Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes

Caen, les 6 et 7 juillet 2017

dans le cadre de la Plate-Forme de l'Intelligence Artificielle

Table des matières

Djamila Baroudi, Chihab Hanachi, Frédéric Maris.
Génération de plans à partir d'une spécification déclarative d'une collaboration

Guillaume Bono, Jilles Dibangoye, Laetitia Matignon, Florian Pereyron, Olivier Simonin.
Classification des problèmes stochastiques et dynamiques de collectes et de livraisons par des véhicules intelligents

Olivier Buffet, Vincent Thomas, Jilles Dibangoye.
MDP s-lipschitziens et ρ-POMDP non-convexes

Jean-Alexis Delamer, Yoko Watanabe, Caroline P. Carvalho Chanel.
MOMDP modeling for UAV safe path planning in an urban environment

Guillaume Desquesnes, Guillaume Lozenguez, Arnaud Doniec, Eric Duviella.
Coordination distribuée et hors-ligne de planifications locales

Alexis Ducarouge, Olivier Sigaud.
The Successor Representation as a model of behavioural flexibility

Matthieu Geist, Bilal Piot, Olivier Pietquin.
Faut-il minimiser le résidu de Bellman ou maximiser la valeur moyenne ?

Erwan Lecarpentier, Sebastian Rapp, Marc Melo, Emmanuel Rachelson.
Empirical evaluation of a Q-Learning Algorithm for Model-free Autonomous Soaring

Zoltán Nagy, Zsófia Lendek.
Quadcopter modeling and control

Mihai-Ioan Popescu, Hervé Rivano, Olivier Simonin.
Multi-cycle Coverage for Multi-robot Patrolling - application to data collection in WSNs

José-Luis Vilchis Medina, Pierre Siegel, Andrei Doncescu.
Pilotage stable d'un planeur en utilisant une logique non monotone

Olivier Gasquet, Andreas Herzig, Dominique Longin, Frédéric Maris, Maël Valais.
La logique facile avec TouIST — Formalisez et résolvez facilement des problèmes avec des solveurs SAT, SMT ou QBF

Génération de plans à partir d'une spécification déclarative d'une collaboration

Djamila Baroudi, Chihab Hanachi, Frédéric Maris

IRIT – Université de Toulouse

Résumé : Ce travail est consacré à la planification dans la gestion de crises, telles que les inondations, les tremblements de terre ou explosions de violences urbaines. L'objectif de ce travail est de générer des plans à partir d'une spécification déclarative de la collaboration entre les différents acteurs impliqués dans la gestion de crise. Nous explorons les possibilités des prouveurs SAT comme un moyen de générer efficacement ces plans qui peuvent être utilisés par une cellule de crise pour coordonner au mieux les acteurs sur le terrain. Cet article précise notre problématique, les contributions attendues, ainsi qu'une première modélisation SAT.

Mots-clés : Planification, Gestion des crises, Collaboration des processus, Prouveurs SAT.

1 Contexte

La planification des actions à réaliser est déterminante dans le cadre de la gestion de crises compte tenu des risques humains et matériels encourus. Le plus souvent, la planification dans ce contexte se base sur des documents papiers (plans de coordination) qui peuvent donner lieu à plusieurs interprétations et ne permettent pas de faire de la simulation pour comparer diverses solutions et en extraire ensuite la meilleure en terme de coût, de délai ou de risques. De nouvelles solutions à base de processus métiers (BPM) Le *et al.* (2016) voient le jour mais ces dernières manquent de formalisation et traitent de manière insuffisante les contraintes de temps et de ressources.
Compte tenu de ces observations, l'objectif à long terme de ce travail, est de proposer un outil accessible à une cellule de crise pour pouvoir générer efficacement des plans de résolution de crises, les simuler et recommander le meilleur relativement à certains critères. Le point de départ de notre approche est une description déclarative des acteurs, de leurs compétences (actions) et des relations qu'elles entretiennent entre elles. L'amélioration des performances des prouveurs SAT pour la résolution des problèmes de planification, comme le montre Rintanen (2012), nous amène à les considérer pour traiter notre problème.

2 Approche proposée

Le format textuel des plans de coordination ne permet pas leur exploitation efficace par des programmes informatiques (analyse, simulation, ...). Afin d'apporter une solution à ce problème, nous proposons un modèle qui représente les connaissances préalables en terme de gestion de crise (Figure 1) : les acteurs, leurs actions, les dépendances entre elles et les risques qu'elles traitent. Quand une crise survient, ce modèle est utilisé pour déduire un plan de résolution de crise via une planification par satisfaction de base de clauses (planification SAT) Kautz *et al.* (1992). De manière plus précise, étant donnés d'une part ces *connaissances préalables* et d'autre part un état initial et un but fixé suite aux observations d'une crise en cours, une planification SAT est formalisée afin de générer un plan solution permettant d'atteindre le but prédéfini. Le but peut correspondre au traitement d'un ou plusieurs risques.
La formalisation (pendant la crise) se compose de trois étapes : encodage formel, résolution et décodage (figure 1). Une longueur de plan k (nombre de niveaux) doit préalablement être fixée pour permettre l'*encodage formel* du problème. Cela permet de générer une formule logique en forme normale conjonctive (FNC) qui indique l'existence ou non d'un plan solution de longueur k. Une *table de symboles* est nécessaire afin de sauvegarder les correspondances entre les propositions logiques et leurs significations dans le problème d'origine. Après la génération de modèle qui satisfait la formule par le *prouveur SAT*, l'étape de *décodage* utilise la *table de symboles* afin d'avoir le *plan* solution du problème réel sous un format lisible

par la cellule de crise. Dans le cas où aucun modèle n'est trouvé, nous revenons à l'étape d'encodage tout en augmentant le nombre k de niveaux du plan.

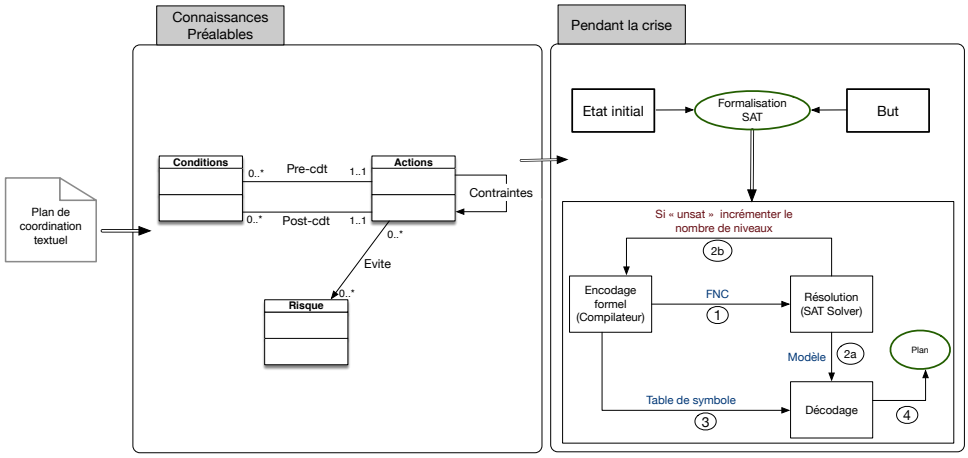


FIGURE 1 – Architecture d'une planification de gestion de crise basée sur SAT

Par ailleurs, il est nécessaire d'avoir une méthode qui permette aux agents de raisonner d'une manière autonome afin de coordonner les activités entre elles. Pour cette raison, il est important de distinguer les interdépendances et les contraintes qui peuvent se présenter lors d'une collaboration. Les contraintes représentent la manière dont les actions sont exécutées, par exemple : *fill(A, p, B)* impose que A soit exécutée avant B, et que A produise une ressource p que B consommera. Tandis que la contrainte *choice(A, B)* représente un choix exclusif entre la réalisation de A et B. D'autres contraintes ont également été définies dans le travail de Le *et al.* (2016). Nous envisageons d'étendre cet ensemble de contraintes afin de permettre la définition d'autres propriétés prenant en compte les notions de temps et de ressources. La table 1 montre la formalisation que nous proposons pour les contraintes *fill* et *choice*.

$$fill(A,p,B) \equiv \bigwedge_{i\in[1,..k]} \left(A(i) \implies \bigvee_{i<j\leq k} \left(B(j) \wedge \bigwedge_{i<l<j} \bigwedge_{\substack{C\in Actions,\\ \neg p\in Effets(C)}} \neg C(l) \right) \right)$$

$$choice(A,B) \equiv \bigwedge_{i\in[1,..k]} \left(\left(A(i) \implies \bigwedge_{j\in[1..k]} \neg B(j) \right) \wedge \left(B(i) \implies \bigwedge_{j\in[1..k]} \neg A(j) \right) \right)$$

TABLE 1 – Traduction des contraintes en formules logiques

Notre contribution vise à traduire un problème de coordination d'acteurs en un problème de planification STRIPS, Fikes & Nilsson (1971), et de le résoudre en utilisant le codage dans les espaces des états avec frame-axiomes explicatifs proposé par Kautz *et al.* (1992). Outre le domaine d'application, l'intérêt de ce travail est l'introduction de contraintes dans le codage du problème de planification, aspect qui n'a pas été pris en compte dans des travaux précédents.

Références

FIKES R. E. & NILSSON N. J. (1971). Strips : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, **2**(3-4), 189–208.

KAUTZ H. A., SELMAN B. *et al.* (1992). Planning as satisfiability. In *ECAI*, volume 92, p. 359–363 : Citeseer.

LE N.-T.-T., HANACHI C., STINCKWICH S. & HO T.-V. (2016). Discovering crisis models to help assess coordination plans. *Vietnam Journal of Computer Science*.

RINTANEN J. (2012). Planning as satisfiability : Heuristics. *Artificial Intelligence*, **193**, 45–86.

Classification des problèmes stochastiques et dynamiques de collectes et de livraisons par des véhicules intelligents

Guillaume Bono[1,2], Jilles S. Dibangoye[1,2], Laëtitia Matignon[1,2,3], Florian Pereyron[4], Olivier Simonin[1,2]

[1] INSA de Lyon, laboratoire CITI
6 avenue des Arts, 69621 Villeurbanne
prenom.nom@insa-lyon.fr

[2] INRIA, équipe Chroma
prenom.nom@inria.fr

[3] Univ Lyon, Université Lyon 1, LIRIS, CNRS, UMR5205
Villeurbanne, F-69622, France
laetitia.matignon@univ-lyon1.fr

[4] Volvo Group, Advanced Technology and Research
99 route de Lyon, 69806 Saint-Priest
florian.pereyron@volvo.com

Résumé : Les problèmes de planification de tournées de véhicules présentent une très grande richesse et disposent de nombreux raffinements dans la littérature. Les progrès récents autour des véhicules autonomes ouvrent certaines perspectives quant à leur usage pour le transport de marchandises. Nous avons étudié la classification de cette famille de problèmes, et les modèles qui en découlent, pour tenter de nous positionner dans ce domaine avec cette nouvelle approche, intégrant l'utilisation d'une flotte de véhicules autonomes et intelligents.

Mots-clés : Recherche Opérationnelle, Systèmes Multi-Agents, Véhicules autonomes, Transport Intelligent Collaboratif, Optimisation dans l'incertain.

1 Introduction

Les métiers de la logistique et du transport routier sont aujourd'hui confrontés à la révolution numérique, et sont encore assez loin d'exploiter pleinement les récents progrès en informatique et télécommunication. Dans un avenir imminent, ils auront tout intérêt à intégrer une nouvelle « révolution » : celle du véhicule autonome. Les recherches gravitant autour de ces systèmes de transports intelligents aboutissent aujourd'hui de plus en plus à des expérimentations sur routes ouvertes et à des preuves de concepts d'exploitation réelle. Néanmoins, en France, elles visent surtout le transport de personnes, et ne se sont pas encore beaucoup tournées vers le transport de marchandises.

La littérature dans le domaine du transport logistique est extrêmement riche, grâce aux enjeux économiques importants auxquels elle tente d'apporter des solutions. En tant qu'extension du problème du voyageur de commerce, le problème de planification de tournées de véhicules, introduit dans (Dantzig & Ramser, 1959), est difficile. Il présente une complexité algorithmique non polynomiale (NP-hard), comme l'ont montré (Lenstra & Kan, 1981). De plus les instances réelles de ce problème rencontrées dans l'industrie du transport ont souvent des dimensions très importantes.

Il en existe de nombreux raffinements, dont le problème de collectes et de livraisons introduit dans (Wilson *et al.*, 1971) que nous décrirons plus en détails, et que nous formulerons en problème d'optimisation mathématique, dans la deuxième section. Puis nous explorerons quelques familles de méthodes de résolution appliquées dans la littérature, pour enfin conclure sur les perspectives futures de notre travail, potentiellement tourné vers de nouvelles approches d'optimisation et d'apprentissage, utilisant des systèmes multi-agents distribués.

2 Description du problème

2.1 Concepts

Le problème de collectes et de livraisons (ou PDP pour Pickup and Delivery Problem en anglais) est une extension du problème de tournées de véhicules (VRP), lui-même variante du problème du voyageur de commerce (TSP). Dans l'idéal, résoudre un PDP consiste à trouver le parcours optimal d'une flotte de m véhicules devant transporter un ensemble de n biens d'un point de collecte à un point de livraison à travers un réseau de routes.

Une partie des demandes de transport peut arriver en cours d'exécution, rendant le problème dynamique. La proportion de ces requêtes initialement inconnues constitue le degré de dynamisme (DoD) du problème, comme l'ont défini (Lund *et al.*, 1996). De plus, de nombreux paramètres incertains peuvent être pris en compte et anticipés lors de l'optimisation, comme la congestion du réseau routier, ou les lieux de collecte et de livraison des requêtes dynamiques. On obtient à ce stade un problème dynamique et stochastique de collectes et de livraisons (SDPDP).

À ceci s'ajoute la possibilité d'intégrer diverses contraintes et degrés de liberté, comme une capacité de charge limitée (cVRP) étudiée dans les travaux de (Ralphs *et al.*, 2003), et très souvent considérée de façon implicite dans d'autres instances du problème. Une autre contrainte classique est l'introduction de fenêtres temporelles, comme le fait (Solomon, 1987) restreignant le temps pendant lequel les véhicules peuvent procéder à une prise en charge ou une livraison (PDP-TW). On peut également s'intéresser au cas où l'autonomie énergétique des véhicules est limité (eVRP) comme (Artmeier *et al.*, 2010), ou encore à la possibilité de transférer des biens entre véhicules (PDP-T), traitée par (Cortés *et al.*, 2010). La figure 1 ci-dessous illustre les différents éléments intervenant dans la version étendue du problème.

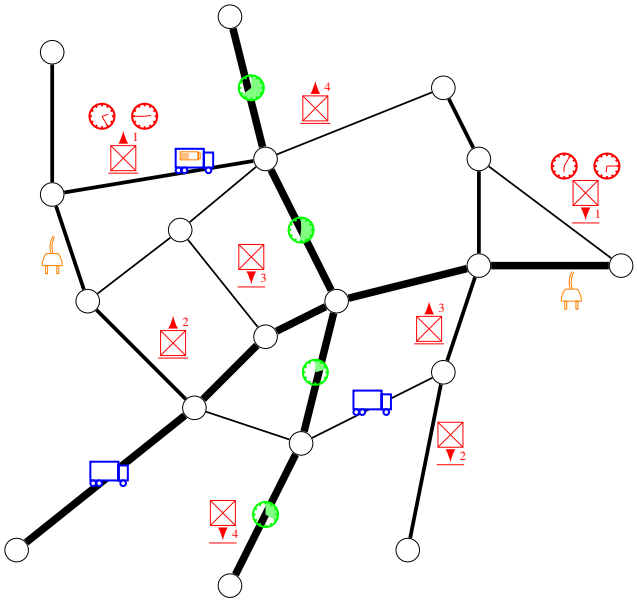


FIGURE 1 – Illustration du problème : 3 véhicules (en bleu) doivent effectuer 4 livraisons (en rouge) ayant des horaires de collecte et de livraison limités (cadrans rouges). Ils doivent tenir compte des temps de trajets variables sur chaque axe (cadrans verts), et de leur capacité énergétique limitée (pile orange), les forçant à se rendre régulièrement à une station de charge (prise orange).

Dans un contexte différent, le problème de transport à la demande (ou DARP, pour Dial-A-Ride Problem en anglais), qui s'applique au transport de passagers, est une forme de PDP où chaque personne n'occupe qu'un seul siège, et où certaines contraintes plus spécifiques sont prise en compte (temps de détour maximum par exemple). La Figure 2 ci-dessous détaille la construction de ces différents raffinements, et explicite les acronymes anglais.

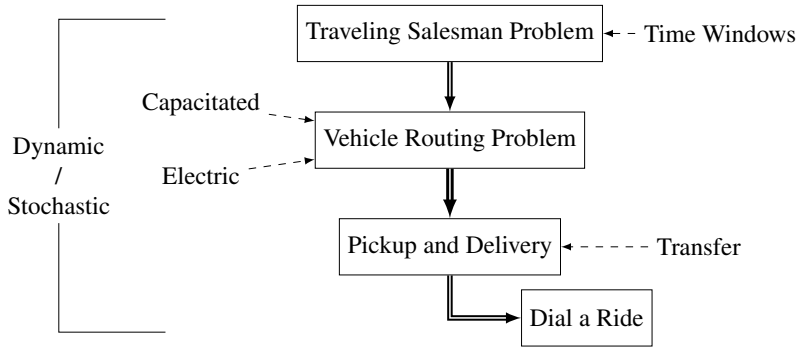


FIGURE 2 – Taxonomie des variantes du problème

2.2 Formulation mathématique

Dans cette section, nous reprendrons le formalisme utilisé dans leur chapitre de livre par (Desaulniers *et al.*, 2001). Nous essayerons de reconstruire de façon incrémentale le problème et ses raffinements pour mieux identifier les inéquations et les contraintes auxquelles elles correspondent. Nous tâcherons aussi de faire ressortir les variables de décision binaire x_{ijk}, définissant l'assignation des requêtes aux véhicules ainsi que l'enchaînement par lequel elles seront desservies, et les variables continues (par exemple L_{ik}, T_{ik}, E_{ik}) maintenant la cohérence de l'état du système d'un nœud à l'autre. Cette mise en valeur progressive des différentes contraintes vise à faciliter les parallèles avec d'autres formalisations. Nous montrerons également comment étendre le programme mathématique original des auteurs pour enrichir l'état du système en prenant pour exemple l'intégration de la contrainte d'autonomie énergétique limitée (eVRP).

2.2.1 VRP

Ramené au cas le plus simple du problème de planification de tournées où sont donnés l'ensemble des lieux à visiter $N = \{1, \ldots, n\}$ et les positions de départ et d'arrivée $o(k), d(k)$ de chaque véhicule $k \in K$, on construit un ensemble de $|K| = m$ graphes complets $\{G_k = (V_k, A_k)\}_{k \in K}$. Chaque nœud correspond à la visite d'un véhicule sur un lieu : $V_k = \{o(k), d(k)\} \cup N_k$, avec $N_k \subset N$ le sous-ensemble des lieux que le véhicule est autorisé à desservir. Chaque arc $(i, j) \in A_k$, pondéré par un coût c_{ijk}, représente le trajet par le véhicule $k \in K$ d'un lieu i à un lieu j. Outre la prise en compte des spécificités de chaque véhicule, cette modélisation en m graphes permet de visualiser plus simplement l'existence d'un arc par véhicule et par paire de nœud de service nécessaire à maintenir la cohérence de chaque route. L'optimisation repose alors sur le programme en nombres entiers (binaires) suivant :

$$\min_{x_{ijk}} \sum_{k\in K} \sum_{ij\in A_k} x_{ijk} c_{ijk} \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in K} \sum_{j\in N_k\cup\{d(k)\}} x_{ijk} = 1 \quad \forall i \in N \tag{2}$$

$$\sum_{j\in N_k\cup\{d(k)\}} x_{o(k),j,k} = 1 \quad \forall k \in K \tag{3}$$

$$\sum_{j\in N_k\cup\{d(k)\}} x_{ijk} - \sum_{j\in N_k\cup\{o(k)\}} x_{jik} = 0 \quad \forall k \in K, \forall i \in N_k \tag{4}$$

$$\sum_{j\in N_k\cup\{o(k)\}} x_{j,d(k),k} = 1 \quad \forall k \in K \tag{5}$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in K, \forall ij \in A_k$$

Les variables de décision binaires x_{ijk} indique si le véhicule $k \in K$ servira le nœud $j \in N_k \cup d(k)$ après avoir visiter le nœud $i \in N_k \cup o(k)$. La contrainte (2) garantie que chaque lieu est desservi une et une seule fois, en imposant qu'un seul arc sortant du nœud soit utilisé. Elle aurait pu s'écrire de façon équivalente en utilisant les arcs entrant dans le nœud. Les contraintes (3) à (5) assurent la cohérence des trajets par conservation du flux.

Le problème ainsi posé peut donner lieu à des résultats infaisables et absurdes, car rien n'empêche la solution de contenir des boucles. Il faudrait ajouter une variable entière pour chaque nœud, qui tienne compte de leur ordre de passage. Néanmoins dans la formulation complète de (Desaulniers *et al.*, 2001) que nous reconstruisons, ce rôle sera rempli par les variables permettant d'intégrer les fenêtres temporelles. Nous ne détaillerons donc pas d'avantage ici, pour ne pas surcharger les notations.

2.2.2 PDP

Pour étendre cette formulation au problème de collectes et de livraisons, l'ensemble N se décompose en un ensemble des lieux de collecte $P = \{1, \ldots, n\}$ et un ensemble de lieux de livraison correspondants $D = \{n+1, \ldots, 2n\}$. Les sous-ensembles $P_k \subset P$ et $D_k \subset D$ répondent aux mêmes limitations pour le véhicule $k \in K$ que $N_k \subset N$. Il faut ajouter la contrainte (6) qui assure que le même véhicule réalise la collecte et la livraison d'un bien donné :

$$\sum_{j\in N_k} x_{ijk} - \sum_{j\in N_k} x_{j,n+i,k} = 0 \quad \forall k \in K, \forall i \in P_k \tag{6}$$

Ici aussi, il faudrait maintenir un ordre de passage pour garantir que la livraison ait lieu après la collecte correspondante. Pour ne pas introduire de notations temporaires, nous résoudrons ce défaut dans la partie modélisant le respect de fenêtres temporelles.

2.2.3 cVRP

La prise en compte de la capacité maximale C_k de chaque véhicule $k \in K$ et de la charge l_i de chaque bien $i \in P$ (en posant pour des facilités de notations : $l_{n+i} = -l_i \quad \forall n+i \in D$) donne lieu aux contraintes suivantes :

$$x_{ijk} \cdot (L_{ik} + l_j - L_{jk}) = 0 \quad \forall k \in K, \forall ij \in A_k \tag{7}$$

$$l_i \leq L_{ik} \leq C_k \quad \forall k \in K, \forall i \in P_k \tag{8}$$

$$0 \leq L_{n+i,k} \leq C_k - l_i \quad \forall k \in K, \forall n+i \in D_k \tag{9}$$

$$L_{o(k),k} = 0 \quad \forall k \in K \tag{10}$$

Les variables d'état continues positives L_{ik} représentent la charge d'un véhicule $k \in K$ après avoir desservi le nœud $i \in N_k$. La contrainte (7) garantie une évolution cohérente de la charge, les contraintes (8) et (9) maintiennent les conditions limites, et la contrainte (10) donne leur valeur initiale aux variables.

2.2.4 PDP-TW

La restriction des collectes et des livraisons à des fenêtres temporelles strictes notées $[a_i, b_i] \quad \forall i \in N$, accompagnées d'un temps de service (charge ou décharge) s_i, requiert d'estimer le temps de trajet t_{ijk} pour chaque arc. Le départ et l'arrivée des véhicules peuvent aussi être munis d'une fenêtre temporelle pour limiter leur temps d'utilisation. Les nouvelles contraintes ci-dessous interviennent :

$$x_{ijk} \cdot (T_{ik} + s_i + t_{ijk} - T_{jk}) = 0 \quad \forall k \in K, \forall ij \in A_k \tag{11}$$
$$a_i \leq T_{ik} \leq b_i \quad \forall k \in K, \forall i \in V_k \tag{12}$$
$$T_{ik} + t_{i,n+i,k} \leq T_{n+i,k} \quad \forall k \in K, \forall i \in P_k \tag{13}$$

Les variables d'état continues positives T_{ik} représentent le temps de début de service du véhicule $k \in K$ au nœud $i \in V_k$. La contrainte (11) garantie une évolution cohérente de ces temps de passage, la contrainte (12) assure le respect des fenêtres temporelles, et la contrainte (13) empêche les livraisons d'être réalisées avant les collectes correspondantes.

2.2.5 eVRP

Des deux sous-sections précédentes se dégage une forme commune permettant d'intégrer diverses variables d'état et de maintenir leur cohérence par rapport aux décisions prises. On peut ainsi par exemple faire intervenir la notion d'autonomie énergétique limitée de véhicules électriques.

