouse, France, 9-14 July (submited : notification of acceptance 20 february)*.

NAIR R., VARAKANTHAM P., TAMBE M. & YOKOO M. (2005). Networked Distributed POMDPs : A Synthesis of Distributed Constraint Optimization and POMDPs. In *National Conference on Artificial Intelligence*, p. 7–.

PUTERMAN M. L. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

YOKOO M. & HIRAYAMA K. (1996). Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the Second International Conference on Multi-Agent Systems*, p. 401–408.