Pour cela, notons B_k la capacité des batteries de chaque véhicule $k \in K$, et attribuons à chaque arc $ij \in A_k$ un coût en énergie w_{ijk}. Un ensemble S de nouveaux nœuds représentant les stations de charge est intégré à tous les graphes G_k. Ces stations ne sont pas soumises à la contrainte de visite unique imposée pour les autres nœuds par l'équation (2). De plus, dans la contrainte (11), leurs visites ont un temps de service s_i variable, fonction du niveau d'énergie du véhicule : $\forall i \in S, s_i = (B_k - E_{ik})/\rho_i$, avec ρ_i la puissance de charge disponible. Chaque nœud est aussi affublé d'une variation d'énergie $q_i \leq 0$ négative ou nulle, sauf pour les stations, où cette variation est positive et vaut $q_i = (B_k - E_{ik})$, si on se limite au cas où le plan ne peut pas faire intervenir des rechargements incomplets.

$$x_{ijk} \cdot (E_{ik} + q_i - w_{ijk} - E_{jk}) = 0 \quad \forall k \in K, \forall ij \in A_k \tag{14}$$
$$|q_i| \leq E_{ik} \leq B_k \quad \forall k \in K, \forall i \in V_k \setminus S \tag{15}$$
$$E_{o(k),k} = B_k \quad \forall k \in K \tag{16}$$

Les variables d'état continues positives E_{ik} représentent le niveau d'énergie du véhicule $k \in K$ lorsqu'il atteint le nœud $i \in V_k$. La contrainte (14) garantie une évolution cohérente du niveau d'énergie, la contrainte (15) impose que le véhicule ait suffisamment d'énergie avant de servir un nœud autre qu'une station, et la contrainte (16) donne leur valeur initiale aux variables.

2.2.6 DPDP

Ajouter une dimension dynamique au problème signifie que l'ensemble des données d'entrée est amené à évoluer lors de l'exécution du plan. Dans le cas du problème de collectes et de livraisons, le principal degré de dynamisme pouvant être intégré au problème est le taux de requêtes inconnues en début de planification. L'ensemble des nœuds n'est donc pas connu dans son intégralité avant le début de la tournée de chaque véhicule.

Dans un cadre dynamique, il est difficile de définir l'optimalité d'une solution. En effet, le plan construit de façon incrémentale avec des requêtes découvertes pendant l'exécution a de fortes chances d'être sous optimal par rapport à celui qui aurait été élaboré en ayant connaissance de toutes les requêtes dès le départ.

Ce dynamisme met aussi d'autant plus en avant le fait qu'il n'existe peut-être pas de solution servant toutes les requêtes et respectant l'ensemble des contraintes. Il est alors utile d'introduire une mesure de qualité du service, pouvant par exemple prendre en compte : le taux de requêtes desservies (garantie de service), le retard sur les fenêtres temporelles, le temps de détour maximum par requête, l'écart à l'optimum (regret) par rapport au cas statique où tout est fixé à l'avance.

Pour permettre un service incomplet, la contrainte (2) doit être relâchée, en introduisant des variables d'écart ξ_i. De façon similaire, la contrainte (12) peut être relâchée par l'ajout de variables ζ_i pour permettre des retards. La fonction objectif est modifiée pour décourager l'utilisation de ces variables d'écart, avec deux nouveaux meta-paramètres $\alpha, \beta \in \mathbb{R}$ permettant d'équilibrer les coûts.

$$\min_{x_{ijk}} \sum_{k \in K} \sum_{ij \in A_k} x_{ijk} c_{ijk} + \alpha \sum_{i \in P} \xi_i + \beta \sum_{i \in N} |\zeta_i| \tag{17}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j \in N_k \cup \{d(k)\}} x_{ijk} + \xi_i = 1 \quad \forall i \in P \tag{18}$$

$$a_i \leq T_{ik} + \zeta_i \leq b_i \quad \forall k \in K, \forall i \in V_k \tag{19}$$

$$\xi_i \in \{0, 1\} \quad \forall i \in P$$

$$\zeta_i \in \mathbb{R} \quad \forall i \in N$$

2.2.7 SPDP

Le caractère stochastique du problème peut être introduit en considérant plusieurs sources d'incertitude :

— Les temps de trajets t_{ijk} dépendant de l'état de congestion du réseau routier et d'événements difficiles à modéliser de façon déterministe,
— La consommation d'énergie des véhicules w_{ijk} pouvant elle aussi s'éloigner fortement des modèles déterministes les plus simples,
— Les coûts c_{ijk}, liés aux deux points précédents,
— Les pannes sur des véhicules, pouvant invalider une partie du plan,
— La position des lieux de collectes et de livraisons, dont la forme de la distribution dépend de l'instance du problème considérée (potentiellement uniforme pour du transport de personnes, et peut-être plus ponctuelle pour du transport de marchandises…)

Bien que le problème puisse alors s'exprimer comme un programme stochastique (SP), comme dans les travaux de (Laporte & Louveaux, 1993), beaucoup d'auteurs préfèrent reformuler le problème sous la forme d'un Processus de Décision Markovien, comme (Simão *et al.*, 2009) ou (Cortés *et al.*, 2009). Dans ce modèle, un vecteur $s \in \mathcal{S}$ décrit l'état du système, à partir duquel une politique $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ indique les probabilités de choisir chaque action $a \in \mathcal{A}$. Une fonction de transition $f : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ donne une distribution de probabilité sur l'état suivant du système s', tandis qu'une fonction de récompense $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ fournit la récompense associée à cette transition. L'optimisation de la politique est alors guidée par la maximisation de la récompense amortie, accumulée sur une suite de transitions constituant des épisodes. Les concepts d'apprentissage par renforcement utilisés pour optimiser la politique π dans notre contexte, comme l'approximation de la fonction de valeur, sont détaillés dans le livre de (Sutton & Barto, 1998).

2.2.8 PDP-T

Permettre les transferts de biens d'un véhicule à l'autre donne un degré de liberté supplémentaire au problème, et complexifie énormément l'optimisation. Deux cas peuvent être identifiés : celui où les lieux d'échange sont fixés à l'avance, et celui où les transferts peuvent avoir lieu n'importe où.

C'est dans cet espace d'actions plus large que (Coltin & Veloso, 2014) tente d'explorer la possibilité de transfert, en se limitant néanmoins dans leur expérimentation à identifier les intersections existantes entre les routes individuelles de chaque robot à l'itération courante, et à explorer de façon exhaustive tous les transferts pouvant avoir lieu à ce point, sans autoriser de détours.

Pour revenir à notre modèle, on peut imaginer dans le premier cas ajouter ces lieux de transfert connus aux nœuds des graphes, avec un nœud par bien à transporter et par véhicule, et ajouter des contraintes s'apparentant à celles déjà introduites pour ne pas transférer un bien avant de le collecter, pour respecter la capacité des véhicules et la cohérence temporelle.

Dans le deuxième cas, il est assez intuitif de visualiser un transfert comme la création d'un nouveau nœud de livraison intermédiaire pour le véhicule transportant le bien, et la création d'un nouveau nœud de collecte pour tous les autres véhicules de la flotte. Les possibilités de créer un nœud de transfert sont innombrables, cette décision est donc souvent guidée par des heuristiques.

3 Approches de résolution existantes

De nombreux passages en revue des différentes approches aux problèmes de collectes et de livraisons existent dans la littérature, comme celle mené récemment par (Ritzinger *et al.*, 2016), qui se focalise sur les méthodes intégrants les dimensions dynamiques et stochastiques du problème. Le Table 1 ci-dessous synthétise et étend ce passage en revue à d'autres exemples extrait de la littérature, et les algorithmes qui ont été appliqués, dont le nom sera donné en anglais pour ne pas perdre en spécificité.

	Problem flavors							*Var. properties*			
	Traveling Salesman Problem	Vehicle Routing Problem	Pickup and Delivery Problem	⋱ with capacitated vehicles	⋱ with electric vehicles	⋱ with Time Window	⋱ with Transfers	Dynamic replanning	Stochastic demand	Stochastic customers	Stochastic travel times
Branch-and-bound / cut		1,22		1,22	22	22					
Chance Constrained Programming		2,3		2,3					2		3
Stochastic Programming with recourse		3,4							4		3
Approximate Linear Programming	5								5		5
Approximate Dynamic Programming		8,9	6,7		(7)	6,7	7	9	8,9	6,7	6,7
Variable Neighboorhood Search		21	10	10,21	21	10,21					
Stochastic VNS		21	12	12		12		12		12	12
Multiple Plan Approach			11,12	11,12		11,12		11,12			
Multiple Scenario Approach			11,12	11,12		11,12		11,12		11,12	12
Genetic Algorithm		13	14	13,14		14		13,14		14	14
Particle Swarm Optimization			15,16	15,16		15		15	15		
Auction-based algorithm			17,18	17,18		17,18	18	17,18			
Monte-Carlo Tree Search	20	19		19		20					

TABLE 1 – Exemple de méthodes appliquées aux différentes variantes du problème

Références

1	(Fisher, 1994)	12	(Schilde *et al.*, 2011)
2	(Dror *et al.*, 1993)	13	(Marinakis & Marinaki, 2010)
3	(Laporte *et al.*, 1992)	14	(Sáez *et al.*, 2008)
4	(Laporte & Louveaux, 1993)	15	(Cortés *et al.*, 2009)
5	(Toriello *et al.*, 2014)	16	(Chen *et al.*, 2016)
6	(Simão *et al.*, 2009)	17	(Mes *et al.*, 2013)
7	(Bouzaiene-Ayari *et al.*, 2016)	18	(Coltin & Veloso, 2014)
8	(Meisel *et al.*, 2011)	19	(Mańdziuk & Nejman, 2015)
9	(Novoa & Storer, 2009)	20	(Rimmel *et al.*, 2011)
10	(Parragh *et al.*, 2010)	21	(Schneider *et al.*, 2014)
11	(Bent & Hentenryck, 2004)	22	(Desaulniers *et al.*, 2016)

3.1 Recherche par voisinage

La méthode de recherches par voisinage (VNS) introduite par (Mladenović & Hansen, 1997) semble être une référence de pointe dans la littérature traitant de problèmes dynamiques. (Parragh *et al.*, 2010) applique cette méthode générale au problème du transport à la demande (DARP). À partir d'une solution initiale, la méthode repose sur diverses heuristiques d'échanges entre véhicules, d'insertions et de décalages de tâches dans leurs plans. Elles servent à construire des solutions « voisines » dont on conserve la plus prometteuse lors d'une recherche locale, avant d'explorer de nouveaux voisinages pour échapper aux minima locaux.

La variante stochastique (SVNS) de (Gutjahr *et al.*, 2007) utilise un modèle génératif pour estimer la valeur accumulée par une solution sur un court horizon. Elle a été adaptée au problème de transport à la demande par (Schilde *et al.*, 2011). Introduites par (Bent & Hentenryck, 2004), l'approche par plans multiples (MPA), et son équivalent stochastique l'approche par scénarios multiples (MSA), appliquent la recherche par voisinage à un ensemble de solutions qui évoluent au déclenchement d'événements comme

l'arrivée d'une nouvelle requête, ou le départ d'un véhicule. Une fonction de consensus permet de générer un plan à exécuter en ligne à partir de l'ensemble de solutions.

(Schilde *et al.*, 2014) explorent ces différentes variantes (VNS, SVNS, MPA, MSA) combiné à un algorithme de planification par blocs et mettent en avant l'intérêt de prendre en compte les phénomènes dynamiques et d'anticiper les éléments stochastiques lors de l'optimisation.

3.2 Programmation dynamique approchée

La programmation dynamique approchée telle qu'elle est introduite par (Powell, 2007) présente le problème sous la forme d'un ensemble de ressources (dans notre problème, la flotte de véhicules) et de tâches (les requêtes de transport) dotées d'un vecteur d'attributs. L'état du système est un immense vecteur dont chaque élément correspond au nombre de ressources ayant une combinaison particulière d'attributs. La prise de décision consiste à affecter les tâches aux ressources. L'évolution du système intervient en deux temps, faisant d'abord évoluer les comptes des différents type de ressources suite à la décision, puis perturbant ce nouvel état par les phénomènes stochastiques issus de l'environnement. L'optimisation d'une politique d'assignation passe par l'estimation de la fonction de valeur par les méthodes classiques de l'apprentissage par renforcement comme l'approximation de la fonction de valeur ou les traces d'éligibilités (Sutton & Barto, 1998).

(Simão *et al.*, 2009) appliquent ce modèle au problème de la planification de tournée de flotte au sein d'une entreprise de transport américaine, donnant au final un véritable système d'aide à la décision utilisé dans l'industrie. Après avoir entrainé le modèle à reproduire les résultats historiques de l'entreprise grâce à de nombreuses traces enregistrées, ils démontrent l'intérêt de leur modèle en proposant des méthodes d'estimation de gains suite à un changement de composition de flotte, et une estimation de la valeur marginale de chaque type de ressources disponible. Le modèle passe à l'échelle de ce problème aux très larges dimensions grâce à un mécanisme d'agrégation réduisant efficacement la taille du vecteur d'état.

(Bouzaiene-Ayari *et al.*, 2016) proposent une autre mise en place de ce modèle pour aider à la planification de l'assignation de locomotives aux différents trajets à effectuer, en prenant en compte des contraintes très serrées concernant leur maintenance et la disponibilité réduite des ateliers capables de l'effectuer. Le système final est constitué de plusieurs algorithmes de résolutions travaillant à différentes échelles d'agrégation. Leurs résultats montrent l'influence des différents paramètres d'optimisation, et de fortes réductions des temps d'inactivité et des files d'attente aux ateliers.

L'utilisation de la programmation dynamique approchée et des techniques d'apprentissage par renforcement citées ci-dessus n'est bien sûr pas limité à cette formalisation en ressources et tâches issue des travaux de (Powell, 2007).

3.3 Recherche arborescente Monte-Carlo

Rencontrant beaucoup de succès dans le domaine des jeux (échecs, go...), cette méthode de recherche « meilleur d'abord » consiste à construire progressivement un arbre de décision en explorant en priorité les branches les plus prometteuses. Une politique d'arbre, comme celle utilisée dans l'algorithme UCT (pour Upper-Confidence bound 1 applied to Tree) présentée par (Kocsis & Szepesvári, 2006), dicte le choix d'actions jusqu'à un nœud feuille, à partir duquel une nouvelle action est explorée, créant un nouveau nœud en estimant l'état du système par un modèle génératif. Cet état est évalué en procédant à plusieurs simulations choisissant les actions par une politique plus simple (au hasard, par exemple) jusqu'à atteindre un état terminal. Autant de simulations que possible sont effectuées dans le temps de calcul imparti et la valeur obtenue est rétro-propagée jusqu'au nœud racine.

(Couetoux, 2013) présente dans sa thèse différentes extensions de la recherche arborescente classique permettant de travailler sur des espaces d'action continus et d'exploiter les valeurs obtenues dans des branches parallèles en comparant les états et actions en utilisant des fonctions noyaux. Il applique entre autre ces méthodes au problème de planification de tournées de véhicules avec fenêtres temporelles.

L'utilisation de cette méthode dans le cadre du transport logistique semble plutôt focalisé sur des problèmes statiques. Néanmoins l'engouement récent dont elle bénéficie et les extensions existantes pour passer à des états et des actions continus nous permettrons peut-être d'explorer cette approche dans des cas plus complexes. D'autant plus que dans une architecture de décision distribuée, des objectifs collaboratifs mais aussi compétitifs entre agents peuvent être identifiés, nous rapprochant d'un modèle de jeu stochastique pour lequel MCTS s'est illustré avec succès.

Les méthodes par recherche arborescente peuvent facilement bénéficier d'heuristiques et d'apport de connaissances expertes pour guider le choix d'actions à explorer dans les différentes phases de l'algorithme. Il n'est également pas nécessaire de poursuivre les simulations jusqu'à un état final, en utilisant des estimateurs de la fonction de valeur pour donner un score approximatif à l'état atteint.

4 Conclusion et discussions

Le problème de collectes et de livraisons dans sa forme dynamique et stochastique présente un grand nombre de défis à relever. Les grandes dimensions des instances du problème habituellement rencontrées et les contraintes temporelles de la prise de décision en ligne ont poussé les recherches en Recherche Opérationnelle dans la direction d'heuristiques efficaces et rapides, comme celles utilisées dans le VNS appliqué au DARP par (Parragh *et al.*, 2010). Nous souhaitons poursuivre les travaux visant une prise de décision en ligne exploitant l'information disponible dans un environnement dynamique et incertain. De plus, inspiré par les travaux récents autour des systèmes de transports intelligents collaboratifs (cITS), nous envisageons la possibilité de distribuer la prise de décision sur des agents autonomes.

Cette architecture distribuée prend d'autant plus de sens quand les agents peuvent prendre certaines décisions de façon indépendante. Cela pourrait être le cas dans le contexte de problèmes de planification de tournées si l'on suppose possible d'obtenir une « bonne » assignation initiale des requêtes aux différents véhicules. Chaque véhicule pourrait alors planifier sa tournée indépendamment avec son sous-ensemble de requêtes sauf lors de négociations ponctuelles avec les autres véhicules pour l'assignation de nouvelles requêtes, la détection de pannes ou de plans invalidés, ou encore le transfert de biens entre véhicules.

Une piste intéressante dans la direction d'un apprentissage par renforcement distribué au sein d'un système multi-agents relativement indépendants a été soulevées par (Melo & Veloso, 2009). En utilisant une action « de synchronisation » particulière permettant de se coordonner ponctuellement, ils parviennent à limiter l'empreinte mémoire de l'approximation de la fonction de Q-valeurs. D'autres pistes sont orientées vers des algorithmes d'optimisation par enchères, comme dans (Coltin & Veloso, 2014) ou (Mes *et al.*, 2013).

Cette article prospectif ouvre la voie dans la suite de nos travaux à l'application de ce genre de méthodes distribuées d'optimisation et d'apprentissage par renforcement. Elles ont fait leurs preuves dans des cadres dynamiques et stochastiques dans la littérature des transports intelligents et de la robotique. Nous souhaitons les appliquer à notre problème de planification de tournées, plutôt lié à la recherche opérationnelle, en intégrant un maximum de contraintes, dont les fenêtres temporelles et l'autonomie énergétique limitée, dans un environnement dynamique et incertain.

Références

ARTMEIER A., HASELMAYR J., LEUCKER M. & SACHENBACHER M. (2010). *The Shortest Path Problem Revisited : Optimal Routing for Electric Vehicles*, In R. DILLMANN, J. BEYERER, U. D. HANEBECK & T. SCHULTZ, Eds., *KI 2010 : Advances in Artificial Intelligence : 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24, 2010. Proceedings*, p. 309–316. Springer Berlin Heidelberg : Berlin, Heidelberg.

BENT R. W. & HENTENRYCK P. V. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, **52**(6), 977–987.

BOUZAIENE-AYARI B., CHENG C., DAS S., FIORILLO R. & POWELL W. B. (2016). From single commodity to multiattribute models for locomotive optimization : A comparison of optimal integer programming and approximate dynamic programming. *Transportation Science*, **50**(2), 366–389.

CHEN M.-C., HSIAO Y.-H., REDDY R. H. & TIWARI M. K. (2016). The self-learning particle swarm optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks. *Transportation Research Part E : Logistics and Transportation Review*, **91**, 208 – 226.

COLTIN B. & VELOSO M. (2014). Online pickup and delivery planning with transfers for mobile robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, p. 5786–5791.

CORTÉS C. E., MATAMALA M. & CONTARDO C. (2010). The pickup and delivery problem with transfers : Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, **200**(3), 711 – 724.

CORTÉS C. E., SÁEZ D., NÚÑEZ A. & MUÑOZ-CARPINTERO D. (2009). Hybrid adaptive predictive control for a dynamic pickup and delivery problem. *Transportation Science*, **43**(1), 27–42.

COUETOUX A. (2013). *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. Theses, Université Paris Sud - Paris XI.

DANTZIG G. B. & RAMSER J. H. (1959). The truck dispatching problem. *Management Science*, **6**(1), 80–91.

DESAULNIERS G., DESROSIERS J., ERDMANN A., SOLOMON M. M. & SOUMIS F. (2001). The vehicle routing problem. chapter VRP with Pickup and Delivery, p. 225–242. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics.

DESAULNIERS G., ERRICO F., IRNICH S. & SCHNEIDER M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, **64**(6), 1388–1405.

DROR M., LAPORTE G. & LOUVEAUX F. V. (1993). Vehicle routing with stochastic demands and restricted failures. *Zeitschrift für Operations Research*, **37**(3), 273–283.

FISHER M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, **42**(4), 626–642.

GUTJAHR W. J., KATZENSTEINER S. & REITER P. (2007). *A VNS Algorithm for Noisy Problems and Its Application to Project Portfolio Analysis*, In J. HROMKOVIČ, R. KRÁLOVIČ, M. NUNKESSER & P. WIDMAYER, Eds., *Stochastic Algorithms : Foundations and Applications : 4th International Symposium, SAGA 2007, Zurich, Switzerland, September 13-14, 2007. Proceedings*, p. 93–104. Springer Berlin Heidelberg : Berlin, Heidelberg.

KOCSIS L. & SZEPESVÁRI C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning*, p. 282–293 : Springer.

LAPORTE G., LOUVEAUX F. & MERCURE H. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science*, **26**(3), 161–170.

LAPORTE G. & LOUVEAUX F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, **13**(3), 133 – 142.

LENSTRA J. K. & KAN A. H. G. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, **11**(2), 221–227.

LUND K., MADSEN O., RYGAARD J. & OF DENMARK. INSTITUTE OF MATHEMATICAL MODELLING T. U. (1996). *Vehicle Routing Problems with Varying Degrees of Dynamism*. IMM-REP. Technical Univ.

MAŃDZIUK J. & NEJMAN C. (2015). *UCT-Based Approach to Capacitated Vehicle Routing Problem*, In L. RUTKOWSKI, M. KORYTKOWSKI, R. SCHERER, R. TADEUSIEWICZ, L. A. ZADEH & J. M. ZURADA, Eds., *Artificial Intelligence and Soft Computing : 14th International Conference, ICAISC 2015, Zakopane, Poland, June 14-18, 2015, Proceedings, Part II*, p. 679–690. Springer International Publishing : Cham.

MARINAKIS Y. & MARINAKI M. (2010). A hybrid genetic – particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Applications*, **37**(2), 1446 – 1455.

MEISEL S., SUPPA U. & MATTFELD D. (2011). *Serving Multiple Urban Areas with Stochastic Customer Requests*, In H.-J. KREOWSKI, B. SCHOLZ-REITER & K.-D. THOBEN, Eds., *Dynamics in Logistics : Second International Conference, LDIC 2009, Bremen, Germany, August 2009, Proceedings*, p. 59–68. Springer Berlin Heidelberg : Berlin, Heidelberg.

MELO F. S. & VELOSO M. (2009). Learning of coordination : Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, p. 773–780 : International Foundation for Autonomous Agents and Multiagent Systems.

MES M., VAN DER HEIJDEN M. & SCHUUR P. (2013). Interaction between intelligent agent strategies for real-time transportation planning. *Central European Journal of Operations Research*, **21**(2), 337–358.

MLADENOVIĆ N. & HANSEN P. (1997). Variable neighborhood search. *Computers and Operations Research*, **24**(11), 1097 – 1100.

NOVOA C. & STORER R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, **196**(2), 509 – 515.

PARRAGH S. N., DOERNER K. F. & HARTL R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, **37**(6), 1129 – 1138.

POWELL W. B. (2007). *Approximate Dynamic Programming : Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

RALPHS T., KOPMAN L., PULLEYBLANK W. & TROTTER L. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, **94**(2), 343–359.

RIMMEL A., TEYTAUD F. & CAZENAVE T. (2011). Optimization of the Nested Monte-Carlo Algorithm on the Traveling Salesman Problem with Time Windows. In *Evostar*, Turin, Italy.

RITZINGER U., PUCHINGER J. & HARTL R. F. (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, **54**(1), 215–231.

SCHILDE M., DOERNER K. & HARTL R. (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers and Operations Research*, **38**(12), 1719 – 1730.

SCHILDE M., DOERNER K. & HARTL R. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, **238**(1), 18 – 30.

SCHNEIDER M., STENGER A. & GOEKE D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, **48**(4), 500–520.

SIMÃO H. P., DAY J., GEORGE A. P., GIFFORD T., NIENOW J. & POWELL W. B. (2009). An approximate dynamic programming algorithm for large-scale fleet management : A case application. *Transportation Science*, **43**(2), 178–197.

SOLOMON M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254–265.

SUTTON R. & BARTO A. (1998). *Reinforcement Learning : An Introduction*. A Bradford book. Bradford Book.

SÁEZ D., CORTÉS C. E. & NÚÑEZ A. (2008). Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers and Operations Research*, **35**(11), 3412 – 3438. Part Special Issue : Topics in Real-time Supply Chain Management.

TORIELLO A., HASKELL W. B. & POREMBA M. (2014). A dynamic traveling salesman problem with stochastic arc costs. *Operations Research*, **62**(5), 1107–1125.

WILSON N. H. M., SUSSMAN J. M. & WONG H.-K. (1971). *Scheduling algorithms for a dial-a-ride system / by Nigel H.M. Wilson ... [et al.]*. MIT, Urban Systems Laboratory Cambridge, Mass.

MDP s-lipschitziens et ρ-POMDP non-convexes

Olivier Buffet[1,2], Vincent Thomas[2,1], Jilles Dibangoye[3]

[1] INRIA Nancy Grand-Est, LORIA
615, rue du jardin botanique, Villers-lès-Nancy
prenom.nom@loria.fr et https://members.loria.fr/olivier.buffet

[2] Université de Lorraine / CNRS, LORIA
Campus scientifique Victor Grignard, Vandœuvre-lès-Nancy
prenom.nom@loria.fr et https://members.loria.fr/vincent.thomas

[3] Univ Lyon, INSA Lyon, Inria, CITI
6 Avenue des Arts, F-69621 Villeurbanne
prenom.nom@inria.fr et http://dibangoye.fr

Résumé : Cet article s'intéresse aux MDP à états continus quand leurs fonctions de transition et de récompense sont lipschitziennes (par rapport à l'état). Il montre que, dans ce cadre, la fonction de valeur optimale à horizon fini est elle-même lipschitzienne, ce qui permet d'employer des représentations adaptées (en dents de scie : vers le bas pour un majorant et vers le haut pour un minorant) et de proposer des algorithmes de résolution à erreur bornée directement inspirés d'algorithmes de l'état de l'art. Cela permettra en particulier de résoudre des ρPOMDP (Araya-López *et al.*, 2010) – c'est-à-dire des POMDP dont la récompense dépend de l'état de croyance – même si la fonction de récompense n'est pas convexe dans l'espace des états de croyance. Un objectif à plus long terme est de pouvoir adopter des méthodes similaires pour résoudre des jeux de Markov partiellement observables (POSG). **Mots-clés** : MDP, état continu, POMDP, ρ-POMDP, Lipschitz, HSVI

1 Introduction

Nous nous intéressons au moyen de résoudre des MDP à états continus quand leurs fonctions de transition et de récompense sont relativement régulières (*smooth*), plus précisément quand elles sont lipschitziennes (par rapport à l'état). Cette situation se rencontre dans tout belief MDP utilisé dans la résolution d'un POMDP, cas dans lequel on préférera exploiter la convexité de la fonction de valeur. Dans le cas des ρPOMDP (Araya-López *et al.*, 2010) – c'est-à-dire des POMDP dont la récompense dépend de l'état de croyance – la convexité de la fonction de valeur n'est garantie que si la fonction de récompense est elle-même convexe. Par ailleurs, on pourra envisager dans certains cas d'approcher un problème à l'aide d'un modèle lipschitzien.

Comme on va le voir, sous l'hypothèse que la dynamique comme la fonction de récompense sont lipschitziennes (par rapport à l'état), la fonction de valeur optimale à horizon fini est, elle aussi, lipschitzienne, ce qui permet de l'approcher ou de l'encadrer en contrôlant l'erreur faite à l'aide d'approximateurs "en dents-de-scie coniques". On va ainsi pouvoir proposer des algorithmes de résolution à horizon temporel fini ou infini, et avec ou sans état initial. Dans chacun de ces cas, on suivra une démarche similaire à celle suivie dans de multiples solveurs de POMDP (exploitant, eux, la convexité de la fonction de valeur), mais en utilisant des approximateurs non-convexes.

Les sections 2 et 3 présentent respectivement des travaux connexes et un bref état de l'art sur les MDP, POMDP et ρPOMDP. La section 4 décrit les résultats de lipschitz-continuité de la fonction de valeur optimale obtenus, en proposant essentiellement des majorants de la constante de Lipschitz. Sur cette base, la section 5 explique comment approcher la fonction de valeur optimale V^* en exploitant cette propriété de Lipschitz, et comment adapter des algorithmes de l'état de l'art dans cette situation, tout en contrôlant l'erreur faite. Une discussion vient conclure sur les perspectives ouvertes par cette approche.

2 Travaux connexes

L'exploitation de la propriété de linéarité par morceaux et de convexité (PWLC) de la fonction de valeur dans les POMDP (Sondik, 1971; Smallwood & Sondik, 1973) est très comparable à l'exploitation de la Lipschitz-continuité que nous allons aborder ici. Cette dernière est une hypothèse moins forte, donc applicable dans un cadre plus large. Elle fournit des approximateurs moins efficaces (nécessitant une résolution plus fine pour obtenir la même précision), mais qui permettront quand même de contrôler l'erreur faite.

Plus proches du présent travail concernant la Lipschitz-continuité du problème, Ieong *et al.* (2007) ont considéré un cadre reposant sur les hypothèses suivantes :

— la dynamique du MDP est déterministe (ce qui ne sera qu'un cas particulier pour nous) ;
— l'espace des états et l'espace des actions sont continus ;
— le modèle de la dynamique (T) et le modèle de récompense (r) sont lipschitziens par rapport aux états comme aux actions ;
— une heuristique admissible lipschitzienne est disponible ;
— l'objectif est de trouver un chemin le plus court jusqu'à un état but en présence de culs-de-sac.

Sur cette base, ils proposent un algorithme de type "meilleur d'abord", lequel doit subir d'onéreuses mises à jour de ses minorants (dans un cadre de minimisation). En comparaison, nous (i) ignorons les actions continues, (ii) tenons compte de dynamiques stochastiques, (iii) nous attaquons à des problèmes à horizon fini ou infini, (iv) caractérisons la forme de la fonction de valeur, et (v) nous orientons plutôt vers des algorithmes reposant sur des générations de trajectoires. Comme argumenté par Smith & Simmons (2004), (a) de tels algorithmes en "profondeur d'abord" permettent un meilleur compromis entre consommation mémoire et temps de calcul, et (b) des algorithmes de type "meilleur d'abord" (comme celui de Ieong *et al.* (2007)) doivent propager des mises à jours des bornes dans leur file des priorités, ce qui est très coûteux.

3 Etat de l'art

Nous allons aborder deux cadres : d'une part celui des MDP (à espace d'état continu) dits s-lipschitziens, et d'autre part celui des ρ-POMDP. Comme on pourra l'observer, les résultats théoriques sont proches dans les deux cas.

3.1 MDP s-lipschitziens

Un MDP (Bellman, 1957) est ici défini par un tuple $\langle \mathcal{S}, \mathcal{A}, T, r \rangle$ dans lequel :

— $\mathcal{S}$ est un ensemble continu d'états, doté d'une métrique $d(\cdot, \cdot')$;
— $\mathcal{A}$ est un ensemble fini d'actions ;
— T est une fonction de transition soit déterministe (de sorte qu'appliquer une action a dans un état s amène toujours au même état $s' = T(s, a)$), soit stochastique (de sorte qu'appliquer une action a dans un état s amène à un état s échantillonné selon une distribution $T(s, a)$) ; et
— r est une fonction de récompense définie sur $\mathcal{S} \times \mathcal{A}$ et à valeurs dans $\mathbb{R}$; on supposera en outre r majorée (respectivement minorée) par une constante R^{max} (resp. R^{min}).

Dans le cas déterministe, le caractère s-lipschitzien des MDP vient du fait que les fonctions de transition et de récompense sont lipschitziennes par rapport à l'état, ce qui s'écrit, avec les constantes ρ et τ (pour tout s, s', a) :

$$|r(s, a) - r(s', a)| \leq \rho d(s, s'),$$
$$d\left(T(s, a), T(s', a)\right) \leq \tau d(s, s'),$$

où $d(\cdot, \cdot)$ est la métrique sur $\mathcal{S}$.

Dans le cas stochastique, il n'est pas possible de traiter la fonction de transition de la même manière, essentiellement parce qu'il est difficile de définir une métrique sur des distributions de probabilité. Nous faisons ici l'hypothèse (inspirée indirectement par Somani *et al.* (2013)) qu'il est possible d'écrire la dynamique stochastique du système avec une fonction de transition déterministe

$$T : \mathcal{S} \times \mathcal{A} \times \mathcal{X} \to \mathcal{S},$$

où $\mathcal{X}$ est le domaine d'une variable aléatoire X suivant une densité de probabilité conditionnelle $f(x|s, a)$. En d'autres termes, (i) la variable aléatoire X représente un événement dont la probabilité d'occurrence

dépend de s et a, et (ii) cet événement a une influence sur le prochain état qui dépend aussi de s et a. Nous verrons que ces deux propriétés sont aussi présentes dans les POMDP. Avec cette modélisation, le caractère s-lipschitzien de la fonction de transition passe par ces deux étapes ($\forall a \in \mathcal{A},\ x \in \mathcal{X},\ (s,s') \in \mathcal{S}^2$) :

$$|f(x|s,a) - f(x|s',a)| \leq \phi d(s,s'), \text{ et}$$
$$d\left(T(s,a,x), T(s',a,x)\right) \leq \tau d(s,s').$$

Dans tous les cas (déterministe ou stochastique), on cherche ici des politiques optimisant un critère total avec facteur d'atténuation γ et horizon temporel (éventuellement infini) H :

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{H} \gamma^t R_t\right].$$

On utilisera pour cela des algorithmes calculant tout ou partie de la fonction de valeur optimale obtenue à l'aide de l'opérateur de Bellman ($\forall s \in \mathcal{S}$) :

$$V_t^*(s) = 0, \text{ et}$$
$$V_{t-1}^*(s) = \max_a \left[r(s,a) + \gamma \int_{s'} P(s'|s,a) V_t^*(s') ds'\right] \quad (\forall t \in \{0,\dots,H-1\}).$$

Cette fonction de valeur sera couramment majorée par

$$V_t^{max} = \begin{cases} \frac{1-\gamma^{H-t}}{1-\gamma} R^{max} & \text{si } \gamma < 1; \\ (H-t)R^{max} & \text{si } \gamma = 1; \end{cases} \qquad (\forall t \in \{0,\dots,H\})$$

voire, pour éviter la dépendance en t,

$$V^{max} = \begin{cases} \frac{R^{max}}{1-\gamma} & \text{si } \gamma < 1; \\ HR^{max} & \text{si } \gamma = 1. \end{cases}$$

On définira de même les minorants V_t^{min} et V^{min}, mais aussi $V_t^{lim} = \max\{|V_t^{max}|, |V_t^{min}|\}$ et $V^{lim} = \max\{|V^{max}|, |V^{min}|\}$.

3.2 POMDP et ρ-POMDP

Nous rappelons d'abord la définition d'un POMDP avant de passer à celle, moins connue, d'un ρ-POMDP.

Un MDP partiellement observable (POMDP) (Astrom, 1965) est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, T, O, r\rangle$ où

— $\langle \mathcal{S}, \mathcal{A}, T, r\rangle$ est un MDP ;
— $\mathcal{Z}$ est un ensemble fini d'observations possibles ; et
— O est une fonction d'observation donnant $O(a,s',z) = Pr(z|a,s')$, la probabilité d'observer z si l'action a conduit dans l'état s'.

La recherche d'une politique optimale se fait souvent en passant par le MDP sur les états de croyance (bMDP) correspondant, dans lequel

— la connaissance actuelle de la situation est résumée par un état de croyance b, c'est-à-dire une distribution de probabilité sur les états possibles ;
— la fonction de transition $T(b,a,b')$ donne la probabilité d'aller d'un état de croyance b à un autre b' étant donnée une action a (en tenant compte des observations qui pourront être rencontrées) ; et
— la fonction de récompense donne la récompense moyenne étant donnée l'état de croyance : $r(b,a) = \sum_{s\in\mathcal{S}} b(s) r(s,a)$.

Pour des raisons pratiques, nous emploierons les mêmes notations T et r qu'il s'agisse des fonctions décrivant un MDP, un POMDP, ou son bMDP associé.

Araya-López *et al.* (2010) ont étendu le cadre des POMDP pour pouvoir traiter non seulement des problèmes de contrôle de l'état, mais aussi de recherche d'information. Pour cela, ils remplacent la donnée de $r(s,a)$ par une fonction de récompense définie sur l'état de croyance $r(b)$, par exemple liée à une mesure d'entropie. [1] Pour des raisons pratiques, nous ne notons pas cette fonction ρ comme les auteurs, réservant ρ pour dénoter la constante de Lipschitz associée à cette fonction le cas échéant. Dans ce contexte, Araya-López *et al.* ont montré que :

1. On notera que cette fonction de récompense peut aussi dépendre de s et/ou a.

— si r est linéaire par morceaux et convexe (PWLC) sur $\mathcal{B}$, le simplexe des états de croyance, alors la fonction de valeur est aussi PWLC, et de nombreux solveurs reposant sur cette propriété peuvent être exploités ;
— si r est convexe et λ-lipschitzienne ou α-hölderienne, alors on peut l'approcher arbitrairement bien par une fonction PWLC, et donc approcher la fonction de valeur arbitrairement bien.

Dans cet article, nous allons montrer comment se passer de la contrainte de convexité de r, tout en mettant en œuvre une approche assez similaire.

4 Lipschitz-continuité de V^*

Avant de présenter les résultats de Lipschitz-continuité de V^* dans différents cadres, la section suivante décrit quelques résultats préliminaires utiles.

4.1 Résultats préliminaires

Du fait de l'opérateur de Bellman utilisé pour les mises à jour de la fonction de valeur, dans les différents scénarios que nous allons étudier, nous rencontrerons des suites arithmético-géométriques, en l'occurrence de la forme $u_t = \alpha \cdot u_{t+1} + \beta$ initialisée par $u_H = 0$, pour t allant de H à 0, et avec $\alpha > 0$. On a alors le résultat suivant.

Lemme 1
On a, pour tout $t \in \{0, \cdots, H\}$,

$$u_t = \begin{cases} \frac{1-\alpha^{H-t}}{1-\alpha}\beta & \text{si } \alpha \neq 1, \text{ et} \\ (H-t)\beta & \text{si } \alpha = 1. \end{cases}$$

Démonstration 1
La preuve est immédiate en modifiant les indices de résultats classiques pour les suites arithmético-géométriques (et avec une initialisation à 0*).* [2] □

Par ailleurs, nous aurons besoin de la propriété suivante pour le produit de deux fonctions scalaires lipschitziennes.

Lemme 2
Soit X *un ensemble doté d'une métrique* d*, et soient* f *et* g *deux fonctions de* X *dans* $\mathbb{R}$*, lipschitziennes (de constantes respectives* k_f *et* k_g*). Alors, pour tout* $(x, x') \in X^2$,

$$|f(x) \cdot g(x) - f(x') \cdot g(x')| \leq (|g(x)| \cdot k_f + |f(x)| \cdot k_g) \cdot d(x, x').$$

Démonstration 2
Pour tout $(x, x') \in X^2$,

$$\begin{aligned} |f(x) \cdot g(x) - f(x') \cdot g(x')| &= |f(x) \cdot g(x) - f(x') \cdot g(x) + f(x') \cdot g(x) - f(x') \cdot g(x')| \\ &\leq |f(x) \cdot g(x) - f(x') \cdot g(x)| + |f(x') \cdot g(x) - f(x') \cdot g(x')| \\ &\leq |g(x) \cdot (f(x) - f(x'))| + |f(x') \cdot (g(x) - g(x'))| \\ &\leq |g(x)| \cdot |f(x) - f(x')| + |f(x')| \cdot |g(x) - g(x')| \\ &\leq |g(x)| \cdot k_f \cdot d(x, x') + |f(x)| \cdot k_g \cdot d(x, x') \\ &= (|g(x)| \cdot k_f + |f(x)| \cdot k_g) \cdot d(x, x'). \end{aligned}$$

□

En outre, si $|f|$ et $|g|$ sont majorées respectivement par F et $G \in \mathbb{R}$, alors

$$|f(x) \cdot g(x) - f(x') \cdot g(x')| \leq \underbrace{(G \cdot k_f + F \cdot k_g)}_{k_{f \cdot g}} \cdot d(x, x'),$$

donc la fonction $f \cdot g$ est lipschitzienne.

2. https://fr.wikipedia.org/wiki/Suite_arithm\%C3\%A9tico-g\%C3\%A9om\%C3\%A9trique

4.2 MDP s-lipschitziens, cas déterministe

Dans le cas des MDP s-lipschitziens à dynamique déterministe, on a le résultat suivant.

Lemme 3
Sur un horizon fini H, la fonction de valeur optimale vérifie, $\forall t \in \{0, \dots, H\}$, $\forall s, s' \in S$,

$$|V_t^*(s) - V_t^*(s')| \leq \nu_t^{max} d(s, s'),$$
$$\text{où } \nu_t^{max} = \begin{cases} \rho \frac{1-(\gamma\tau)^{H-t}}{1-\gamma\tau} & \text{si } \gamma\tau \neq 1, \\ \rho(H-t) & \text{si } \gamma\tau = 1. \end{cases}$$

Démonstration 3
Pour $t = H$, on a, pour tout (s, s'), $|V_H^*(s) - V_H^*(s')| = |0 - 0| = 0$, donc la propriété est vérifiée.
Supposons maintenant que la propriété est vérifiée pour $t \in \{1, \dots, H\}$. Alors, pour tout (s, s'),

$$\begin{aligned} V_{t-1}^*(s) &= \max_a \left[r(s,a) + \gamma V_t^*(T(s,a)) \right] \\ &\leq \max_a \Big[\underbrace{\left(r(s',a) + \rho d(s,s') \right)}_{r \text{ lipschitz.}} + \gamma \underbrace{\left(V_t^*(T(s',a)) + \nu_t^{max} \tau d(s,s') \right)}_{V_t^* \text{ et } T \text{ lipschitz.}} \Big] \\ &= \underbrace{\max_a \left[r(s',a) + \gamma V_t^*(T(s',a)) \right]}_{V_{t-1}^*(s')} + \left(\rho d(s,s') + \gamma \nu_t^{max} \tau d(s,s') \right), \end{aligned}$$

d'où

$$V_{t-1}^*(s) - V_{t-1}^*(s') \leq \rho d(s,s') + \gamma \nu_t^{max} \tau d(s,s') = \underbrace{\left(\gamma\tau \nu_t^{max} + \rho \right)}_{\nu_{t-1}^{max}} d(s,s').$$

On peut alors appliquer le lemme 1 avec $u_t = \nu_t^{max}$, $\alpha = \gamma\tau$ et $\beta = \rho$ pour obtenir le résultat escompté pour tout t. □

Notons que, quand l'horizon H croît, la constante de Lipschitz pour $t = 0$ ainsi obtenue tend vers :

$$\begin{cases} \rho \frac{1}{1-\gamma\tau} & \text{si } \gamma\tau < 1, \\ +\infty & \text{sinon (linéairement ssi } \gamma\tau = 1). \end{cases}$$

Autrement dit, le facteur d'atténuation γ peut compenser l'expansion de l'espace d'état due à la dynamique du système. A horizon infini, si $\gamma\tau \geq 1$, la fonction de valeur optimale n'est pas nécessairement lipschitzienne, mais il est possible de se ramener à un horizon fini en cherchant une solution ϵ-optimale. A ce propos, on remarquera que, même dans des problèmes de chemin le plus court déterministes, le nombre minimum de pas de temps pour atteindre le but n'est pas majoré, à moins que :

— l'ensemble des états soit fini (donc non continu), ou
— l'ensemble des actions garantisse que l'ensemble des états *atteignables* soit fini.

Cela pourrait expliquer que Ieong *et al.* (2007) ne font pas l'hypothèse que la fonction de valeur est lipschitzienne et ne cherchent pas à majorer la constante associée, mais se reposent sur l'existence d'une fonction heuristique (admissible) qui soit lipschitzienne. [3]

4.3 MDP s-lipschitziens, cas stochastique

Dans le cas des MDP s-lipschitziens à dynamique stochastique, des constantes de Lipschitz peuvent être calculées de manière récursive, comme décrit par le lemme suivant.

Lemme 4
Supposons que $\mathcal{X}$ soit borné, donc ait un volume fini $vol(\mathcal{X})$. Alors, sur un horizon fini H, la fonction de valeur optimale est ν_t-lipschitzienne pour tout $t \in \{0, \dots, H\}$, où

$$\nu_H = 0, \quad \text{et, pour } t \in \{1, \dots, H\},$$
$$\nu_{t-1} = (\gamma\tau)\nu_t + \left(\rho + \gamma V_t^{lim} \phi vol(\mathcal{X}) \right).$$

3. On observera qu'il n'est pas nécessaire que la fonction de valeur optimale soit lipschitzienne pour qu'elle puisse être minorée ou majorée par une fonction lipschitzienne.

Démonstration 4
Pour $t = H$, on a, pour tout (s, s'), $|V_H^*(s) - V_H^*(s')| = |0 - 0| = 0$, donc la propriété est vérifiée.
Supposons que la propriété soit vérifiée pour $t \in \{1, \dots, H\}$. Alors, pour tout (s, s'),

$$V_{t-1}^*(s) = \max_a \left[r(s,a) + \gamma \int_{x \in \mathcal{X}} V_t^*(T(s,a,x)) f(x|s,a) dx \right]$$

Or, V_t, T, et f étant lipschitziennes, en appliquant le lemme 2 et en utilisant le fait que la composition de fonctions lipschitziennes est lipschitzienne,

$$\begin{aligned} & V_t^*(T(s,a,x)) f(x|s,a) \\ & \leq V_t^*(T(s',a,x)) f(x|s',a) + \Big(\underbrace{|f(x|s',a)|}_{\leq 1 \text{ après intégration}} \cdot \nu_t \tau + \underbrace{|V_t^*(T(s',a,x))|}_{\leq V_t^{lim}} \cdot \phi \Big) \cdot d(s,s'). \end{aligned}$$

D'où,

$$\begin{aligned} & V_{t-1}^*(s) \\ & \leq \max_a \Big[\underbrace{(r(s',a) + \rho d(s,s'))}_{r \text{ lipschitz.}} \\ & \qquad + \gamma \int_{x \in \mathcal{X}} \left[V_t^*(T(s',a,x)) f(x|s',a) + \left(f(x|s',a) \cdot \nu_t \tau + V_t^{lim} \cdot \phi \right) \cdot d(s,s') \right] dx \Big] \\ & \leq \max_a \Big[(r(s',a) + \rho d(s,s')) + \gamma \Big[\int_{x \in \mathcal{X}} V_t^*(T(s',a,x)) f(x|s',a) dx \\ & \qquad + \nu_t \tau d(s,s') \underbrace{\int_{x \in \mathcal{X}} f(x|s',a) dx}_{=1} + V_t^{lim} \phi d(s,s') \underbrace{\int_{x \in \mathcal{X}} 1 dx}_{=vol(\mathcal{X})} \Big] \Big] \\ & = \max_a \left[r(s',a) + \gamma \int_{x \in \mathcal{X}} V_t^*(T(s',a,x)) f(x|s',a) dx \right] \\ & \qquad + \left(\rho d(s,s') + \gamma \left[\nu_t \tau d(s,s') + V_t^{lim} \phi d(s,s') vol(\mathcal{X}) \right] \right) \\ & = V_{t-1}^*(s') + \left(\rho + \gamma \left[\tau \nu_t + V_t^{lim} \phi vol(\mathcal{X}) \right] \right) d(s,s') \\ & = V_{t-1}^*(s') + \underbrace{\left((\gamma\tau) \nu_t + \left(\rho + \gamma V_t^{lim} \phi vol(\mathcal{X}) \right) \right)}_{\nu_{t-1}} d(s,s'). \end{aligned}$$

On a donc bien la formule de récurrence attendue. □

Cette suite de constantes de Lipschitz est exploitable dans les algorithmes, mais il est difficile de déterminer leur évolution quand l'horizon croît. En faisant des approximations plus grossières, on peut majorer cette suite par une autre suite de constantes de Lipschitz dont on peut obtenir une expression non récursive, comme détaillé dans le lemme suivant.

Lemme 5
Supposons que $\mathcal{X}$ soit borné, donc ait un volume fini $vol(\mathcal{X})$. Alors, sur un horizon fini H, la fonction de valeur optimale vérifie, $\forall t \in \{0, \dots, H\}$, $\forall s, s' \in S$,

$$|V_t^*(s) - V_t^*(s')| \leq \nu_t^{max} d(s,s'),$$

$$\text{où } \nu_t^{max} = \begin{cases} \left(\rho + \gamma V^{lim} \phi vol(\mathcal{X}) \right) \dfrac{1 - (\gamma\tau)^{H-t}}{1 - \gamma\tau} & \text{si } \gamma\tau \neq 1, \\ \left(\rho + \gamma V^{lim} \phi vol(\mathcal{X}) \right) (H - t) & \text{si } \gamma\tau = 1. \end{cases}$$

Démonstration 5
Comme précédemment, la propriété est trivialement vérifiée pour $t = H$.
En remplaçant, pour tout $t \in \{1, \dots, H\}$, V_t^{lim} par $V^{lim} (> V_t^{lim})$ dans l'expression reliant ν_{t-1} à ν_t, on construit une suite $(\nu_t^{max})_{t \in \{0,\dots,H\}}$ vérifiant, pour tout $t \in \{1, \dots, H\}$,

$$\nu_{t-1}^{max} = \underbrace{(\gamma\tau)}_{\alpha} \nu_t^{max} + \underbrace{\left(\rho + \gamma V^{lim} \phi vol(\mathcal{X}) \right)}_{\beta}.$$

On peut alors appliquer le lemme 1 pour obtenir le résultat escompté pour tout t. □

On peut remarquer à propos de la suite de constantes de Lipschitz ainsi obtenue :
— qu'elle diverge ici dès que $\gamma\tau \geq 1$;
— qu'en prenant $\phi = 0$ (indépendance de la première "étape" par rapport à s), on retrouve l'expression obtenue dans le cas déterministe) ;
— qu'elle dépend du volume (fini) de $\mathcal{X}$; et
— que $vol(\mathcal{X})$ et ϕ sont liés puisqu'on pourrait ré-écrire le même modèle de dynamique en effectuant des changements d'échelle ; on pourrait ainsi toujours se ramener à $vol(\mathcal{X}) = 1$.

4.4 POMDP et ρ-POMDP

Partant du fait que la fonction de valeur d'un POMDP à horizon fini est linéaire par morceaux et convexe (PWLC), il est évident qu'elle est aussi lipschitzienne. Nous allons ici encadrer la constante de Lipschitz associée à chaque horizon, et ce dans le cas plus général des ρ-POMDP avec une fonction de récompense r qui est lipschitzienne de constante ρ.

Comme précédemment, on va chercher à encadrer l'écart entre la fonction de valeur optimale en un état de croyance b_1 et un autre b_2. Nous allons pour cela avoir besoin des résultats préliminaires suivants. [4]

Lemme 6
Etant donnés deux états de croyance b_1 et b_2, une action effectuée a et une observation reçue z, on a :

$$\begin{aligned}\|b_1^{a,z} - b_2^{a,z}\|_\infty &\leq \lambda_{a,z}\|b_1 - b_2\|_\infty \\ \text{où } \lambda_{a,z} &= \max_{s'} \sum_s \frac{P(z|a,s')P(s'|s,a)}{\sum_{s"} P(z|a,s")P(s"|s,a)} \\ &= \max_{s'} \sum_s \frac{P(z,s'|s,a)}{\sum_{s"} P(z,s"|s,a)} \\ &= \max_{s'} \sum_s P(s'|s,a,z).\end{aligned}$$

Démonstration 6
Rappelons d'abord le calcul de la mise à jour d'un état de croyance pour une paire action-observation (a, z) :

$$\begin{aligned}\forall s', \quad b^{a,z}(s') = P(s'|b,a,z) &= \sum_s P(s'|s,a,z)b(s) \\ &= \sum_s \frac{P(z|a,s')P(s'|s,a)}{\sum_{s"} P(z|a,s")P(s"|s,a)} b(s).\end{aligned}$$

De là, on tire :

$$\begin{aligned}\forall s', \quad b_1^{a,z}(s') - b_2^{a,z}(s') &= \sum_s \frac{P(z|a,s')P(s'|s,a)}{\sum_{s"} P(z|a,s")P(s"|s,a)} [b_1(s) - b_2(s)] \\ &\leq \left[\sum_s \frac{P(z|a,s')P(s'|s,a)}{\sum_{s"} P(z|a,s")P(s"|s,a)}\right] \|b_1 - b_2\|_\infty \\ &\leq \max_{s'} \left[\sum_s \frac{P(z|a,s')P(s'|s,a)}{\sum_{s"} P(z|a,s")P(s"|s,a)}\right] \|b_1 - b_2\|_\infty.\end{aligned}$$

D'où le résultat attendu. □

En pratique, on utilisera comme constante de Lipschitz $\lambda = \max_{a,z} \lambda_{a,z}$.

4. Dans le cadre des POMDP, Platzman (1977) montre un résultat équivalent au lemme 6, mais pour une distance entre états de croyance différente, et avec une constante de Lipschitz inférieure ou égale à 1. Une question ouverte est donc de savoir si, en démontrant le même résultat pour la norme infinie ou en utilisant la même distance que Platzman, on ne pourrait pas obtenir de meilleurs encadrements de la fonction de valeur optimale que dans le présent article.

Lemme 7
Etant donnés deux états de croyance b_1 et b_2, une action effectuée a et une observation reçue z, on a :

$$|P(z|b_1,a) - P(z|b_2,a)| \leq \mu_{a,z}\|b_1 - b_2\|_\infty$$
$$\text{où} \quad \mu_{a,z} = \sum_s P(z|a,s) = \sum_{s,s'} P(z|a,s')T(s,a,s').$$

Démonstration 7
Le calcul de la probabilité d'une observation z pour un état de croyance b et une action a donne :

$$P(z|b,a) = \sum_s P(z|s,a)b(s) = \sum_{s,s'} P(z|a,s')T(s,a,s')b(s).$$

De là, on tire :

$$\begin{aligned} P(z|b_1,a) - P(z|b_2,a) &= \sum_{s,s'} P(z|a,s')T(s,a,s')\,[b_1(s) - b_2(s)] \\ &\leq \Big[\sum_{s,s'} P(z|a,s')T(s,a,s')\Big]\|b_1 - b_2\|_\infty. \end{aligned}$$

D'où le résultat attendu. □

En pratique, on utilisera comme constante de Lipschitz $\mu = \max_{a,z} \mu_{a,z}$.

Des deux lemmes précédents, on tire le corollaire suivant (où la norme utilisée est toujours la norme infinie).

Corollaire 1
Sur un horizon fini H, la fonction de valeur optimale est ν_t-lipschitzienne pour tout $t \in \{0,\ldots,H\}$, où

$$\nu_H = 0, \quad \textit{et pour } t \in \{1,\ldots,H\},$$
$$\nu_{t-1} = (\gamma\lambda)\nu_t + \left(\rho + \gamma|\mathcal{Z}|V^{lim}\mu\right).$$

Démonstration 8
Pour $t = H$, on a trivialement $|V_H^*(b_1) - V_H^*(b_2)| = 0$, donc la propriété est vérifiée.

Supposons maintenant que la propriété est vérifiée pour $t \in \{1,\ldots,H\}$. On a alors, pour tout (b_1,b_2) :

$$V_{t-1}^*(b_1) = \max_a \Big[r(b_1,a) + \gamma \sum_z P(z|b_1,a)V_t^*(b_1^{a,z})\Big].$$

Or, V_t étant lipschitzienne, avec les lemmes 6 et 7, et en appliquant le lemme 2 :

$$P(z|b_1,a)\cdot V_t^*(b_1^{a,z}) \leq P(z|b_2,a)\cdot V_t^*(b_2^{a,z}) + \Big(\underbrace{|V_t^*(b_1^{a,z})|}_{\leq V^{lim}}\cdot\mu + |P(z|b_1,a)|\cdot\nu_t\lambda\Big)\|b_1 - b_2\|.$$

D'où,

$$\begin{aligned} &V_{t-1}^*(b_1) \\ &\leq \max_a \Big[r(b_2,a) + \rho\|b_1 - b_2\| + \gamma\sum_z \left(P(z|b_2,a)V_t^*(b_2^{a,z}) + \left(V_t^{lim}\mu + P(z|b_1,a)\nu_t\lambda\right)\|b_1 - b_2\|\right)\Big] \\ &\leq \max_a \Big[r(b_2,a) + \gamma\sum_z \left(P(z|b_2,a)V_t^*(b_2^{a,z})\right) \\ &\qquad + \rho\|b_1 - b_2\| + \gamma\sum_z \left(V_t^{lim}\mu\right)\|b_1 - b_2\| + \gamma\underbrace{\sum_z P(z|b_1,a)}_{=1}\nu_t\lambda\|b_1 - b_2\|\Big] \\ &\leq \max_a \Big[r(b_2,a) + \gamma\sum_z \left(P(z|b_2,a)V_t^*(b_2^{a,z})\right) + \left(\rho + \gamma|\mathcal{Z}|V_t^{lim}\mu + \gamma\nu_t\lambda\right)\|b_1 - b_2\|\Big] \\ &\leq V_{t-1}^*(b_2) + \underbrace{\left((\gamma\lambda)\nu_t + \left(\rho + \gamma|\mathcal{Z}|V_t^{lim}\mu\right)\right)}_{\nu_t}\|b_1 - b_2\|. \end{aligned}$$

On a donc bien la formule de récurrence attendue. □

TABLE 1 – Analogies entre MDP s-lipschitziens stochastiques et (ρ)-POMDP

MDP	(ρ-)POMDP
τ	λ
ϕ	μ
$vol(\mathcal{X})$	$\lvert\mathcal{Z}\rvert$

Comme dans le cas des MDP stochastiques, on peut dériver une suite de constantes de Lipschitz plus grossière mais dont on peut obtenir une expression non récursive.

Corollaire 2
Sur un horizon fini H, la fonction de valeur optimale vérifie, $\forall b_1, b_2$, $\forall t \in \{0, \dots, H\}$,

$$|V_t^*(b_1) - V_t^*(b_2)| \leq \nu_t^{max} \|b_1 - b_2\|,$$
$$\text{où } \nu_t^{max} = \begin{cases} \left(\rho + \gamma V^{lim} \mu |\mathcal{Z}|\right) \cdot \dfrac{1 - (\gamma\lambda)^{H-t}}{1 - (\gamma\lambda)} & \text{si } \gamma\lambda \neq 1; \\ \left(\rho + \gamma V^{lim} \mu |\mathcal{Z}|\right) \cdot (H - t) & \text{si } \gamma\lambda = 1. \end{cases}$$

Démonstration 9
Comme précédemment, la propriété est trivialement vérifiée pour $t = H$.

En remplaçant, pour tout $t \in \{1, \dots, H\}$, V_t^{lim} par $V^{lim} (> V_t^{lim})$ dans l'expression reliant ν_{t-1} à ν_t, on construit une suite $(\nu_t^{max})_{t \in \{0,\dots,H\}}$ vérifiant, pour tout $t \in \{1, \dots, H\}$,

$$\nu_{t-1}^{max} = \Big(\underbrace{(\gamma\lambda)}_{\alpha} \nu_t^{max} + \underbrace{\left(\rho + \gamma |\mathcal{Z}| V^{lim} \mu\right)}_{\beta} \Big) \|b_1 - b_2\|.$$

On peut alors appliquer le lemme 1 pour obtenir le résultat escompté pour tout t.

Comme on pouvait s'y attendre, on a les analogies présentées dans la table 1 avec le cas des MDP s-lipschitziens stochastiques.

5 Algorithmes

Nous allons maintenant discuter de différents algorithmes possibles selon que l'horizon est fini ou non, et qu'un état de croyance initial est fourni ou non. Les résultats sont présentés dans le cadre des MDP continus stochastiques, mais transposables directement aux (ρ)POMDP. On fait toutefois l'hypothèse que le nombre d'états atteignables suite à l'application d'une action est fini et borné, de manière à remplacer les intégrales par des sommes finies.

5.1 Préliminaire : Approximation d'une fonction lipschitzienne

Une étape préliminaire est l'approximation d'une simple fonction k-lipschitzienne $f : X \to \mathbb{R}$ à erreur bornée $\epsilon > 0$ ($k \in \mathbb{R}^+$). Soit X' un ensemble de N points de X. Si la valeur de f n'est connue qu'en les points de X', alors pour tout $x \in X$, on a :

$$\underbrace{\max_{x' \in X'} \left[f(x') - k \cdot d(x, x')\right]}_{L_f^{X'}(x)} \leq f(x) \leq \underbrace{\min_{x' \in X'} \left[f(x') + k \cdot d(x, x')\right]}_{U_f^{X'}(x)}.$$

Désormais, on considérera essentiellement les deux types d'approximateurs (minorant et majorant) que sont $L_f^{X'}$ et $U_f^{X'}$. La figure 1 ne correspond pas tout à fait à cette situation car d'une part les valeurs exactes de référence $f(x)$ y sont remplacées par des valeurs majorantes (pour $U_f^{X'}$) ou minorantes (pour $L_f^{X'}$), et d'autre part des hyperplans constants sont aussi employés.

A l'évidence, pour que l'incertitude sur la valeur de f soit bornée, il faut que la distance maximale entre tout point de X et l'ensemble X' soit majorée : $\max_{x \in X} \min_{x' \in X'} d(x, x') \leq \delta (\in \mathbb{R}^+)$. En le point x

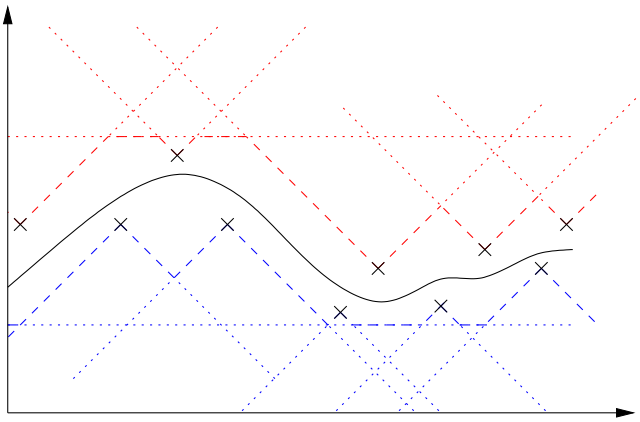


FIGURE 1 – Une fonction de valeur (mono-dimensionnelle) hypothétique et ses approximations en dents de scie (plus un hyperplan constant) majorantes (en rouge) et minorante (en bleu)

maximisant cette distance, l'incertitude $U_f^{X'}(x) - L_f X'(x)$ est majorée par $2k\delta$. L'existence de δ implique que le domaine X lui-même soit borné ($\max_{x,x'} d(x,x') \leq M(\in \mathbb{R}^+)$) pour qu'évaluer la fonction en un nombre fini de points suffise à obtenir une approximation bornée.

Etant donné X et une erreur souhaitée ϵ, le nombre minimum de points à sélectionner dans X' est le nombre de couverture de X pour ϵ, c'est à dire le nombre minimum de boules de rayons $\frac{\epsilon}{2k}$ (et ici de centres dans X) nécessaire pour couvrir X. La condition d'appartenance des centres à X pouvant compliquer la construction d'une telle couverture minimale, nous supposerons X convexe, hypothèse toujours vérifiée dans les bMDP. On peut espérer générer moins de points en construisant un ensemble X' de manière incrémentale, comme suit (on notera $X'_i = \{x_1, \ldots, x_i\}$),

— en prenant un premier point x_1 au hasard (ou minimisant $\max_{x \in X} d(x_1, x)$),

— puis en ajoutant tout nouvel x_{i+1} de manière à minimiser $U_f^{X'_i}(x_{i+1}) - L_f^{X'_i}(x_{i+1})$.

Cette méthode requiert toutefois de résoudre de multiples problèmes d'optimisation sur des ensembles continus, ce qui peut s'avérer coûteux.

Une approche inspirée de Munos (2011, 2014) consisterait à utiliser un partitionnement hiérarchique ou, en d'autres termes, un partitionnement de plus en plus fin de X, partitionnement dans lequel chaque sous-ensemble est associé à de simples majorants et minorant de f déductibles de la taille de ce sous-ensemble et de la constante de Lipschitz. En exploitant la capacité à calculer des constantes de Lipschitz locales, on peut obtenir un partitionnement plus ou moins fin selon la région de X. Par contre on n'échappe pas à la nécessité d'avoir une approximation aussi bonne sur tout X, alors que Munos (2011) n'a besoin que de garantir qu'une solution ϵ-optimale de f est trouvée.

5.2 Opérateur de Bellman

L'opérateur (d'optimalité) de Bellman, noté ici $\mathcal{H}$, met normalement à jour la fonction de valeur pour tout état (que l'horizon soit fini ou infini), ce qui ne sera pas possible en pratique dans nos cadres continus. Les familles d'approximateurs employées ici (minorant ou majorant) vont toutefois nous permettre à la place de mettre à jour la fonction de valeur en un ensemble de points à la fois (ceux de $\mathcal{S}'$) – opérateur $\mathcal{H}^{\mathcal{S}'}$ – ou en un seul point s – opérateur $\mathcal{H}^s$, mises à jour qui auront un impact sur toute une région de $\mathcal{S}$.

5.3 Programmation dynamique

Supposons d'abord qu'aucun état (de croyance) initial n'est défini, et que l'horizon temporel est fini.

La section précédente nous fournit deux types d'approximateurs à base d'un ensemble de points d'évaluation, l'un minorant, l'autre majorant.

— Si on choisit d'être pessimiste, on utilisera des approximateurs minorants L_t que l'on complétera par le minorant toujours valable V_t^{min}. Si, pour t donné, on note $\mathcal{S}'_t$ l'ensemble des points utilisés dans L_t, et $l_t(s)$ la valeur associée à un de ces points, alors $L_t(s) \stackrel{\text{def}}{=} \max\left\{V_t^{min}, \max_{s' \in S_t} \left(l_t(s') - \nu_t d(s,s')\right)\right\}$.

— Si on choisit d'être optimiste, on utilisera des approximateurs majorants U_t que l'on complétera par le majorant toujours valable V_t^{max}. Si, pour t donné, on note $\mathcal{S}'_t$ l'ensemble des points utilisés dans L_t, et $u_t(s)$ la valeur associée à un de ces points, alors $U_t(s) \stackrel{\text{def}}{=} \min\{V_t^{max}, \min_{s' \in S_t}(u_t(s') + \nu_t d(s,s'))\}$.

La figure 1 illustre la fonction de valeur optimale et ces deux fonctions à une étape t, en illustrant le fait que les valeurs de référence $l_t(s)$ (respectivement $u_t(s)$) pour $s \in \mathcal{S}'_t$ ne sont pas les valeurs exactes $V_t^*(s)$, mais des minorants (resp. des majorants). Par la suite, sauf besoin particulier, on notera simplement l'approximateur utilisé V_t (qu'on suive une approche optimiste ou pessimiste), et la valeur en un point s de $\mathcal{S}'_t$ sera $v_t(s)$.

Pour $t = H$, on a trivialement $\mathcal{S}'_H = \emptyset$. Comme $V_H^{min} = V_H^{max} = 0$, l'approximateur V_H est toujours exact. Pour t de $H-1$ à 0, on construit les approximateurs V_t successivement en calculant la valeur $v_t(s)$ de chaque point de l'ensemble $\mathcal{S}'_t$ par application de l'opérateur de Bellman $\mathcal{H}^s$ sur l'approximateur V_{t+1} défini sur l'ensemble de l'espace d'état :

$$v_t(s) \stackrel{\text{def}}{=} \max_a \Big(r(s,a) + \gamma \sum_{s'} T(s,a,s') V_{t+1}(s') \Big).$$

Cette approche est similaire à celle employée pour les POMDP pour encadrer la fonction de valeur PWLC à t par une enveloppe d'hyperplans (comme minorant) et un approximation en dents de scies (comme majorant). Comme pour les POMDP, éliminer régulièrement les éléments inutiles des approximateurs permettra de gagner du temps de calcul (et de la mémoire). Ici, on éliminera des points des ensembles $S_{U,t}$ et $S_{L,t}$ s'ils sont *inutiles*, c'est-à-dire :

$$(s, u_t(s)) \text{ est inutile si } \begin{cases} u_t(s) \geq V_t^{max} \text{ ou} \\ \exists s' \in S_{U,t} \setminus \{s\} \text{ tel que } u_t(s) \geq u_t(s') + \nu_t d(s,s'); \end{cases}$$

$$(s, l_t(s)) \text{ est inutile si } \begin{cases} l_t(s) \leq V_t^{min} \text{ ou} \\ \exists s' \in S_{L,t} \setminus \{s\} \text{ tel que } l_t(s) \leq l_t(s') - \nu_t d(s,s'). \end{cases}$$

Le choix des points à mettre dans $\mathcal{S}'_t$ reste une difficulté. Comparé à l'approximation d'une fonction simple, on va ici cumuler des erreurs d'approximation. Supposons que, quelle que soit la méthode de sélection des ensembles $\mathcal{S}'_t$ choisie, elle garantisse que l'erreur de l'approximateur est au plus de $\epsilon' > 0$ (indépendamment des erreurs cumulées aux itérations précédentes). On a alors le résultat suivant.

Théorème 1 (∼Théorême 3.1 de Pineau *et al.* (2006))
Pour tout t, l'erreur de la programmation dynamique $\epsilon_t = \|V_t - V_t^*\|_\infty$ est majorée par

$$\epsilon_t \leq \frac{1-\gamma^{H-t}}{1-\gamma} \epsilon'.$$

Démonstration 10

$$\begin{aligned}
\epsilon_t &= \|V_t - V_t^*\|_\infty \\
&= \|\mathcal{H}^{\mathcal{S}'_t} V_{t+1} - \mathcal{H} V_{t+1}^*\|_\infty && \textit{(par définition de } \mathcal{H}^{\mathcal{S}'_t}\textit{)} \\
&\leq \|\mathcal{H}^{\mathcal{S}'_t} V_{t+1} - \mathcal{H} V_{t+1}\|_\infty + \|\mathcal{H} V_{t+1} - \mathcal{H} V_{t+1}^*\|_\infty && \textit{(par inégalité triangulaire)} \\
&\leq \epsilon' + \|\mathcal{H} V_{t+1} - \mathcal{H} V_{t+1}^*\|_\infty && \textit{(par hypothèse)} \\
&\leq \epsilon' + \gamma \|V_{t+1} - V_{t+1}^*\|_\infty && \textit{(par contraction de la mise à jour exacte)} \\
&= \epsilon' + \gamma \epsilon_{t+1} && \textit{(par définition de } \epsilon_{t+1}\textit{)} \\
&\leq \frac{1-\gamma^{H-t}}{1-\gamma} \epsilon'. && \textit{(par sommation d'une série géométrique)}
\end{aligned}$$

□

En notant ν le maximum de ν_t sur $t \in \{0, \ldots, H\}$ et en prenant par exemple des ensembles $\mathcal{S}'_t$ formant une δ-couverture de $\mathcal{S}$, on a $\epsilon' \leq \delta\nu$ et $\epsilon_h \leq \frac{1-\gamma^{H-t}}{1-\gamma}\delta\nu$.

Pour obtenir une erreur ϵ_0 donnée, ce qui est souvent le critère premier, on peut donc employer $\epsilon' = \left(\frac{1-\gamma^H}{1-\gamma}\right)^{-1} \epsilon_0$ à chaque étape (pour chaque $t \in \{0, \ldots, H-1\}$). Si on construit à l'avance les ensembles de point $\mathcal{S}'_t$ (et non incrémentalement), alors le même ensemble $\mathcal{S}'$ peut être repris pour tout t.

Approximation d'un horizon temporel infini

Dans le cas d'un problème à horizon temporel infini, en l'absence de garantie que la suite de constantes de Lipschitz (ν_t) converge, il est préférable de se ramener à un problème à horizon fini. Pour garantir une erreur inférieure à $\epsilon > 0$, on peut choisir ϵ' et H par exemple de sorte que

$$\epsilon' = \left(\frac{1-\gamma^H}{1-\gamma}\right)^{-1} \frac{\epsilon}{2}$$

$$\text{et} \qquad \gamma^H \frac{R_{max}}{1-\gamma} \leq \frac{\epsilon}{2} \qquad \text{(pour que ce qui se passe au-delà de } H \text{ soit négligeable)}$$

$$\text{c'est-à-dire} \qquad \gamma^H \leq \frac{\epsilon}{2}\frac{1-\gamma}{R_{max}}$$

$$exp\left(H \ln \gamma\right) \leq exp\left(\ln\left(\frac{\epsilon}{2}\frac{1-\gamma}{R_{max}}\right)\right)$$

$$H = \left\lceil \frac{\ln\left(\frac{\epsilon}{2}\frac{1-\gamma}{R_{max}}\right)}{\ln \gamma} \right\rceil .$$

5.4 Itération sur la valeur

Approcher un problème à horizon infini en tronquant l'horizon peut être très coûteux en mémoire si l'horizon employé est grand. Si la suite des constantes de Lipschitz (ν_t) converge, on peut alors préférer employer un algorithme d'itération sur la valeur, par exemple synchrone. On indicera alors les fonctions de valeur par h pour désigner l'*horizon* qui a été considéré jusque là.

L'algorithme d'itération sur la valeur revient alors, en ayant par exemple choisi une δ-couverture $\mathcal{S}'$ de $\mathcal{S}$, à initialiser V_0, puis à calculer itérativement $V_h = \mathcal{H}^{cS'} V_{h-1}$ pour tout $h \geq 1$. Les bornes d'erreurs vues précédemment pour la programmation dynamique s'étendent de manière immédiate.

5.5 HSVI

Les algorithmes présentés jusqu'ici ont deux défauts importants :

— ils requièrent de construire des ensembles de points $\mathcal{S}'$ pour couvrir suffisamment bien l'espace d'états, ce qui implique soit une répartition régulière d'un grand nombre de points, soit une construction incrémentale (avec des optimisations continues à chaque étape) d'un nombre un peu moins important de points ; et

— ils n'exploitent pas la connaissance possible d'un état (de croyance) initial, ce qui permettrait de concentrer les efforts de calcul sur les parties atteignables de l'espace d'état.

Nous allons ici proposer une adaptation de l'algorithme *Heuristic Search Value Iteration* (HSVI) de Smith & Simmons (2004, 2005); Smith (2007), laquelle va permettre de pallier (pour partie) ces défauts. Evidemment, d'autres algorithmes à base de points (PBVI (Pineau *et al.*, 2003), PERSEUS (Spaan & Vlassis, 2005), FSVI (Shani *et al.*, 2007), SARSOP (Kurniawati *et al.*, 2008), GapMin (Poupart *et al.*, 2011) ...) pourraient être envisagés. Nous optons pour HSVI entre autres parce qu'il est relativement simple et permet de contrôler l'erreur faite (du fait de l'utilisation conjointe d'un minorant et d'un majorant).

Comme dans la version originale, nous considérons ici le cas d'un horizon temporel infini [5]. Le problème est caractérisé par un état initial s_0 et une erreur maximum souhaitée en s_0 bornée par $\epsilon > 0$.

5.5.1 Approximateurs uniformément améliorables

On remarquera d'abord que, avec une initialisation minorante (pour L_0) ou majorante (pour U_0), les approximateurs employés ici sont en outre *uniformément améliorables* (Zhang & Zhang, 2001; Smith, 2007), c'est-à-dire que, pour tout h :

$$L_h \leq \mathcal{H}L_h \leq V^* \leq \mathcal{H}U_h \leq U_h,$$

5. Il est toutefois possible de déterminer une politique à horizon temporel fini comme cela a été fait pour des DecPOMDP (Dibangoye *et al.*, 2013, 2016).

ce qui implique aussi :

$$L_h \leq L_{h+1} \leq V^* \leq U_{h+1} \leq U_h.$$

Ainsi, chaque itération ne peut faire qu'améliorer l'approximation de V^*.

Nous partons des travaux de Hauskrecht (1997, 2000); Smith (2007) pour chercher des initialisations possibles L_0 et U_0 plus fines que V^{min} et V^{max} dans le cas de ρ-POMDP non convexes. Pour L_0, faire une recherche directe de politique dans un espace de politiques restreint (par exemple celui des politiques aveugles ou à action fixe) et évaluer la politique trouvée est une méthode applicable. L'évaluation de la politique sera toutefois plus coûteuse que dans le cas d'un POMDP (avec les enveloppes d'alpha-vecteurs). Pour U_0, une approche courante est de mettre en œuvre un algorithme d'itération sur la valeur avec un opérateur "optimiste" par rapport à l'opérateur de Bellman. Toutefois, les méthodes "MDP", "QMDP" et "FIB" (*fast informed bound*) ne sont plus valables, du fait de la non-convexité de la fonction de valeur. Un moyen de pouvoir quand même s'y ramener serait de majorer la fonction de récompense par une fonction linéaire et de calculer un majorant de la fonction de valeur du POMDP ainsi obtenu.

5.5.2 L'algorithme s-Lipschitz HSVI

Les approximateurs employés U et L étant des minorants et majorants uniformément améliorables, on peut reprendre le schéma d'HSVI (voir algorithme 1), lequel génère des trajectoires

— en partant de l'état initial s_0 ;
— en choisissant, dans chaque état s, l'action a maximisant $Q^U(s,a) = r(s,a) + \gamma \sum_{s'} T(s,a,s')U(s')$;
— en choisissant le prochain état s' maximisant $T(s,a,s')\,(U(s') - L(s'))$;
— en mettant à jour la fonction de valeur localement en chaque s visité ; et
— en s'arrêtant quand $\gamma^d(U(s) - L(s)) \leq \epsilon$, où d est la profondeur actuelle de la trajectoire en cours.

Algorithme 1 : s-Lipschitz Heuristic Search Value Iteration (sL-HSVI)

Fct `HSVI` (ϵ)
  Initialiser L et U (par exemple avec V^{max} et V^{min})
  Calculer ν_∞^{max}
  si $\nu_\infty^{max} = \infty$ **alors**
    retourner *Erreur*
  tant que $(U(s_0) - L(s_0)) > \epsilon$ **faire**
    `EssayerRécursivement` $(s_0, d = 0)$
  retourner L
Fct `EssayerRécursivement` (s, d)
  si $\gamma^d(U(s) - L(s)) > \epsilon$ **alors**
    `MettreAJour` (s)
    $a^* \in \arg\max_{a \in \mathcal{A}} Q^U(s,a)$
    $s^* \in \arg\max_{s' \in Suivants(s,a^*)} Pr(s,a^*,s')(U(s') - L(s'))$
    `EssayerRécursivement` $(s^*, d+1)$
    `MettreAJour` (s)
  retourner
Fct `MettreAJour` (s)
  $L \leftarrow$ miseAJour(L, s)
  $U \leftarrow$ miseAJour(U, s)

Une des spécificités de l'algorithme *s-Lipschitz HSVI* (sL-HSVI) obtenu est de d'abord calculer la constante de Lipschitz ν_∞^{max}, et de s'interrompre si cette limite diverge. Une autre spécificité est évidemment l'utilisation des fonctions majorantes et minorantes en dents de scie lipschitziennes de constante ν_∞^{max}. La fonction `MettreAJour`(s) effectue une mise à jour locale de U comme de L. Elle calcule $u(s) = \max_a r(s,a) + \gamma \sum_{s'} T(s,a,s')U(s')$ (et, en même temps, l'action la plus prometteuse a^*) et $l(s) = \max_a r(s,a) + \gamma \sum_{s'} T(s,a,s')L(s')$ pour ajouter les points $(s, u(s))$ et $(s, l(s))$ aux deux approximateurs (sauf s'ils atteignent V^{max} ou V^{min}).

On notera par ailleurs :

— qu'une procédure d'élagage de l'ensemble S_U (resp. de S_L) peut être appliquée quand la taille de cet ensemble dépasse un certain seuil (par exemple lorsque cet ensemble a cru de 10% depuis le dernier élagage) ; et

— que, dans le cas de problèmes à horizon temporel fini, on calculera une fonction de valeur par pas de temps, ce qui passera par la détermination d'une constante ν_t pour chaque pas de temps par récurrence (ce qui permettra des approximations plus fines) comme vu dans le lemme 4 et le corollaire 1.

Une question intéressante est de savoir s'il est préférable de traiter les problèmes à horizon infini comme présenté ci-dessus, ou en les approchants avec un horizon temporel fini, ce qui permet d'avoir des approximateurs plus efficaces (du fait des constantes de Lipschitz plus fines).

Le schéma algorithmique employé ici reste identique à celui introduit par Smith & Simmons (2004). Les propriétés théoriques l'accompagnant sont par ailleurs préservée (du fait de l'uniforme améliorabilité de U et L). Ainsi l'algorithme converge en un nombre fini d'itérations.

6 Discussion et conclusion

Nous avons ici montré que, moyennant l'hypothèse que son modèle (T et r) était lipschitzien par rapport à l'état, on pouvait approcher la fonction de valeur d'un MDP à espace d'état continu par une fonction lipschitzienne en dents de scie tout en bornant l'erreur faite. A l'instar de la propriété de convexité et de linéarité par morceaux dans les POMDP, cette propriété permet de proposer des algorithmes de résolution à erreur bornée pour ces MDP lipschitziens. Ceci s'applique en particuliers aux ρ-POMDP – variantes de POMDP pour la recherche d'information –, problèmes dans lesquels la fonction de valeur n'est pas nécessairement convexe dans l'espace des états de croyance. Ces approximations par des fonctions en dent de scie lipschitziennes sont nettement moins fines que les approximations (pseudo-)convexes [6] employées pour les POMDP, mais des expérimentations seraient nécessaires pour mieux évaluer la dégradation obtenue, en particulier en fonction de la dimensionalité du problème. Des expérimentations permettraient par ailleurs de juger de la finesse des majorants des constantes de Lipschitz obtenus.

Une perspective intéressante serait aussi d'approcher (à erreur bornée) un modèle non-lipschitzien de MDP par un modèle lipschitzien. On pourrait alors appliquer des algorithmes de résolution tels que présentés ici tout en majorant l'erreur faite en les employant sur un modèle approché plutôt que sur le problème original. Une autre perspective serait de lever l'hypothèse du facteur de branchement fini (utile pour les MDP), par exemple en employant des algorithmes reposant sur de l'échantillonnage de trajectoires, donc des garanties moins fortes. Une inspiration pourrait être *Forward Search Sparse Sampling* (FSSS) de Walsh *et al.* (2010), un algorithme cousin de Sparse Sampling (SS) (Kearns *et al.*, 2002) et de MCTS (Coulom, 2006; Chaslot *et al.*, 2008), mais employant des estimations majorantes et minorantes de la fonction de valeur comme HSVI. En revanche FSSS n'exploite pas de généralisation de ces minorants et majorants comme le fait HSVI. Enfin, on notera que la Lipschitz-continuité par rapport à un espace d'actions continu est aussi une propriété que l'on pourrait aborder (comme l'ont fait Ieong *et al.* (2007)), par exemple en utilisant des algorithmes d'optimisation spécifiques tels que proposés par Munos (2011, 2014), et toujours en majorant l'erreur faite.

Mais ces travaux ont d'abord été motivés par le souhait de résoudre des jeux de Markov à observabilité partielle (POSG). En effet, si la transformation en un *occupancy MDP* d'un DecPOMDP a permis à Dibangoye *et al.* (2013, 2016) de se ramener à un cadre comparable à celui des bMDP (avec fonction de valeur PWLC), la même transformation, appliquée à un POSG, devrait permettre de se ramener à un jeu de Markov (complètement observable) pour lequel la fonction de valeur (scalaire dans un jeu à 2 joueurs et somme nulle, vectorielle sinon) devrait être non-convexe, mais lipschitzienne.

Références

ARAYA-LÓPEZ M., BUFFET O., THOMAS V. & CHARPILLET F. (2010). A POMDP extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems 23 (NIPS-10)*.

ASTROM K. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, **10**(1), 174 – 205.

6. *Saw-tooth* n'est pas convexe.

BELLMAN R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, **6**(5), 679–684.

CHASLOT G., BAKKES S., SZITA I. & SPRONCK P. (2008). Monte-Carlo tree search : A new framework for game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*.

COULOM R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the Fifth International Conference on Computer and Games (CG-2006)*.

DIBANGOYE J., AMATO C., BUFFET O. & CHARPILLET F. (2013). Optimally solving Dec-POMDPs as continuous-state MDPs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-13)*.

DIBANGOYE J., AMATO C., BUFFET O. & CHARPILLET F. (2016). Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, **55**, 443–497.

HAUSKRECHT M. (1997). Incremental methods for computing bounds in partially observable Markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)*.

HAUSKRECHT M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, **13**, 33–94.

IEONG S., LAMBERT N., SHOHAM Y. & BRAFMAN R. (2007). Near-optimal search in continuous domains.

KEARNS M., MANSOUR Y. & NG A. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, **49**, 193–208.

KURNIAWATI H., HSU D. & LEE W. (2008). SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics : Science and Systems IV*.

MUNOS R. (2011). Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems (NIPS'11)*.

MUNOS R. (2014). From bandits to Monte-Carlo Tree Search : The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, **7**(1), 1–130.

PINEAU J., GORDON G. & THRUN S. (2003). Point-based value iteration : An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, p. 1025–1032.

PINEAU J., GORDON G. & THRUN S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, **27**, 335–380.

PLATZMAN L. K. (1977). *Finite Memory Estimation and Control of Finite Probabilistic Systems*. PhD thesis, Massachusetts Institute of Technology (MIT).

POUPART P., KIM K.-E. & KIM D. (2011). Closing the gap : Improved bounds on optimal POMDP solutions. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS)*.

SHANI G., BRAFMAN R. & SHIMONY S. (2007). Forward search value iteration for POMDPs. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI'07)*.

SMALLWOOD R. & SONDIK E. (1973). The optimal control of partially observable Markov decision processes over a finite horizon. *Operation Research*, **21**, 1071–1088.

SMITH T. (2007). *Probabilistic Planning for Robotic Exploration*. PhD thesis, The Robotics Institute, Carnegie Mellon University.

SMITH T. & SIMMONS R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

SMITH T. & SIMMONS R. (2005). Point-based POMDP algorithms : Improved analysis and implementation. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI)*.

SOMANI A., YE N., HSU D. & LEE W. S. (2013). DESPOT : Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems (NIPS)*.

SONDIK E. (1971). *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Standford University.

SPAAN M. & VLASSIS N. (2005). Perseus : Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, **24**, 195–220.

WALSH T., GOSCHIN S. & LITTMAN M. (2010). Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'10)*.

ZHANG N. L. & ZHANG W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, **14**, 29–51.

MOMDP modeling for UAV safe path planning in an urban environment

Jean-Alexis Delamer[1,2], Yoko Watanabe[1], and Caroline P.Carvalho Chanel[2]

[1]ONERA - The French Aerospace Lab, Information Processing and Systems Departement (DTIS)
[2]Université de Toulouse, ISAE-SUPAERO Institut Supérieur de l'Aéronautique et de l'Espace, Design and Control of Aerospace Vehicles Departement (DCAS)

Abstract

Path planning is a research domain very active and applied among others on autonomous vehicle such as UAV. In recent years, a lot of progress has been made on path planning under uncertainties issued by a vehicle navigation system, for example in localization or environment mapping. However, such uncertainties are often treated by the path planner in a deterministic way. That is, the navigation system's performance is deterministically given in function of the environment. This paper tackles a more complex problem of UAV safe path planning in an urbain environment, in which UAV is at risks of GPS signal occusion and obstacle collision. The key idea is to make a UAV path planning along with its navigation and guidance mode planning, where each of such mode uses different set of sensors and whose availability and performance are environment-dependent. A partial knowledge on the environment is supposed to be available, in a form of probability maps of obstacles and sensor availabilities. To solve this problem the UAV need to be well represented in the planner model and so do the associated uncertainty propagation. This paper proposes a model based on Mixed Observability Markov Decision Process (MOMDP). The proposed MOMDP model allows the planner to choose the best path-direction with the adapted sensor set for an UAV to reach a mission goal efficiently and safely. This paper only provides a MOMDP model for the planner, and the planning algorithm and preliminary results will be expected in the final paper.

1 Introduction

These last years there has been a growing need for UAVs (Unmanned Aerial Vehicles) to accomplish distant mission in cluttered environments (urban areas with high presence of building, ...). Such missions can be done only if the safety conditions are gather, and especially the planned path must be collision risk free. The safety of the UAV depends on its onboard navigation capabilities (including onboard sensor performances) as well as on the environment. Moreover the sensors availability and performance depends on the environment, for example GPS is particularly dependent of the quality of the signal (occlusion / degradation) and especially in cluttered environment. All this parameter affect the navigation performance and must be taking into account in path planning to ensure the success of the mission.

Therefore, the community have proposed different frameworks and algorithms to solve path planning under uncertainty in cluttered and continuous environment. A method was proposed by [1] to compute an efficient colision-free path for MAV (Micro Aerial Vehicles) while taking into account the dynamic of the MAV. This vehicle has onboard camera to estimate its localization and the path is computed based on the algorithm RRBT (Rapidly-exploring Random Belief Trees [6]) which allows to consider position uncertainty during planning. The autonomous vehicles can have several onboard sensors to ensure the safety of the vehicles and its passengers. Another path planning approach has been proposed for autonomous vehicles in unknowned semi-structered environment [7] by using an hybride-state A* search algorithm. Note that these previous frameworks compute a static path (a plan and not a conditionnal plan or policy). When considering a dynamic path which is able to self-adapt in function of the events, some frameworks were also proposed by the community. One could cite, the work of [5] that propose a method based on POMDP (Partially observable Markov decision process)

for autonomous vehicles in real situations. Thus this approach take into account other vehicles to take its decisions (changing lane, break, ...).

All these works are based on various sensors to enable the vehicle to guide itself and compute the best path. One limitation of these works, in the case of UAV (or MAV) is that they do not consider path planning in urban environment. The number of onboard sensors are often relatively low and it do not take into account the availability of the sensors as for example, their precision in function of the environment and there is only one navigation mode (using all the sensors).

In this paper, it is proposed a path planner model that will ensure an efficient risk-free path. To success, it will be take into account the dynamic of the UAV, and the availibility of sensors in the environment through a set of availability map. The sensor availability maps are probability grids which are in overlay of the environment map, which is not discretized (Figure 3b). Like the sensor availability maps, the obstacles map is a grid in overlay of the environment map. The path computed called a policy will allow at each instant the UAV to take the best action in function of the sensor's availability. Depending on the availability of the sensors, different navigation modes could be used, resulting in different localization and execution error propagation. this point should be considered in path computation. So, in this problem the UAV have an intertial navigation system (INS), a GPS, an optical flow field measurement and a landmark pixel coordinate measurement to navigate. The UAV have two guidance modes, the waypoint tracking whose precision depends on the navigation mode used and the wall following mode. These sensors and the guidances modes allow the planner to optimize the combination sensors/guidance mode for each event that can occur.

Navigate through an cluttered environment can be hard for an UAV due to the proximity of the obstacle and the great variance on the availability of the sensors. Therefore the objective of this works are to find a plan (or policy in MOMDP) that take into account the availability of the onboard sensors to ensure the shortest collision-free path. The policy must take in entry the belief on the actual state and return the best action to execute. The main idea is to combine the transition function of the decision process to the error (localization and execution) propagation function, which depends on the localisation and guidance mode for each part of the path. The advantage of this method is to incorporate a priori knowledges on the disponibility of the sensor in the path planning, and to propagate this informations on the futur path. This allows the algorithm to calculate a path to guide the vehicule to the safest and shortest path. Taking account of the previous idea, and with our uncertainty model, we have chosen to lean our model on the Markov Decision Process [3]. Given the nature of our problem and the partially informations on the state of the system, we will use the MDP extension : Mixed Observability Markov Decision Processes (MOMDP) [10].

The paper is organized as follows: in Section 2 it will be explained the problem including the system architecture, the GNC's transition model and details on the maps used. In Section 3 it will be recalled the definition of the MOMDP and then it will be defined the MOMDP planning model. Finaly, in Section 4 it will be given the perspectives of the future works and a conclusion.

2 Problem statement

2.1 System architecture and time differentiation

The architecture of the overall system considered in this problem combine the GNC's transition model which includes the vehicle motion model, onboard sensor model and flight control systems, and the MOMDP planning model. The planner will consider GNC's[1] transitions, which are known to compute the possible evolutions of the system's state. The policy given by the MOMDP planner take as inputs the probability distribution over the actual state b_{s_c} and a vector of boolean on sensors availability s'_v. And it will return an action, which will select the direction and the navigation and guidance modes to perform during a certain amount of time, a new vector of sensor's availability is observed, and a belief state update is performed. Again this new belief state is used by the policy to define the next action. During the following article, the diagram presented in Figure 1 will be referred to facilitate understanding.

The formalisms such as the MDPs, in their majority and most of the variants, consider the actions to accomplish instantly and not as a process during over the time. There is a variant, the semi-Markov Decision Problems (SMDP), which applicate the MDP to the continuous time, where each actions have

[1] Guidance, Navigation and Control (GNC)

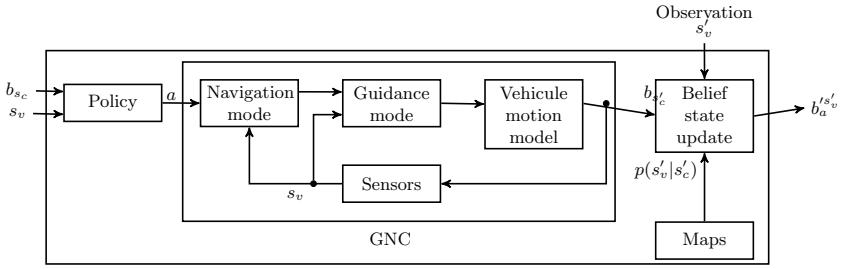


Figure 1: Architecture diagram

a execution time [4]. However, in this case, the problem is not that actions are durative, but the state of the UAV changes during an action. Our action include a navigation mode used to locate the UAV during the entire movement in the direction d.

In this sense, it will be distinguished two unit of time due to the difference between the unit of time of the GNC's transition model and the time unit of the MOMDP planning model named *epoch*. The MOMDP planning model works at a lower frequency that the system, thus an epoch is equivalent to several unit of time of the system's transition model. The reason is to lower the complexity of the algorithm by reducing the total number of actions to complete the task.

To differentiate the unit of time, the GNC unit of time will be noted k and the epoch of the MOMDP will be noted t (Figure 2).

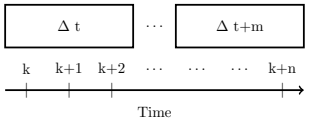


Figure 2: Difference between the units of time

2.2 GNC's transition model

As stated earlier and shown in Figure 1, the system's transition model consists of the vehicle motion model, onboard sensor model, and the UAV flight guidance, navigation and control (GNC) system. This section presents these models which construct a state transition model used in by the planning model.

2.2.1 State transition model

The state x of the UAV is defined by its positon, its velocity and the accelerometer bias such as $x = \begin{bmatrix}\mathcal{X}^T & \mathcal{V}^T & b_a\end{bmatrix}^T$, where $\mathcal{X}$,$\mathcal{V}$ and b_a are respectively the UAV position, the velocity and the accelerometer bias. Then the state transition can be defined such as :

$$\dot{x} = \begin{bmatrix}\dot{\mathcal{X}} \\ \dot{\mathcal{V}} \\ \dot{b_a}\end{bmatrix} = \begin{bmatrix}\mathcal{V} \\ a \\ 0\end{bmatrix} + \begin{bmatrix}v_x \\ v_v \\ v_a\end{bmatrix} = \begin{bmatrix}0 & I & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} x + \begin{bmatrix}0 \\ I \\ 0\end{bmatrix} a + v = Ax + Ba + v \tag{1}$$

where a is the acceleration and $v \sim N(0, \tilde{Q})$ is the process noise. According to this model, the state transition from $x(t_k) = x_k$ to $x(t_{k+1} = t_k + \Delta t) = x_{k+1}$ can be derived as:

$$x_{k+1} = \Phi x_k + B a_k + v_{k+1} \tag{2}$$

where $v_{k+1} \sim N(0, Q)$ is the discretized process noise and

$$\Phi = \begin{bmatrix}I & \Delta t I & 0 \\ 0 & I & 0 \\ 0 & 0 & I\end{bmatrix}, B = \begin{bmatrix}\frac{\Delta t^2}{2} I \\ \Delta t I \\ 0\end{bmatrix}, Q \simeq \Delta t \tilde{Q}$$

2.2.2 State estimator

A true value of the vehicle state x is never accessible in reality, and so is estimated by the navigation module by using sensor measurements available onboard (Figure 1). This problem supposes the state estimator based on a EKF (Extended Kalman Filter [15]) which includes two steps :

i. the INS prediction

ii. the sensor measure (if available) used for correction

The INS measurement integration enables a high frequency state estimation, but it suffers from the estimation drift due to the measurement bias. In order to limit or even correct such drift, other sensors are fused with INS through the second step of Kalman filter. The most common and effective sensor is GPS (or other similar satellite-based navigation system) which can provide an accurate absolute position measurement of an UAV. But it is at a high risk of its performance degradation due to multi-path effect or signal occlusion in a cluttered environment. Alternative approaches proposed by the robotics community (particularly for indoor UAVs) are based on onboard vision information. Such approaches include visual odometry and SLAM algorithms for motion estimation, or image matching with geo-referenced map (e.g. image database, landmarks) for absolute localization. As an example, we consider GPS and vision-based landmark detection as navigation sensors in this paper.

INS Prediction : The accelerometer measurement a_{IMU_k} is used to propagate the estimated state. It measures the biased, non-gravitational UAV body acceleration

$$a_{\text{IMU}_k} = R_{BI_k}(a_k - g) + b_{a_k} + \xi_{\text{IMU}_k} \tag{3}$$

where R_{BI_k} is a rotation matrix from the inertial to the UAV body frames (assumed to be known), g is the gravity vector and $\xi_{\text{IMU}} \sim N(0, R_{\text{IMU}})$ is the IMU measurement error. According to the process model (Eq 2), the estimated state $\hat{x}_k$ is propagated to

$$\begin{aligned}
\hat{x}^-_{k+1} &= \Phi\hat{x}_k + B\left(R^T_{BI_k}\left(a_{\text{IMU}_k} - \hat{b}_{a_k}\right) + g\right) \\
&= \Phi\hat{x}_k + B\left(a_k + R^T_{BI_k}\left(\tilde{b}_{a_k} + \xi_{\text{IMU}_k}\right)\right) \\
&= x_{k+1} - \Phi\tilde{x}_k - v_{k+1} + BR^T_{BI_k}\left(\tilde{b}_{a_k} + \xi_{\text{IMU}_k}\right)
\end{aligned} \tag{4}$$

Then the state prediction error is given by :

$$\begin{aligned}
\tilde{x}^-_{k+1} = x_{k+1} - \hat{x}^-_{k+1} &= \Phi\tilde{x}_k + v_{k+1} - BR^T_{BI_k}\left(\tilde{b}_{a_k} + \xi_{\text{IMU}_k}\right) \\
&= \left(\Phi - \Delta\Phi^a_k\right)\tilde{x}_k + v_{k+1} - BR^T_{BI_k}\xi_{\text{IMU}_k}
\end{aligned} \tag{5}$$

where $\Delta\Phi^a_k = BR^T_{BI_k}\begin{bmatrix}0 & 0 & I\end{bmatrix}$. The associated error covariance is given by :

$$P^-_{k+1} = \left(\Phi - \Delta\Phi^a_k\right)P_k\left(\Phi - \Delta\Phi^a_k\right)^T + Q + \tilde{R}_{\text{IMU}_k} \tag{6}$$

here $\tilde{R}_{\text{IMU}_k} = BR^T_{BI_k}R_{\text{IMU}}R_{BI_k}B^T$. For simplicity, we can consider the case of $R_{\text{IMU}} = \sigma^2_{IMU}I$ and hence $\tilde{R}_{\text{IMU}} = BR_{\text{IMU}}B^T$ remains constant for all k.

GPS correction : When GPS is available at t_{k+1}, the predicted state (Eq: 5) can be corrected by using its position and velocity measurement $z_{\text{GPS}_{k+1}}$.

$$z_{\text{GPS}_{k+1}} = \begin{bmatrix}I & 0 & 0\\ 0 & I & 0\end{bmatrix}x_{k+1} + \xi_{\text{GPS}_{k+1}} = H_{\text{GPS}}x_{k+1} + \xi_{\text{GPS}_{k+1}}$$

where $\xi_{\text{GPS}} \sim N(0, R_{\text{GPS}})$ is the GPS measurement error. Then, the GPS measurement is used to correct the estimated state such as :

$$\begin{aligned}
\hat{x}_{k+1} &= \hat{x}^-_{k+1} + K_{\text{GPS}_{k+1}}H_{\text{GPS}}\left(z_{\text{GPS}_{k+1}} - H_{\text{GPS}}\hat{x}^-_{k+1}\right) \\
&= \hat{x}^-_{k+1} + K_{\text{GPS}_{k+1}}H_{\text{GPS}}\tilde{x}^-_{k+1} + K_{\text{GPS}_{k+1}}\xi_{\text{GPS}_{k+1}} \\
&= x_{k+1} - \left(I - K_{\text{GPS}_{k+1}}H_{\text{GPS}}\right)\tilde{x}_{k+1} + K_{\text{GPS}_{k+1}}\xi_{\text{GPS}_{k+1}}
\end{aligned} \tag{7}$$

where $K_{\mathrm{GPS}_{k+1}} = P^-_{k+1} H^T_{\mathrm{GPS}} \left(H_{\mathrm{GPS}} P^-_{k+1} H^T_{\mathrm{GPS}} + R_{\mathrm{GPS}} \right)^{-1}$ is a Kalman gain. Then the error estimate and its covariance are given by :

$$\begin{aligned} \tilde{x}_{k+1} &= x_{k+1} - \hat{x}_{k+1} = \left(I - K_{\mathrm{GPS}_{k+1}} H_{\mathrm{GPS}} \right) \tilde{x}_{k+1} - K_{\mathrm{GPS}_{k+1}} \xi_{\mathrm{GPS}_{k+1}} \\ P_{k+1} &= \left(I - K_{\mathrm{GPS}_{k+1}} H_{\mathrm{GPS}} \right) P^-_{k+1} \end{aligned} \tag{8}$$

Landmark correction : When a landmark (whose position $\mathcal{X}_{\mathrm{LM}}$ is a-priori given) is visible and detectable on an onboard camera image at t_{k+1}, its pixel-coordinates information can be used to correct the predicted state (Eq: 5). By assuming a pin-hole camera model, the pixel-coordinates measurement is given by the following nonlinear measurement model.

$$z_{\mathrm{LM}_{k+1}} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \frac{\mathcal{X}^C_{LM}}{e_3^T \mathcal{X}^C_{LM}} + \xi_{\mathrm{LM}_{k+1}} = C \frac{\mathcal{X}^C_{\mathrm{LM}_{k+1}}}{e_3^T \mathcal{X}^C_{\mathrm{LM}_{k+1}}} + \xi_{\mathrm{LM}_{k+1}} = h_{\mathrm{LM}}(x_{k+1}) + \xi_{\mathrm{LM}_{k+1}} \tag{9}$$

where C is a known camera matrix, $\mathcal{X}^C_{\mathrm{LM}_{k+1}} = R_{CB} R_{BI_{k+1}} (\mathcal{X}_{\mathrm{LM}} - \mathcal{X}_{k+1})$ and $\xi_{\mathrm{LM}} \sim N(0, R_{\mathrm{LM}})$ is the landmark image-detection error in pixels. The camera is assumed to be mounted at the c.g. of the UAV with a known camera orientation R_{CB} with respect to the UAV body. An extended Kalman filter can be applied, and similary to (Eq: 8), the resulting estimation error and its covariance matrix are given by :

$$\begin{aligned} \tilde{x}_{k+1} &= \left(I - K_{\mathrm{LM}_{k+1}} H_{\mathrm{LM}_{k+1}} \right) \tilde{x}_{k+1} - K_{\mathrm{LM}_{k+1}} \xi_{\mathrm{LM}_{k+1}} \\ P_{k+1} &= \left(I - K_{\mathrm{LM}_{k+1}} H_{\mathrm{LM}_{k+1}} \right) P^-_{k+1} \end{aligned} \tag{10}$$

where $H_{\mathrm{LM}_{k+1}}$ is a Jacobian matrix of the nonlinear measurement function $h_{\mathrm{LM}}(x_{k+1})$ evaluated at $x_{k+1} = \hat{x}^-_{k+1}$. It should be noted that $H_{\mathrm{LM}_{k+1}}$ thus depends on the predicted state $\hat{x}^-_{k+1}$, while H_{GPS} does not.

INS-only solution : If no correction is made with neither GPS nor Landmark detection, the state estimate at t_{k+1} is given by the predicted one:

$$\begin{aligned} \tilde{x}_{k+1} &= \tilde{x}^-_{k+1} \\ P_{k+1} &= P^-_{k+1} \end{aligned} \tag{11}$$

2.2.3 Guidance laws

A set of actions will be defined in the planner model, such as each action is defined among others as a desired direction of motion control. For example, as the exact definition will be given in subsection 3.3.2, a set of four actions can be defined as : {"Move-to-North", "Move-to-East", "Move-to-South","Move-to-West"}.

Given a direction and a reference speed $\mathcal{V}_{\mathrm{ref}}$ is desired velocity in the desired direction the following linear guidance law can be applied to realize the action :

$$a_k = K_p \mathcal{V}_{\mathrm{ref}} - K_d \hat{\mathcal{V}}_k \tag{12}$$

where K_p,$K_d > O$ are the control gain and $\hat{V}_k$ is the estimated UAV velocity at instant t_k.

This paper considers two different guidance modes in terms of information sources used in the guidance law (Eq 12) for $\hat{\mathcal{V}}_k$. The first guidance mode uses the state estimation result from the navigation module given in Section 2.2.2, while the second mode uses some sensor measurements directly. The former mode corresponds to a conventional absolute guidance approach such as WP tracking (Waypoint Tracking), and the latter to a sensor-based relative guidance approach such as visual servoing to follow a wall. The advantage of the sensor-based relative guidance mode is that it is independent from the navigation (i.e., localization) performance, but in return its applicability is rather limited to a proximity of some known object (e.g. wall).

Absolute guidance : In the absolute guidance mode, we have $\hat{\mathcal{V}}_k = \begin{bmatrix}0 & I & 0\end{bmatrix}\hat{x}_k$ where $\hat{x}_k$ is the estimated state from Section 2.2.2. Then, x_{k+1} can be obtained by substituing this guidance law (Eq 12) into the discretized process model (Eq 2) :

$$\begin{aligned} x_{k+1} &= \Phi x_k + B(K_p\mathcal{V}_{\text{ref}} - K_d\begin{bmatrix}0 & I & 0\end{bmatrix}\hat{x}_k) + v_{k+1} \\ &= (\Phi - \Delta\Phi^{\mathcal{V}})x_k + BK_p\mathcal{V}_{\text{ref}} + \Delta\Phi^{\mathcal{V}}\tilde{x}_k + v_{k+1} \end{aligned} \tag{13}$$

where $\Delta\Phi^{\mathcal{V}} = BK_d\begin{bmatrix}0 & I & 0\end{bmatrix}$. Hence, the state x_{k+1} follows the normal distribution as below.

$$x_{k+1} \sim N((\Phi - \Delta\Phi^{\mathcal{V}})x_k + BK_p\mathcal{V}_{\text{ref}}, \Delta\Phi^{\mathcal{V}}P_k\Delta\Phi^{\mathcal{V}^T} + Q) = N(\bar{x}_{k+1|k}, \tilde{Q}^a_{k+1}) \tag{14}$$

where the covariance $\tilde{Q}^a_{k+1}$ is a function of the estimation error covariance P_k.

Sensor-based relative guidance : In the sensor-based relative guidance mode, $\hat{\mathcal{V}}_k$ in (Eq 12) is directly given from some onboard sensors. For example, optical flow information from a video sequence can be used in combination with a distance measurement. Let us assume the measurement error $\tilde{\mathcal{V}}_k = (\mathcal{V}_k - \hat{\mathcal{V}}_k) \sim N(0, R_{\mathcal{V}_k})$. Then, similarly to (Eq 13),

$$\begin{aligned} x_{k+1} &= (\Phi - \Delta\Phi^{\mathcal{V}})x_k + BK_p\mathcal{V}_{\text{ref}} + BK_d\tilde{\mathcal{V}}_k + v_{k+1} \\ &\sim N(\bar{x}_{k+1|k}, BK_dR_{\mathcal{V}_k}K_d^TB^T + Q) = N(\bar{x}_{k+1|k}, \tilde{Q}^s_{k+1}) \end{aligned} \tag{15}$$

where the covariance $\tilde{Q}^s_{k+1}$ is independent from P_k.

2.2.4 State probability density function

Given the initial state $x(t_0) = x_0$ and the initial estimation error covariance P_0 which gives $\tilde{x}_0 \sim N(0, P_0)$. As defined later in Section 3.3.2, an action a in the planner model corresponds to a combination of the direction of desired motion ($\mathcal{V}_{\text{ref}}$), the navigation mode (GPS, Landmark or INS-only) and the guidance mode (Absolute or Sensor-based). For a given action a, it is possible to obtain the distribution of the next state $x_1 = x(t_0 + \Delta t)$ knowing x_0.

$$f_X(x_1|x_0) \sim N\left(\bar{x}_{1|0}, \tilde{Q}_1\right) \tag{16}$$

where $\tilde{Q}_1$ is either $\tilde{Q}^a_1$ in (Eq 14) or $\tilde{Q}^s_1$ in (Eq 15) depending on the selected guidance mode. At the same time, the state estimation error covariance is updated to P_1 by using the selected navigation mode (Eqs 8, 10 or 11).

Now recall from Section 2.1 that the system's transition model and the planning model do not have the same time unit. Usually, the planning time step is much longer than that of the system's transition model. It means that a single state transition $s = s_0$ to $s' = s_1$ with a selected action a in the planner corresponds to several consecutive state transitions x_0 to x_n in the system's transition model. So we have to continue the state transition further up to $n > 1$ from Eq. 16 and P_1 with the same action a. The conditional state distribution at t_k ($k > 1$) knowing x_0 can be obtained sequentially as follows.

$$f_X(x_k|x_0) = \int f_X(x_k|x_{k-1})f_X(x_{k-1}|x_0)dx_{k-1}$$

where $f_X(x_k|x_{k-1}) \sim N(\bar{x}_{k|k-1}, \tilde{Q}_k)$. In parallel, the Kalman filtering process (of the selected navigation mode) is repeated k times to obtain P_k. When $\tilde{Q}_k$ and P_k do not depend on the state x_{k-1}, the integration above will result in a normal distribution:

$$\begin{aligned} f_X(x_k|x_0) &\sim N((\Phi - \Delta\Phi^{\mathcal{V}})\bar{x}_{k-1|0} + BK_p\mathcal{V}_{\text{ref}}, (\Phi - \Delta\Phi^{\mathcal{V}})\tilde{\Sigma}_{k-1}(\Phi - \Delta\Phi^{\mathcal{V}})^T + \tilde{Q}_k) \\ &= N(\bar{x}_{k|0}, \tilde{\Sigma}_k), \quad k > 1 \end{aligned} \tag{17}$$

where $\bar{x}_{1|0} = (\Phi - \Delta\Phi^{\mathcal{V}})x_0 + BK_p\mathcal{V}_{\text{ref}}$ and $\tilde{\Sigma}_1 = \tilde{Q}_1$. $\tilde{Q}_k$ is obtained by (Eq 14 or 15).

The derivation of the state distribution (Eq 17) becomes more complex in a case of having a dependency of $\tilde{Q}_k$ and P_k on the state x_{k-1}. In order to avoid this complication, the matrices $\tilde{Q}_k$ and

P_k can be approximated by those evaluated at the expected state $\bar{x}_{k-1|0}$. Then, the state transition function required in the planning model can be given by Eq. 18 when $k = n$. The decision of the couple direction / navigation mode need that the navigation mode is available. The navigation and guidance modes are necessary during the entire action a, however during the verification of the availability of a mode it is very difficult to know if the sensor will be available for the entire action. It is necessary to precise that in the case where a sensor will not be available during the entire movement, the navigation mode will switch to $INS - Only$ (the only navigation mode available permanently). To simplify our decisional problem, we will suppose that if the sensor is available at the end of the action, the sensor was available during the entire movement. This state transition function from the state $s = s_0$ to $s' = s_1$ can be re-wtitten with a notation fro the planning model as below.

$$f_S(s' = s_1|s = s_0) = f_X(x_n|x_0) \sim N(\bar{x}_{n|0}, \tilde{\Sigma}_n) = N(s', \Sigma') \tag{18}$$

It is worth emphasizing here that this equation will be the only link to the GNC's transition model with the MOMDP planning model (Section 3.3).

2.3 Environment Maps

The planner will use a-priori knowledge on the environment in which UAV will navigate. This knowledge is provided as a set of maps including information on obstacles as well as on sensor availabilities(Fig 3b). It is supposed that a 3D environment model of the UAV mission operation site is available beforehand, either from some database, from data obtained from the past missions, or from data acquired in a pre-mission flight at high and safe altitude. This 3D environment model can be represented as a discretized 3D occupancy map, where the 3D space is divided into cells. Let us denote the i-th cell of the map by c_i. Then a probability that the cell c_i is occupied, i.e., obstacle, is given as $p(Collision|c_i)$.

Sensor availability maps are also given in the same form, as a set of probabilities that a sensor is available at each cell c_i. These maps can be generated by considering the corresponding sensor characteristiques in relation with the environment (given by the occupancy map). For example, GPS performance suffers from its signal occlusion or multi-path effect due to surrounding obstacles. It is common to measure the performance of GPS by metrics called DOPs (Dilutions of Precision) which corresponds to a standard deviation of the measured position error. Such DOPs can be predicted by a GPS simulator for a specified geo-location, date and time, and environment (obstacles). Figure 3a shows examples of GPS-PDOP (Position Dilution of Precision) map obtained by using OKTAL SE-NAV simulator available at ONERA/DEMR (Dept. of Electro Magnetism and Radar). In this paper, this PDOP map is transformed to GPS availability map by setting a maximum allowable position error threshold ϵ. In short, GPS is considered to be available if its position error $\tilde{\mathcal{X}}_{\mathrm{GPS}}$ is inferior to this threshold. From the corresponding PDOP value, a probability of GPS availability at each cell c_i is calculated by

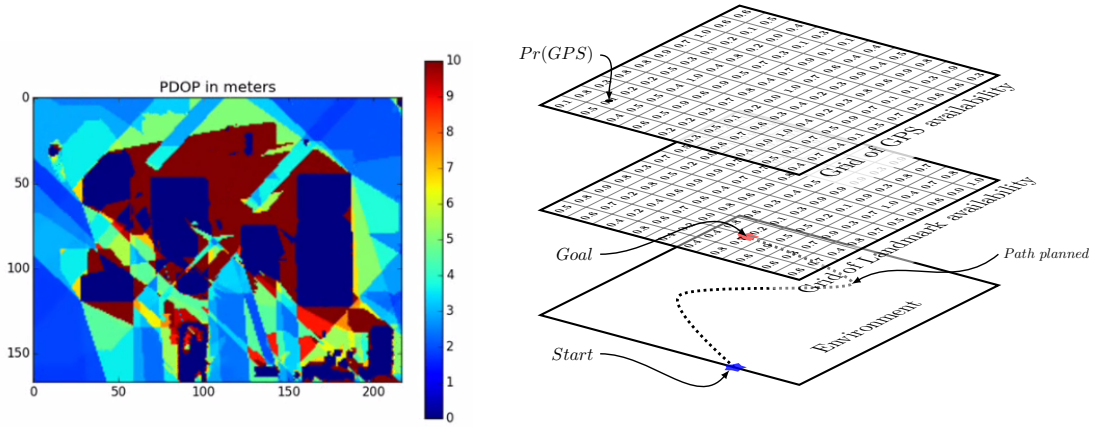
$$p(GPS|c_i) = p(\tilde{\mathcal{X}}_{\mathrm{GPS}}(c_i) < \epsilon), \quad \mathcal{X}_{\mathrm{GPS}}(c_i) \sim N(0, PDOP(c_i)).$$

Availability of the navigation mode using landmark detection at each cell is obtained in function of the camera's field-of-view and detection performance of the image processor. It is straightforward from the pin-hole camera model (Eq 9) to judge whether a given landmark position lies within the camera's field-of-view from a given cell position. The image-detection rate might vary and modeled in function of relative distance. These will give a probability map of availabilities of the navigation mode with landmark detection: $p(LM|c_i)$. Likewise, availability of the sensor-based relative guidance (e.g. wall following) is conditioned by a limited sensing range with respect to an object-of-interest (wall). Its availability map as a set of probabilities $p(Wall|c_i)$ is supposed to be given.

These maps and probabilities will be used as a part of the state transition model in the MOMDP planning model. They serve for determining a state transition probability as well for updating the state distribution function conditioned by an observation made on the sensor availabilities.

3 Planning model

The goal is to propose a decision framework that will calculate the safest and the shortest path. To achieve this goal, it is necessary to have a GNC system which give a good estimate of the state of the



(a) Example of GPS-PDOP map

(b) Environment with a GPS and Landmark disponibility map

Figure 3: Examples of maps

system. The state estimate is necessary to compute the optimal path, indeed for a specific state the decision framework will evaluate for each action the possible results to select the more efficient. This is why the model previously defined take into account the different sensors of the UAV to have the possibility to compute a best approximation of the system state given possible states expectations. This model detailled, it is possible to define the model of the planner.

3.1 Why MOMDP ?

This work is about the computing of a policy which give us at each epoch the action to execute. Therefore, planning can be associated to decision-making. Decision-making is the cognitive process of choosing which action execute confronted to a situation. Decision-making in real life problem is often synonymous of uncertainty resulting of the stochastic dynamics of the agents (here an UAV) and the environment. Thus this problem can be seen as a sequential decision problem, because these problems are charactized by the enrolment of the problem over the time and that each decision lead to uncertain consequences. POMDPs and variants provide several frameworks to model sequential decision problem under uncertainty and partial observability. The idea behind the POMDPs is that the state is not known but for each state several observations are possible with a specific probability. And the agent when being in a state will receive an observation and update his belief on his state. The Mixed Observability Markov Decision Process (MOMDP) is a variant of the POMDPs, where the state can be factorized. Indeed the state is partially observable but some state variables are known at each epoch. In this problem, the UAV always know the sensor available which can be considered as part of the state, consequently MOMDP can be adapted to the problem.

3.2 Recall on MOMDPs

The MOMDP is an extension for the POMDP model recently proposed by [2] and [10], which explores the particular structure where certain state variables are fully observable. This factored model leads to a very significant time gain in policy computation, improving the efficiency of a point-based algorithm.

A MOMDP is a tuple $(\mathcal{S}_v, \mathcal{S}_c, A, \mathcal{O}, T, R, b_0, \gamma)$, where:

- $\mathcal{S}_v$ is the bounded set of fully observable state variables;
- $\mathcal{S}_c$ is the bounded set of partially observable state variables;
- A is a bounded set of actions;
- Ω is a bounded set of observation;
- $T(s, a, s') = T(s_v, s_c, a, s'_v, s'_c) \rightarrow [0; 1]$ a transition function;

- $\mathcal{O}(s', a, o) \rightarrow [0; 1]$ is an observation function such as $\mathcal{O}(s', a, o) = \mathcal{O}(s'_v, s'_c, a, o_v, o_c)$ where $p(o_v, o_c|a, s'_v, s'_c) = p(o_v|a, s'_v, s'_c, o_c)p(o_c|a, s'_v, s'_c)$;
- $R : \mathcal{S}_v \times \mathcal{S}_c \times A \rightarrow \mathbb{R}$ is a reward function associated with a state-action pair; and:
- $b_0 = (s^0_v, b_{c,0})$, where $b_{c,0}$ is the initial probability distribution over the partially observable states, conditionned by s^0_v, the fully observable initial states.
- $\gamma \in [0, 1[$ is the discount factor.

Note that, as the probability distribution over states concerns only the $\mathcal{S}_c$ set and the observation set $\mathcal{O}_c$, the belief state update is redefined as:

$$b_c^{o_c,a,s'_v}(s'_c) = \eta \sum_{s_c \in \mathcal{S}_c} p(o_c|s'_c, s'_v, a)p(s'_c|s_v, s_c, a, s'_v)p(s'_v|s_v, s_c, a)b_c(s_c) \tag{19}$$

where, η is a normalization constant. The belief state b is now noted by the couple (s_v, b_c), and $\mathcal{B}_c$ is the belief state space s_c conditioned by s_v : $\mathcal{B}_c(s_v) = \{(s_v, b_c), b_c \in \mathcal{B}_c\}$. $\mathcal{B}_c(s_v)$ is a sub-space of $\mathcal{B}$, such that $\mathcal{B} = \bigcup_{s_v \in \mathcal{S}_v} \mathcal{B}_c(s_v)$.

Solving MOMDPs consists in finding *a set of policies* $\pi_{s_v} : \mathcal{B}_c \rightarrow A$, which maximize the criterion :

$$\pi^*_{s_v} \leftarrow \arg\max_{\pi_{s_v} \in \Pi} E_{\pi_{s_v}} \left[\sum_{t=0}^{\infty} \gamma^t r((s^t_v, b^t_c), \pi((s^t_v, b^t_c))) \middle| b_0 = (s^0_v, b_{c,0}) \right] \tag{20}$$

As for the POMDP, the value function at a time step $n < \infty$ can be also represented by a set of α-vectors:

$$V_n(s_v, b_c) = \max_{\alpha \in \Gamma^n_{s_v}} (\alpha \cdot b_c) \tag{21}$$

where α is the hyperplan over the space $\mathcal{B}_c(s_v)$. In this way, the value function over the complete state space is parametrized by the set Γ_{s_c}, i.e. $\Gamma = \{\Gamma_{s_v}, s_v \in \mathcal{S}\}$. So, given a belief state (s_v, b_c) the optimal action is defined by the action associated with the α-vector that maximizes $\max_{\alpha \in \Gamma^n_{s_v}} (\alpha \cdot b_c)$. For more details about MOMDP algorithm resolution, please see [2].

3.3 MOMDP planning model

Considering the differences between the problem presented and a standard problem from the litterature, we propose a model inspired by the MOMDP. Some modifications wills be made to the MOMDP formalism (Figure : 4b). Moreover, the planner model will be based on the GNC model to create the best policy possible.

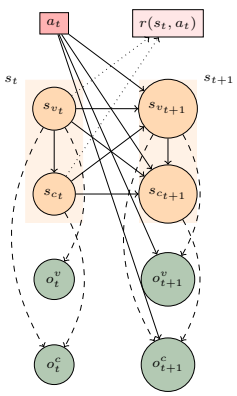
Our model is defined as a tuple $\{\mathcal{S}_v, \mathcal{S}_c, \mathcal{A}, \Omega, \mathcal{T}, \mathcal{O}, \mathcal{C}, b_0\}$:

- $\mathcal{S}_v$: bounded set of totally observable states.
- $\mathcal{S}_c$: bounded set of non observable states.
- $\mathcal{A}$: bounded set of actions.
- Ω : bounded set of observations.
- $\mathcal{T}$: The state transition function composed of two functions:
  - $T_{\mathcal{S}_c} : \mathcal{S}_c \times \mathcal{S}_v \times A \times \mathcal{S}_c \rightarrow [0; 1]$ a transition function such as : $T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) = p(s'_c|s_c, s_v, a)$.
  - $T_{\mathcal{S}_v} : \mathcal{S}_v \times \mathcal{S}_c \rightarrow [0; 1]$ a transition function such as : $T_{\mathcal{S}_v}(s'_c, s'_v) = p(s'_v|s'_c)$;
- $\mathcal{O} : \Omega \times \mathcal{S}_c \rightarrow [0; 1]$: observation function such as :

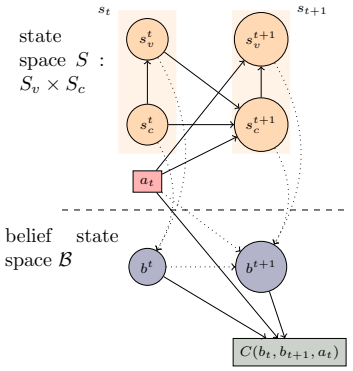
$$O(o, a, s'_c, s'_v) = p(o|s'_c, s'_v, a) = \begin{cases} 1 & o = s'_v \\ 0 & \text{otherwise} \end{cases}$$

- $\mathcal{C} : \mathcal{B} \times \mathcal{B} \times A \rightarrow \mathbb{R}$: the cost function, where $\mathcal{B}$, are the belief state space defined on $\mathcal{S}$.
- $b_0 = (s^0_v, b^0_{\mathcal{S}_c})$, où $b^0_{\mathcal{S}_c} \in \mathcal{B}_c$ is the intiale probability distribution over the non observable states, conditioned to $s^0_v \in S_v$, the intiale totally observable state.

Note : The set of observations Ω is equal to S_v and consequently the observation function $\mathcal{O}$ is deterministic since $p(o|s'_c, s'_v, a) = 1$ regardless of s'_c, s'_v and a, since $o = s'_v$. Moreover, the Bayesian dependancy in our model changes from a MOMDP proposed in [2] such as s'_v depends on s'_c and s'_c depends on s_v and s_c, therefore it depends on the position of the vehicule in the environment.



(a) Transition model for a MOMDP



(b) MOMDP model modified, we do not explicit the observations but the complete belief state space $\mathcal{B}$.

Figure 4: Difference between the transition model

3.3.1 State space of the decisional problem

In our problem, a state s is composed of a set of state variables in two categories : the first being the fully observables state variables s_v and the second being the non observable state variables s_c. The state space $\mathcal{S}$ represent all the states such as $|\mathcal{S}| = |\mathcal{S}_v| \times |\mathcal{S}_c|$, where $\mathcal{S}_c$ represent all the value possible of the state variables s_c and $\mathcal{S}_v$ represent all the value possible of the state variables s_v. It must noted that for our model, it is the entire state space $S : S_v \times S_c$ that are partially observable and we are choosing to factorise according to fully observable and non observable state variables. More specifically, we define s_c such as : $s_c = \begin{bmatrix} \mathcal{X}^T & \mathcal{V}^T & b_a \end{bmatrix}^T$. This is the same definition that previously presented in the state transition model. It is necessary to keep the velocity and the bias for the transition function (that will be presented after), because it contributes to compute the next position. $\mathcal{X}^T = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the vector corresponding to the vehicule position, that is defined on a *non observable continuous bounded space.*

As well as $s_v = \begin{bmatrix} GPS \\ Optical\ Flow \\ Landmark \\ WallFollowing \\ Collision \\ P \end{bmatrix}$ is the vector containing the totally observable boolean availability $[0; 1]$ of the sensors and guidance modes, a boolean on the collision, as well as P the localization error covariance matrix.

3.3.2 Action space of the decisional problem

In contrary to the position (x, y, z) that is in a continuous space, we define a discret action space. This action space is composed by three action variables.

- $d \in D$: are the directions where the vehicule can move. Since we are in a continuous environment with discrete actions, the actions are chosen arbitrarily. Assuming the UAV was in a 3D grid composed of voxels, the UAV would have 26 adjacent voxels and thus $|D| = 26$ directions d available. It will be considered $|D| = 26$ directions d available in our model even if the UAV is in a continuous environment.

- $m_n \in \mathcal{M}_n$: designate the different navigation mode available on the vehicule. The navigation mode determine the sensor set that will be used for the localization.

- $m_g \in \mathcal{M}_g$: designate the guidance mode available on the UAV. The guidance mode define the guidance law that will be used for the movement.

We pose the hypothesis that the UAV possesses the following navigation mode :

$$\mathcal{M}_n = \{INS - only, GPS/INS, Optical\ flow/INS, Landmark/INS\}$$

and the following guidance mode :

$$\mathcal{M}_g = \{Absolute\ Guidance, Relative\ Guidance\}$$

Then we define the action $a = (d, m_n, m_g)$ as a tuple containing a direction d, a navigation mode $m_n \in \mathcal{M}_n$ and a guidance mode $m_g \in \mathcal{M}_g$. According to the number of directions, navigation mode and guidance mode there are at most 208 actions available. Note that the navigation and guidance modes depend of the sensors currently available at a time t and consequently reduce the number of action possible.

3.3.3 Observation space and observation function of the decision problem

As explained, the set of observation Ω is considered equal to S_v and consequently the observation function is deterministic since $p(o|s'_c, s'_v, a) = 1$ regardless of s'_c, s'_v and a as $o = s'_v$. In contrary to the standard case of POMDP, where Ω is the set of all observation that the UAV could receive and give it partial information about the state s_c. In our case, the UAV do not receive any observation in the classical meaning.

Note: Considering the navigation error propagation – by using a Kalman Filter –, the drone will receive an inaccuracy measurement on the position due to the sensor. This measurement are used to estimate the non observable state variables. Unfortunately, we can't use this measurement as observations, because it is hard to approach a probabilistic function allowing to anticipate the chance to have this or that measure. In this sense, and in the decision problem, s_c is considered as a non observable state variable.

3.3.4 Transition function

Our transition function between our states is composed of two functions :

- $T_{\mathcal{S}_c} : \mathcal{S}_c \times \mathcal{S}_v \times A \times \mathcal{S}_c \to [0; 1]$ a transition function such as :

  $$T_{\mathcal{S}_c}(s_c, s_v, a, s'_c) = f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a) \sim N(\bar{s'_c}, \tilde{\Sigma}'(s_v))$$

  As previously devellopped (Eq : 18) $N(\bar{s'_c}, \tilde{\Sigma}'(s_v))$ is a normal distribution, which designate the probability of a predicted state s'_c, is function of the previous state s_c, of the action a and the noise v (Eq: 2) due to the dynamic system. The next state s'_c depends of the sensors because the sensor-based relative guidance mode require informations from the onboard sensor used during the transition (Eq 15). Moreover, this transition function correspond to process in the motion model (Fig: 1).

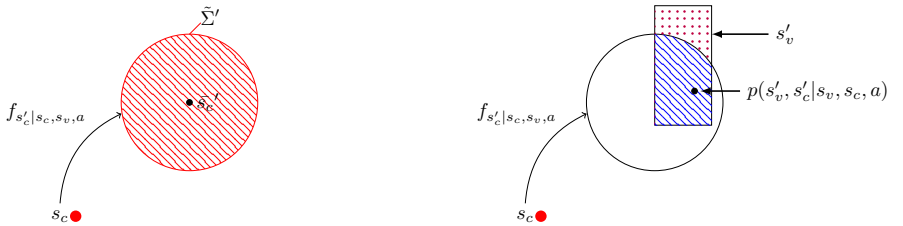
- $T_{\mathcal{S}_v} : \mathcal{S}_v \times \mathcal{S}_c \to [0; 1]$ is a transition function such as:

  $$T_{\mathcal{S}_v}(s'_c, s'_v) = p(s'_v|s'_c)$$

  The function $T_{\mathcal{S}_v}(s'_c, s'_v) = p(s'_v|s'_c)$ represent the transition to s'_v. This transition function depend of the sensor availability map (as indicated in figure 1) and therefore depends only on the next state s'_c. Concretely $p(s'_v|s'_c)$ is a product of probability on the availability (or not) for each sensor. Which give us $p(s'_v|s'_c) = \prod_{i=1}^{|\mathcal{S}_v \setminus P|} p(s'_v(i)|s'_c)$ where $|\mathcal{S}_v \setminus P|$ is the number of sensor on the vehicule and $s'_v(i)$ is a sensor of the vehicule. Noting that P is not included in the calculation of $T_{\mathcal{S}_v}(s'_c, s'_v)$ and in this particular case s'_c represent only the position $\mathcal{X}$.

Then we can tighten the belief state by the probability on the sensor availability. In this intention, we define

$$T(s_v, s'_v, a, s'_c, s_c) = p(s'_v, s'_c|s_v, s_c, a) = p(s'_v|s'_c) f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a)$$



(a) Propagation of s'_c from s_c with an action a (b) Reduction of the belief space in function of s'_v

Figure 5: Example of a transition function

3.3.5 Belief state

The belief state condenses all the accumulated informations during the path of length N, which is the complete information history h defined by :

$$h = \{a_0, o_1, a_1, o_1, ..., a_{N-1}, o_N\}$$

A belief state b can be seen as the probability density function over the possible states at each time step t,$(b(s^t) = f_s(s = s^t), \forall s^t \in S)$. This belief state can be updated after each action a done and at each observation o perceived using the Bayes rule. Thereby, it respects the Markov propriety, since a belief state at the instant t only depends in the belief state at the instant $t-1$, the action done at $t-1$ and the observation observed at t. Thus, we must define properly the belief update of our problem, based on the transition and observation function defined before and the previous belief state.

In this approach, it is considered only the reachable belief from the original belief b^0. The belief state is a probability density function calculated from the prediction and updated with the sensor available. The first belief state b^0 is a confidence ellipse, based on a normal function with the average state and the initial localization error, which has been updated with the sensor available s_v. However it is true only for the first belief, because due to the update with the sensor availability map of our model, the belief will not remain a normal function and in our case, it will be represented by a probability density function. By factoring the state space according to our model we obtain :

$$b = (s_v, b_{s_c}), \text{ with } b_{s_c} = f_{s_c|s_v}(s_c|s_v) \text{ and } b^0_{s_c} = N(\bar{s}_{c_0}, P_0)$$

3.3.6 Belief state update

After each action we must update the belief state. We join this update with the transition function previously explained (Section: 3.3.4), which depend of an Extended Kalman Filter (EKF). This belief update correspond to the architecture defined previously and respresented by the figure 1. The update is done in two steps based on the transition function and consequently on the three sub-functions :

1. From a belief state b_{s_c} and an action a, we will use the state propagation to predict the futur belief state named $b_{s'_c}$. To make the parallel with the diagram of the architecture, this step correspond to the system's dynamic model which take in inputs b_{s_c} and a and return a new partial belief $b_{s'_c}$.

$$b_{s'_c}(s'_c) = f_{s'_c|b,a}(s'_c|b, a) = \int f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a) b_{s_c}(s_c) ds_c$$

2. Then, we can calculate the probability to obtain s'_v based on the belief $b_{s'_c}$:

$$
\begin{aligned}
p(s'_v|b,a) &= \int p(s'_v|s'_c) \int f_{s'_c|s_c,s_v,a}(s'_c|s_c,s_v,a) b_{s_c}(s_c) ds_c ds'_c \\
&= \int p(s'_v|s'_c) f_{s'_c|b,a}(s'_c|b,a) ds'_c \\
&= \sum_{i=0}^{|G|} p(s'_v|s'_c \in c_i) \int_{c_i} f_{s'_c|b,a}(s'_c|b,a) ds'_c \\
&= \sum_{i=0}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b,a)
\end{aligned}
\tag{22}
$$

where c_i corresponding to a i^{th} cell of the sensor availablity map and $|G|$ is the number of cell in the map.

3. The final belief update step correspond to the "Belief state update" of the architecture (Fig: 1). The final $b'^{s'_v}_{s'_c,a}$ is compute in function of s'_v, the completly observable state. Remember that this update depends on the real a priori information of the sensor availability map.

$$
\begin{aligned}
b'^{s'_v}_{s'_c,a}(s'_c) &= f_{s'_c|b,a,s'_v}(s'_c|b,a,s'_v) = \frac{p(s'_v|s'_c) b_{s'_c}(s'_c)}{p(s'_v|b,a)} \\
&= \frac{p(s'_v|s'_c) b_{s'_c}(s'_c)}{\int p(s'_v|s'_c) \int f_{s'_c|s_c,s_v,a}(s'_c|s_c,s_v,a) b_{s_c}(s_c) ds_c ds'_c} \\
&= \frac{p(s'_v|s'_c) b_{s'_c}(s'_c)}{\int p(s'_v|s'_c) f_{s'_c|b,a}(s'_c|b,a) ds'_c} \\
&= \frac{p(s'_v|s'_c) f_{s'_c|b,a}(s'_c|b,a)}{\sum_{i=0}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b,a)} \\
&= \frac{p(s'_v|s'_c) \int f_{s'_c|s_c,s_v,a}(s'_c|s_c,s_v,a) b_{s_c}(s_c) ds_c}{\sum_{i=0}^{|G|} p(s'_v|s'_c \in c_i) p(s'_c \in c_i|b,a)}
\end{aligned}
$$

Therefore the belief state updated is defined as $b'^{s'_v}_a$ and is derived as follow :

$$
b'^{s'_v}_a = (s'_v, b'^{s'_v}_{s'_c,a}) \tag{23}
$$

Note : When we write $f_{s'_c|b,a}(s'_c|b,a)$ it is a misuse of notation, indeed in this case b is a state probability distribution. In POMDP, the belief state is a complete information state that gather all actions performed and observations received and the intiale state distribution. Thus, when we write in function of b, we write in function of all the past.

3.3.7 Cost function of the model

First cost function We must remember that the objective is to find the shortest and safest path. To avoid prioritizing neither the safety nor the length in artificial way, the uncertainty corridor was previously introduced in [16] and briefly described next. Intuitively the corridor is created by a sequence of confidence ellipses. This way, more important is the uncertainty during the path, larger the ellipses will be and consequently the volume of the corridor depends directly of the length of the path and of the dispersion of the uncertainty. Our cost function will be based on this uncertainty corridor between two state $s = (s_v, s_c)$ and $s' = (s'_v, s'_c)$ knowing that the uncertainty is characterized by P wich is contained in s_v.

Formally, the cost between two state can be defined as :

$$
\mathcal{C}(s,s') = \frac{\pi}{6} \|s_c - s'_c\| \cdot \left(u2_{s'} u3_s + u2_s u3_{s'} + 2(u2_s u3_s + u2_{s'} u3_{s'}) \right) \tag{24}
$$

Where $\|s - s'\|$ represent the euclidian length between the two ellipses, and $u2$,$u3$ are the vectors of the ellipses complementary to $u1$ (direction from s to s').

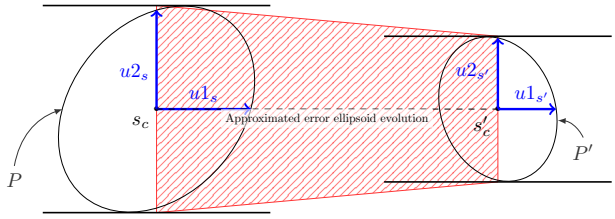


Figure 6: Example of a uncertainty corridor between two states (two dimensions)

Even if the cost function in MOMDPs is defined over $\mathcal{S}$, we calculate the expected cost associated with the transition from b to $b_a^{\prime s'_v}$ such as :

$$\begin{aligned}
\mathcal{C}(b, b_a^{\prime s'_v}) &= \mathbb{E}\left[\mathcal{C}(s,s')|b, b_a^{\prime s'_v}\right] = \int\int \mathcal{C}(s,s') f_{(s_c,s'_c)|b,a,s'_v}(s_c, s'_c|b, a, s'_v) ds'_c ds_c \\
&= \int\int \mathcal{C}(s,s') \frac{f_{(s_c,s'_c,s'_v)|b,a}(s_c, s'_c, s'_v|b,a)}{p(s'_v|b,a)} ds'_c ds_c \\
&= \int\int \mathcal{C}(s,s') \frac{p(s'_v|s'_c) f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a) f_{s_c|s_v}(s_c|s_v)}{p(s'_v|b,a)} ds'_c ds_c \\
&= \frac{1}{p(s'_v|b,a)} \int\int \mathcal{C}(s,s') p(s'_v|s'_c) f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a) b_{s_c} ds'_c ds_c \\
&= \frac{1}{p(s'_v|b,a)} \sum_i p(s'_v|s'_c \in c_i) \int b_{s_c} \int_{c_i} \mathcal{C}(s,s') f_{s'_c|s_c,s_v,a}(s'_c|s_c, s_v, a) ds'_c ds_c
\end{aligned} \tag{25}$$

This cost function is very different from the regular reward function of the POMDP. Classically, a reward function corresponding to $R : S \times A \to \mathbb{R}$, where $R(s,a)$ depends directly of the current state s and the action a done. This way, the function $R(b,a)$ is the average of $R(b,a) = \sum_s R(s,a)b(s)$. It is a linear average that approach the value function of the POMDP with α-vectors based on the PWLC property of it. In our model, the expected cost is no longer a linear average and thus the value function is no more PWLC.

Note This prevents us from using a significant part of the algorithm of the state of the art, such as SARSOP [9], that exploits the piecewise linear and convex (PWLC) representation of the value function to compute the policy.

The second cost function In the previous cost function, we calculate the expected cost (volume of the uncertainty corridor) between two belief states. This expected cost is not easily computable due to the double integral. Consequently we propose a second cost function wich is based on the uncertainty corridor too. But in the contrary to the first function we calculate the cost between the two belief states by computing the uncertainty corridor between the two averaged states.

The cost between two states do not change (see Eq: 24), but the cost between the belief changes :

$$\mathcal{C}(b, b_a^{\prime s'_v}) = \frac{\pi}{6} \|\bar{s_c} - \bar{s_c}'\| \cdot \left(u2_{\bar{s}'} u3_{\bar{s}} + u2_{\bar{s}} u3_{\bar{s}'} + 2(u2_{\bar{s}} u3_{\bar{s}} + u2_{\bar{s}'} u3_{\bar{s}'})\right) = \mathcal{C}(\bar{s}, \bar{s}')$$

Same as before our function est defined by $\mathcal{C} : S_c \times S_c \to \mathbb{R}$. This cost function is easy to calculate and can be considered as an approximation of the first cost function.

3.3.8 Value function

The goal of the agent (in our case an UAV) is to choose, depending on the situations, the actions which will allow it to accomplish the mission. It is necessary to calculate the policy $\pi(b)$, which is a function that takes as input a belief state b and return the action a to be executed such as $\pi(b) \to a$. This

policy is said deterministic and Markovian because there is only one action to each b. We distinguish two types of policies, those that have a finite horizon and those that have an infinite horizion. For the finite horizon policies, we must define an horizon N, at which once the algorithm reaches the policies will be return (that can be seen as a tree). In our case we are interested in the infinite horizon policies. They must possess an horizon far enough such as the returned policies is stationnary (which mean that the expected value do not change for a given ϵ) and that the expected value of the sum of the rewards (or costs) be maximized (respectively minimized).

We define the value function $V^{\pi}(b)$ as being the expected total cost received from starting in b_0 and following the policy π. Moreover we choose a discount factor γ, $0 \leq \gamma < 1$ that will weigth the expected cost over the time, to ensure the convergence of our policies at an infinite horizon [12]. The value function being based on a sum of costs we need to take into account the particularity of our function that need the belief state at time $t+1$ (see section 3.3.7). As precised our cost do not directly depend of the action. But the action have a great impact on it because the uncertainty of a state is represented by a belief state, which depends especially of our navigation and guidance modes and so the action. It is the reason, it is necessary to include in the value function the sensor availability on the entire path. We can define the criterion we want to optimize and the associated optimal policy such as :

$$\begin{aligned}
V^{\pi^*}(b) &= \min_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(s_t, s_{t+1}) | b_t, b^{s_v^{t+1}}_{t+1, a=\pi(b_t)}\right] | b_0 = b\right] \\
&= \min_{a \in A} \mathbb{E}\left[\mathbb{E}\left[\mathcal{C}(s_0, s_1) | b_0, b^{s_v^1}_{1,\pi(b_0)}\right] | b_0\right] \\
&\quad + \mathbb{E}\left[\min_{\pi \in \Pi} \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}(s_t, s_{t+1}) | b_t, b^{s_v^{t+1}}_{t+1, \pi(b_t)}\right] | b_{t=0} = b_1\right] \mid b_0\right] \\
&= \min_{a \in A} \mathbb{E}\left[\mathcal{C}(b = b_0, b^{s_{v_1}}_{1,a}) + \gamma V(b^{s_{v_1}}_{1,a})\right]
\end{aligned}$$

So, one can write

$$V^{\pi^*}(b_0) = \min_{a \in A} \sum_{s_{v_1} \in S_v} p(s_{v_1}|b, a)(\mathcal{C}(b, b^{s_{v_1}}_{1,a}) + \gamma V(b^{s_{v_1}}_{1,a})) \tag{26}$$

We can see that the value function can integrate indifferently the two costs functions that we propose. Indeed, regardless of the cost function used the entries are the same in our practical case.

4 Discussion and perspective

Having a value function which is PWLC is advantageous, there is an important state of art and the theories behind it. Furthermore it allow to represent the policy by a set of vector which is easier and more intuitive. Indeed in contrary to the representation of the policy by a set of all the reachable belief $b \in \mathcal{B}$, it allow to maintain the policy by keeping a set of vectors. Moreover the theorical work on the POMDP [13] ensure that if the value function is PWLC then the value function could be approximate by a PWLC function at each epoch. The algorithms based on this method used this particularity to accelerate the computing of the optimum policy. Consequently, a major part of the algorithm of the state of art are based on this representation. In futur work, the value function is not represented by a set of α-vector.

The algorithm based on PWLC tend to approach the optimum policy, because the optimum is often incomputable due to the curse of dimensionality. To approach and guide reasearch of the best policy, some algorithms use lower and upper bound value functions such as HSVI and SARSOP [14, 9]. This bounds are computed for example using a MDP (in the case of upper bound) that the represent the solution if problem was complety observable, or using other methods. In this work it is not possible to use these algorithms. A solution is to use algorithms that are not based on PWLC value functions like *RTDP-bel* [8] which do not work with α-vector, but keep an hash-table that maps beliefs to values. This algorithm coincides with our problem, but the convergence are not proved. Consequently, the next priority will be to research algorithm that could be adapted or find a new algorithm that can help to solved the problem. Previously, we developed our model in the perspective to calculate a policy in a continuous environment. The goal to work on continuous environment and thus continuous state is to avoid approximation in the planning. In our model, we do not have continuous action which

leads us to important constraints during the path planning. Therefore one important idea we want to incorporate in the future in our model is the continuous actions, which will correspond more to a real application. Continuous actions are not in our first model due to the complexity of the resolution of MOMDP in continuous environment and the complexity induced by continuous actions. Indeed, POMDP with continuous actions are solved with techniques such as particle filter this is why it will be studied later [11].

Références

[1] Markus W. Achtelik, Simon Lynen, Stephan Weiss, Margarita Chli, and Roland Siegwart. Motion- and uncertainty-aware path planning for micro aerial vehicles. Journal of Field Robotics, 31(4):676–698, 2014.

[2] Mauricio Araya-López, Vincent Thomas, Olivier Buffet, and François Charpillet. A closer look at momdps. In Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, volume 2, pages 197–204. IEEE, 2010.

[3] Richard Bellman. A markovian decision process. Indiana Univ. Math. J., 6:679–684, 1957.

[4] Steven J Bradtke and Michael O Duff. Reinforcement learning methods for continuous-time markov decision problems. Advances in neural information processing systems, pages 393–400, 1995.

[5] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Solving continuous pomdps: Value iteration with incremental learning of an efficient space representation. In In Proc. Int. Conf. on Machine Learning, 2013.

[6] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 723–730. IEEE, 2011.

[7] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. The International Journal of Robotics Research, 29(5):485–501, 2010.

[8] Hector Geffner and Blai Bonet. High-level planning and control with incomplete information using pomdps. In Proc. Fall AAAI Symposium on Cognitive Robotics, 1998.

[9] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In Robotics: Science and Systems, volume 2008. Zurich, Switzerland, 2008.

[10] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. The International Journal of Robotics Research, 29(8):1053–1068, 2010.

[11] Josep M Porta, Nikos Vlassis, Matthijs TJ Spaan, and Pascal Poupart. Point-based value iteration for continuous pomdps. Journal of Machine Learning Research, 7(Nov):2329–2367, 2006.

[12] Olivier Sigaud and Olivier Buffet. Processus décisionnels de Markov en intelligence artificielle, volume 1 - principes généraux et applications of IC2 - informatique et systèmes d'information. Lavoisier - Hermes Science Publications, 2008.

[13] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. Operations research, 21(5):1071–1088, 1973.

[14] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In Proceedings of the 20th conference on Uncertainty in artificial intelligence, pages 520–527. AUAI Press, 2004.

[15] Harold Wayne Sorenson. Kalman filtering: theory and application. IEEE, 1985.

[16] Yoko Watanabe, Sylvain Dessus, and Sylvain Fabiani. Safe path planning with localization uncertainty for urban operation of vtol uav. In AHS Annual Forum, 2014.

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) :

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

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

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 :

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

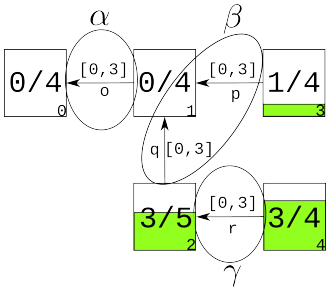


FIGURE 1 – Réseau de l'exemple composé de 5 entités (carrés), 4 actionneurs (arcs) et 3 agents ($\{\alpha, \beta, \gamma\}$) dans son état initial

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.

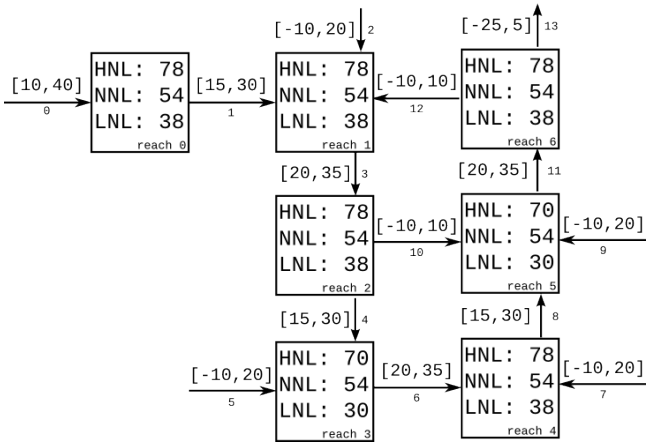


FIGURE 2 – Réseau de voies navigables expérimental composé de 7 biefs (carrés) et de 14 points de transfert (arcs)

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.

The Successor Representation as a model of behavioural flexibility

Alexis Ducarouge, Olivier Sigaud

Sorbonne Universités, UPMC Univ Paris 06, CNRS UMR 7222,
Institut des Systèmes Intelligents et de Robotique, F-75005 Paris, France
olivier.sigaud@isir.upmc.fr +33 (0) 1 44 27 88 53

Abstract :

Accounting for behavioural capabilities and flexibilities experimentally observed in animals is a major issue in computational neurosciences. In order to design a comprehensive algorithmic framework for this purpose, the model-free and model-based reinforcement learning (RL) components are generally taken as reference either in isolation or in combination. In this article, we consider the RL Successor Representation (SR) approach as an alternative. We compare it to the standard model-free and model-based models on three relevant experimental data-sets. These modelling experiments demonstrate that SR is able to account better for several behavioural flexibilities while being algorithmically simpler.

Mots-clés : Reinforcement learning, successor representation, behavioural flexibility, latent learning, internal motivation, policy revaluation.

1 Introduction

Reinforcement learning (RL) is a comprehensive framework that accounts for behavioural and neuronal data of animals addressing temporal difference learning tasks [11]. Because of historical perspectives in psychology [45, 43] and of the somewhat natural division of the RL field, much of the past RL modelling work was striving to disentangle if some aspects of animal behaviour fall under "model-free" or "model-based" mechanisms. The natural reaction to limitations of these quite antagonist systems was to combine them in a dual framework, requiring even an arbitration mechanism as third component [15, 23, 20, 31].

In order to advocate for this algorithmic perspective, the dual system theory has benefited from a lot of experimental works to find potential correlations between neuronal activations and model insights [35, 16, 7, 1, 15, 23, 49]. Nevertheless, even though plenty of correlations have been found (overall, in ventral striatum for reward prediction error of the model-free component and in lateral prefrontal cortex and infraparietal sulcus for state prediction error of the model-based component), results remain really puzzling in so far as there is a lot of crossed significant activations between the two systems which are generally assumed independent [6]. Furthermore, while dopaminergic activity is assumed to correlate with the reward prediction error update of model-free learning, more recent studies measuring dopamine levels show that this activity is much more present during phases in which the model-based system is supposed to be dominant [12]. This essentially challenges the idea that model-based system is inherently not based on temporal-difference learning, and thus that it does not rely on dopaminergic mechanisms. Furthermore, there exists numerous other theories about the role of dopamine which might also be reasonable and which may be inconsistent with the standard reward prediction error interpretation [4, 36].

Given these contradictory findings, some of the researchers supporting these dual system theories have started to advocate for a much more integrated perspective of these two systems [6, 20, 31, 13, 49]. Nonetheless, how they are supposed to interact, and the role of dopaminergic circuitry in such a dual system remain unclear [5].

In this context, the Successor Representation (SR) [8] recently emerged as a promising alternative [51, 10, 14, 5, 39, 2, 29, 34, 22]. The main feature of this approach is that it factors the temporal component of the task into a statistical contingencies RL problem, remodeling the initial estimation problem into a

straightforward reward evaluation which can take into account internal goals of the agent [8]. The learned representation of the task could be viewed as a partial internal model, or cognitive map of it environment [39] which is learned following a model-free fashion using a temporal difference learning rule. This map is then used in an inexpensive way to compute optimal policy given the reward structure of the task. Section 2 of the article introduces the method more formally.

The SR framework is endowed with complementary properties of classical model-based and model-free algorithms. Its learning mechanisms are as biologically plausible and computationally cheap as model-free one; and it allows several behavioural abilities and learning flexibilities which are often considered as sole privileges of model-based one [6, 34], while being simpler. Hence, we investigate in this article if the Successor Representation approach could explain the behavioural data of three classical experiments which emphasize some behavioural abilities as latent learning [3, 33, 45, 19], learning flexibility as immediate reward-policy revaluation [6] and the role of a changing motivation in learning [24, 17, 45]. We compare its capability to quantitatively fit the corresponding experimental data to two model-free (SARSA(λ)) and model-based algorithms as well as a hybrid theory in which both algorithms are computed in parallel and agents can use a weighted combination of the learned action-values to decide what to do [15, 6]. The SR algorithm distinguishes itself by its simplicity and is sometimes the only approach which can account for the observed behaviours.

2 Background and Methods

In this section we introduce the standard RL framework, reminding the model model-free and model-based approaches before presenting the Successor Representation algorithm and its eligibility trace extension.

2.1 RL background

A Markov Decision Process consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a discount factor γ, a reward function $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$, and a transition distribution $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ specifying the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ given action $a \in \mathcal{A}$. An agent chooses actions at each discrete time-step according to a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$.

The goal of the agent is to learn an optimal policy π^* that maximizes the expected cumulative discounted future *value* defined as $V(s_t) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k)]$.

2.2 Model-free and model-based approaches

The classical model-free and model-based approaches rely on different mechanisms; from an algorithmic standpoint the former gradually estimates and caches V directly from sample paths, without building any model of the structure of the MDP. Equivalently, the state-action expected cumulative discounted future *value* Q is generally preferred, defined as

$$Q^\pi(s_t, a_t) = \mathbb{E}_{a_{i>t} \sim \pi} \left[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k) | s_t, a_t \right]. \quad (1)$$

The classical off-policy temporal difference error used in Q-learning to iteratively update Q^π is:

$$\delta = R(s_t) + \gamma \max_a [Q^\pi(s_{t+1}, a)] - Q^\pi(s_t, a_t).$$

The model-based approach rather progressively estimates a model of the environment, namely the transitions $\mathcal{T}$ and rewards $\mathcal{R}$, from sample paths. It then recomputes at each time-step V or Q using either a form of tree search or dynamic programming to infer the optimal policy [41].

2.3 The Successor Representation

The Successor Representation (SR) approach is an approach to RL based on estimating state occupancy relations in a given environment according to the current policy [50, 40, 8, 9]. These relations are stored in a

SR matrix which encodes the expected cumulative discounted future state occupancy M^π of state $\bar{s}$ given a policy π followed from the initial state s_t given the initial action a_t:

$$M^\pi(s_t, \bar{s}, a_t) = \mathbb{E}_{a_{i>t}\sim\pi}\left[\sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{1}[s_k = \bar{s}] | s_t, a_t\right],$$

where $\mathbb{1}[\cdot] = 1$ if its argument is true and 0 otherwise. It implicitly captures contingencies between states given the statistical structure of the MDP and the current policy.

A coefficient of the M^π matrix can be understood as encoding the discounted probability to reach a given state in the future, knowing that the agent takes a given action from a given state and then follows policy π.

As in other RL approaches, several Bellman recursive equations can be defined, one for each future state $\bar{s}$ occupancy prediction problem:

$$M^\pi(s_t, \bar{s}, a_t) = \mathbb{1}[s_t = \bar{s}] + \gamma \mathbb{E}_{a_{i>t}\sim\pi}[M^\pi(s_{t+1}, \bar{s}, a_{t+1})]. \tag{2}$$

One can use this equation to derive the state occupancy temporal difference (TD) error which is the grounding component of the incremental learning algorithm of the SR. Hence, we use the state occupancy TD error to specify the $|\mathcal{S}|$ on-policy update rules of an estimate of the SR - which we simply denote as M:

$$\forall \bar{s} \in \mathcal{S}, \quad M(s_t, \bar{s}, a_t) \leftarrow M(s_t, \bar{s}, a_t) + \alpha(\mathbb{1}[s_t = \bar{s}] + \gamma M(s_{t+1}, \bar{s}, a_{t+1}) - M(s_t, \bar{s}, a_t)).$$

Once the SR is learned, it can be used to infer the action value function defined in (1). Indeed, the expected cumulative discounted future value of choosing action a_t in state s_t given the policy π is given as

$$Q^\pi(s_t, a_t) = \sum_{\bar{s}\in S} M(s_t, \bar{s}, a_t)\widetilde{R}(\bar{s}) \tag{3}$$

where $\widetilde{R}$ is the estimated immediate reward in each state. Estimating the immediate reward function is a simple regression problem.

The key feature of the SR representation is that, given a change in the reward function, the Q-values can be recomputed straightforwardly using (3), giving raise to immediate adaptation of the resulting policy.

2.4 Eligibility traces for the SR

As well as for model-free value-RL approaches, eligibility traces can be naturally defined for the SR approach as a matrix E of size $|\mathcal{A} \times \mathcal{S}|$ associated to a new hyper-parameter λ [37, 40]. Its update rule at each step of the episode is: $\forall s, a \in \mathcal{A} \times \mathcal{S},\ E(s, a) \leftarrow \gamma\lambda E(s, a)$ and $E(s_t, a_t) \leftarrow 1$. Given the current state occupancy TD error written as a vector:

$$\boldsymbol{\delta}_{s_t,a_t} = \begin{vmatrix} \mathbb{1}[s_t = \bar{s_1}] + \gamma M(s_{t+1}, \bar{s_1}, a_{t+1}) - M(s_t, \bar{s_1}, a_t) \\ \vdots \\ \mathbb{1}[s_t = \bar{s_N}] + \gamma M(s_{t+1}, \bar{s_N}, a_{t+1}) - M(s_t, \bar{s_N}, a_t) \end{vmatrix}$$

where $N = |\mathcal{S}|$. The update rule of M is then:

$$\forall s, \bar{s}, a \in \mathcal{S} \times \mathcal{S} \times \mathcal{A},\ M(s, \bar{s}, a) \leftarrow M(s, \bar{s}, a) + \alpha \cdot E(s, a) \cdot \boldsymbol{\delta}_{s_t,a_t}(\bar{s}).$$

3 Experiments

In this section we present the modelling work performed on three classical behavioural experiment. We intend to show that the SR approach can account for several behavioural flexibilities, we compare it with three other RL algorithmic approaches.

3.1 Latent learning (Blodgett, 1929)

The concept of latent learning was introduced by Blodgett [3]. It denotes the ability of animals to learn about the environment structure even when there is no specific goal to achieve nor rewarding events. Once a specific reward is introduced after latent learning, such a latent knowledge appears to enable the agent inferring a good policy significantly faster than without a preliminary latent learning step.

This kind of cognitive aptitude was mainly studied in the first half of the twentieth century, providing a decisive evidence to the "cognitive map" theory thoroughly defended by Tolman [47, 46, 45, 44] and others [33]. This cognitive ability appears crucial to understand the learning and decisional processes in animal and might be a good starting point to reconsider the relevance of dual RL theories [15, 6]. Thus, our first RL model comparison tackles this important issue of latent learning in order to bring decisive quantitative and statistical evidences for a unique comprehensive model.

3.1.1 Outline of the experiment

Blodgett conducted the first experiments on latent learning in rats [3]. He allowed some rats to discover a maze for a few days without rewarding them defining a latent learning period, before introducing a reinforcer at the end of the maze. Overall, he used three groups of 25 to 36 rats in a 6 units T-maze with one-way doors between the T-units (Figure 1).The first two experimental groups did not get any food at the end of the maze for respectively 2 and 6 days. At days respectively 3 and 7, a reward was introduced at the end of the maze; the following day the rats completed the maze much better. The last group was control rats, that always found food at the end of the maze since the first day. For each rat, a maximum of one error was counted each time it took the wrong way in a given T-unit.

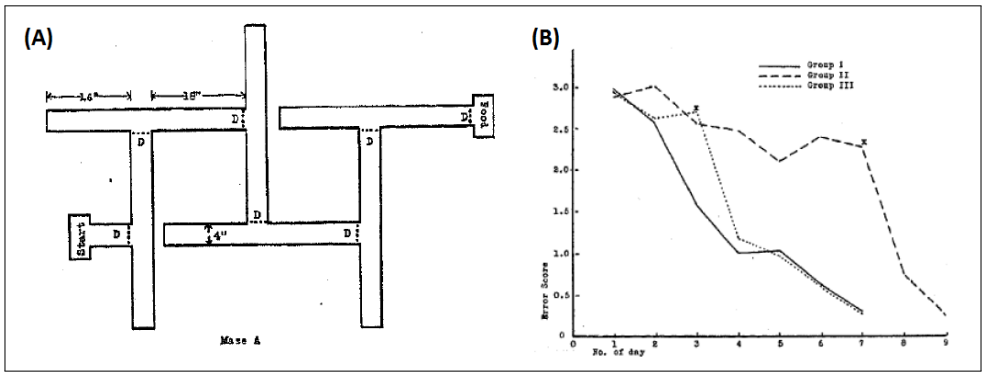


Figure 1: (A) Maze A (one-way doors are depicted by the "D" letters) and (B) results of the main latent learning experience: the curves depict the mean error score of the rats of each group (Blodgett provided no variance information), the "I" letters point the reward introduction (from Blodgett 1929).

The main results obtained by Blodgett are depicted in Figure 1. On average, the control group has a regular learning curve from the beginning of the experiment, while both other groups show a very slow decrease in their error score during the latent learning phase. The day following the reward introduction, one can notice a significant drop of the mean error score; during the following day learning continues, but on a more regular basis. This provides evidence according to Blodgett for the latent learning concept, namely the acquisition of knowledge of the environment before any reinforcement.

3.1.2 Modeling

We model the task as a deterministic Markov Decision Process (MDP) where the state space can be seen as a grid-world made of 6 T-maze as shown in Figure 2. Hence, the agent is able to go back and forth inside each T-unit as the rats sometimes do, contrary to 6 binary forced-choices MDP model designed in other similar modelling experiment [48].

In order to account for the observed mean of 3 errors at the first trial, a small bias is added to the MDP to ensure that an agent making a random walk will do 3 errors in mean at each iteration. The exact value of

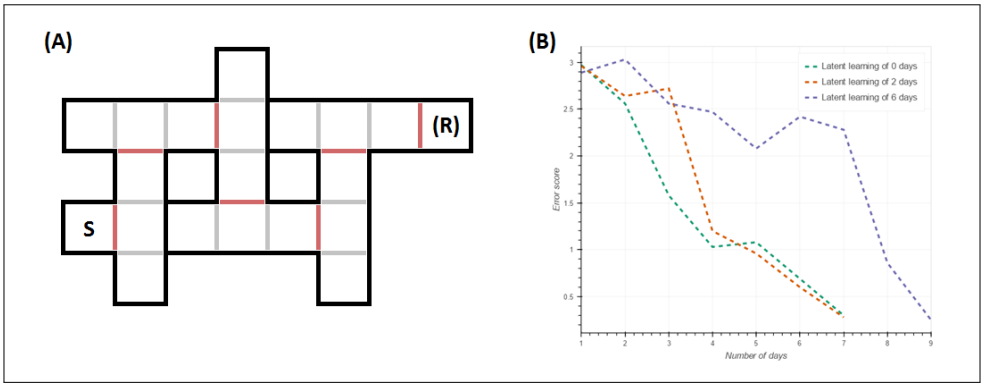


Figure 2: (A) Spatial grid-world representation of the underlying MDP. "S" is the start state, "(R)" the final and potentially rewarded state (once it is rewarded, $r = 1$), the red divisions represent the one-way doors. (B) experimental data of the Blodgett's experiment in a modern fashion (see Figure 1).

the bias is calculated given the structure of the MDP. The rationale behind this mean of 3 errors may be the ability of the rats to sometimes distinguish between dead-ends and one-way doors or their tendency to keep their initial chosen direction.

During latent learning phase, there is a slow decrease of the mean error score despite the absence of any direct reinforcement at the end of the maze. It may be explained by the fact that the rats are eventually fed outside the maze, and that this feeding phase though it occurs long after the experiment may be associated with the end of the maze. Another, but not necessarily incompatible, explanations may be that the end of the maze could be associated with an indirect reward because of the resting time it provides of finishing it, or that the rats become more and more able to distinguish between dead-ends and one-way doors. Thus, and as in [48], we introduced a supplementary hyper-parameter which specifies a tiny pseudo-reward at the end of the maze for the latent learning phase.

We define several models to investigate to what extent they are able to fit the experimental data and transcribe some of their qualitative characteristics. On the one hand, we implement an on-policy model-free RL agent supplemented with eligibility traces (Sarsa(λ)) as well as a model-based RL agent [41]. We also implement the SR approach with eligibility traces as depicted before. Finally, we implement the influential dual model initially proposed by [15] which linearly combines the Q-tables computed according to two competing model-free and model-based components.

The fitting procedure is the following: for each of the models, we proceed with a grid-search over the relevant hyper-parameters on 36 agents. With $\alpha, \lambda, \gamma, \gamma$-MB $\in [0, 0.2, 0.4, 0.6, 0.8, 1]^4$, $\beta \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$, w-Hybrid $\in [0, 10^{-3}, 3.10^{-3}, 10^{-2}, 3.10^{-2}, 10^{-1}, 3.10^{-1}, 1]$ and pseudo-Reward $\in [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$. Thus, each instance (each particular hyper-parameter setting) of each model is run in the modeled environment 36 times, the averaged behaviour is computed. Then, the Akaike Information Criterion (AIC) and its corrected version are used to determine the relative score of each averaged instance given the experimental baseline [30]. Finally, the best instance of each model are compared to infer the best model relatively to the set of models considered.

3.1.3 Results

The results are depicted in Figure 3. They correspond to the best instance obtained for each model given the fitting procedure presented in the previous paragraph. The final AIC scores are shown in Table 1 and the best hyper-parameters are detailed in Table 2. The model-based algorithm immediately computes an optimal policy since it discovers the position of the reward, its decision procedure is randomized to get closer to the experimental data. Nonetheless, this approach barely matches the learning behaviour of rats. Regarding the results of the model-free approach, learning is effective but that it is unable to benefit from the previous latent learning phase.

The algorithm based on the Successor Representation approach obtains the best AIC score. The dual

Table 1: Fitting measure of Blodgett's experiment. Hyper: number of hyper-parameters, LLH: log-likelihood. AIC(c): Akaike Information Criterion (and its corrected version)

Model	Hyper	LLH	AIC	AICc
Model-based	2+1	-32.85	-26.9	-25.58
Model-free(λ)	4+1	-51.46	-41.5	-37.93
Hybrid MB-MF(λ)	6+1	-58.19	-44.2	-36.72
Successor Representation(λ)	**4+1**	**-77.20**	**-67.3**	**-63.67**

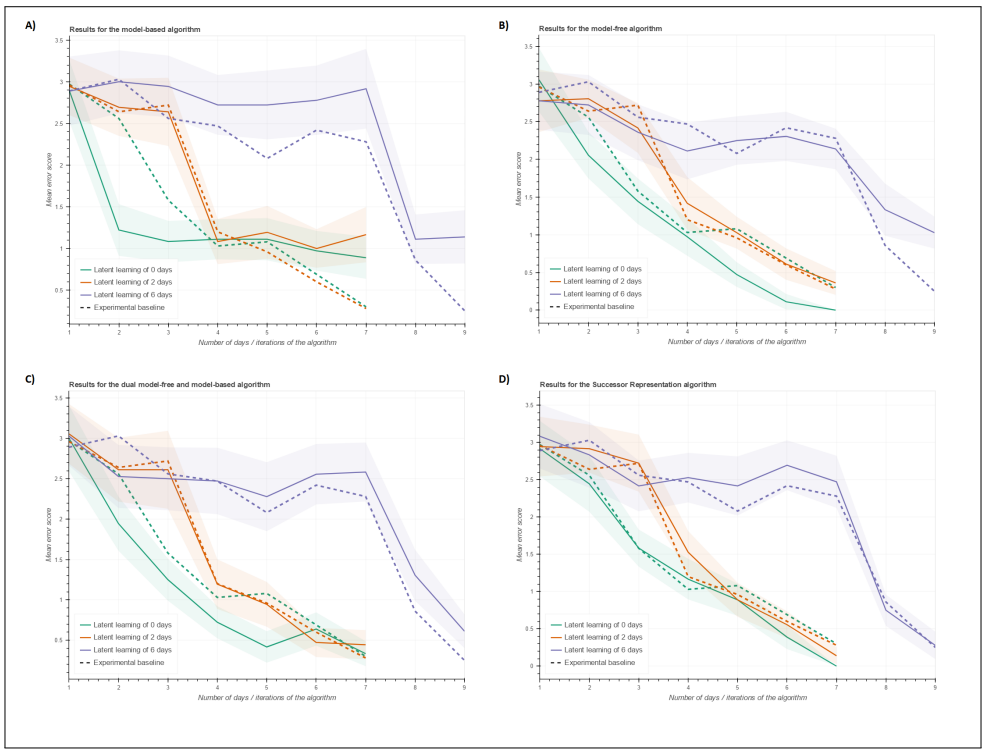


Figure 3: (A) Model-based algorithm. (B) Model-free algorithm with eligibility traces. (C) Hybrid model as in [15, 6]: a linear combination of the model-free(λ) and model-based Q-tables. (D) Successor Representation Algorithm with eligibility traces. The dotted lines represent the experimental data adapted from [3], the solid lines depict the different models (the transparent areas denote the confidence interval of 95%).

model-free and model-based algorithm seems to be far more satisfying than standalone model-free or model-based to fit the experimental data. Nonetheless, the results for the control group are poor and a more detailed analysis shows that the drops of the mean error following the end of the latent learning phase are not significantly different from the greater drop of the control group. That shows that the observed drops are not the expression of a knowledge progressively acquired during the latent learning phase but only the model-based contribution of the behaviour, which is the same, regardless of the latent learning duration.

The statistical analysis made on the top of the best models obtained following the Akaike Information Criterion is based on the non-parametric Mann-Whitney test. Its aim is to compare the distribution of the errors drop following the end of the latent learning period and the greater error drop of the control group to investigate whether they are statistically different or not. The p-values for each condition are given in Table 3.

Table 2: Best hyper-parameters for each model fitting the Blodgett's task.

Model	α	β	λ	γ	w-Hybrid	γ-MB	pseudo-Reward
Model-based		0.1			(1)	0.6	0.01
Model-free(λ)	0.6	0.001	0.2	0.6	(0)		0.0001
Hybrid MB-MF(λ)	0.4	0.001	0.4	0.4	0.003	0.2	0.001
Successor Representation(λ)	1.0	0.01	0.2	1.0			0.001

Table 3: Statistical analysis (Mann-Whitney test) of the error drop for each condition for our Blodgett's experiment modelling.

	p-value		
Model	2-latent	6-latent	Latent
Model-based	0.382	0.387	0.499
Model-free(λ)	0.479	0.155	0.267
Hybrid MB-MF(λ)	0.186	0.308	0.209
Successor Representation(λ)	0.161	**0.00518**	**0.0198**

3.2 Policy revaluation (Daw *et al*, 2011)

The purpose of Daw *et al.* was to investigate the flexibility ability of different RL models, related to observed behaviours in animals. Model-free approach alone is insufficient to account for the observed capacity of human to infer from an unexpected and indirect observation a fresh policy. In light of this, Daw *et al.* designed an experiment to highlight the complementary inputs of model-free and model-based approach [6].

The second goal of this article was to show that the model-free and model-based contribution are clearly dissociable at the neuronal level in the continuity of previous researches [35, 7, 15]. The results of this experiment was largely unexpected; indeed, the alleged contributions of each RL mechanism seem indissociable. These unexpected results opened the way to other researches that defended a more integrated but relatively puzzling vision of the dual theory [20, 12, 13, 31, 42].

Thus, we here intend to show that despite of a complex picture of multiple algorithms operating in parallel and combined in an unclear design, a unique and integrated approach is able to account for this kind of behavioural flexibility. All the more so since SR relies completely on a model-free learning principle.

3.2.1 Outline of the experiment

The behavioural part of the experiment consisted of 17 human subjects who completed a two-stage Markov decision task (Figure 4) for 201 iterations. Each of the states was comprised of two (semantically irrelevant) Tibetan characters which were associated with the two possible actions. The two first-stage actions probabilistically led to the two second-stage states, each action was associated in a privileged fashion with one the two second-stage states (70% of the time versus 30%, *common* and *rare* transitions, as denominated in [6]). These privileged associations were fixed during the experiment. The subjects were informed about the probabilistic structure and ratio of the experiment, but not about the exact mapping between states. Each of the second-stage actions were associated with a payoff probability which evolved independently at each experimental iteration, according to a slow Gaussian random walk (mean 0 and standard deviation of 0.025).

The behavioural questions underlying this experiment was about policy flexibility. For instance, whether an agent who discovers after a *rare* transition that a second-stage state is more likely of being rewarded will adjust his behaviour accordingly. It hence may be more interesting to select the first-stage action which predominantly leads to the state which was rewarded. Model-free and model-based strategies predict different policy revaluation in this case. The former should reinforce the decisional path which leads to the previously experienced reward, even if the underlying transition was unlikely. The latter should infer that the best first-stage decision to reach the rewarded second-stage state is to switch to the other first-stage action, given the higher transition probability.

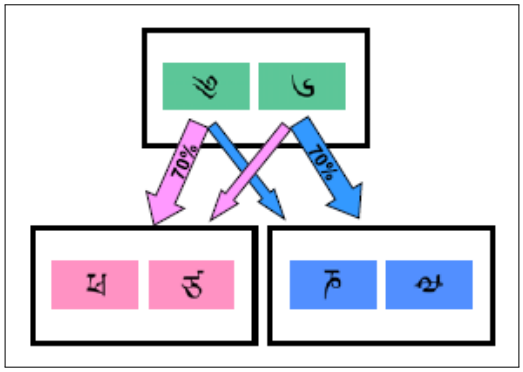


Figure 4: Experimental design of Daw *et al.* (2011). In each state of the modeled MDP (the initial state and the two final rewarded states are represented by bold rectangles) the two possible actions are depicted by the semantically relevant Tibetan characters. The transition probabilities of each first-stage action are illustrated by the arrows. The underlying reward probabilities of each second-stage actions are not shown.

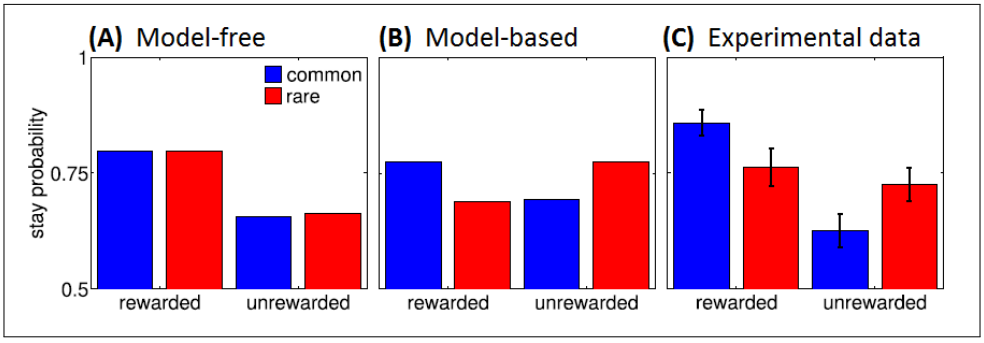


Figure 5: Daw *et al.* (2011) behavioural results (from [6]).

Figure 5 depicts the probability of repeating the previous first-stage choice given the different previously experienced patterns: a *rare* or *common* transition between the first and second-stage states leading to a *rewarded* or *not rewarded* final action. From a general perspective, with the model-free strategy, a finally rewarded choice is reinforced and is more likely to be repeated, irrespective of the probability structure of the task. Following the model-based strategy, both the transition and reward structures impact the first-stage decision in a symmetric design. The experimental data apparently suggests a combination of both properties.

3.2.2 Modeling

In their article, Daw *et al.* advocated for an hybrid algorithm combining a model-free and a model-based component to account for the patterns described in Figure 5. Thus, they adapted the dual approach of Glscher *et al.* [15] to the experimental to fit each of the participant's behaviour, adding numerous hyper-parameters particular to each stage of the Markov task. Unfortunately, they did not show the quantitative results of their hybrid approach as in Figure 5.

Table 4: Best hyper-parameters for each model fitting the Daw *et al.*'s task.

Model	α_1	α_2	β_1	β_2	λ	w-Hybrid	p	α_{reward}
Model-based		1.	1.0	0.001		(1)	1.0	
Model-free(λ)	0.2	0.4	0.0001	0.001	0.5	(0)	-3.	
Hybrid MB-MF(λ)	0.6	0.6	0.1	0.01	0.67	0.3	0	
Successor Representation(λ)	0.2	0.4	0.1	10	0.33		0	0.6

We implement the two-stage MDP as in [6], as well as several small particularities described in the Supplementary material. The MDP's structure is portrayed in the Figure 4. We thus implement the same dual RL model as in [6] (Supplementary material) (a specialized version of the one in [15]) as well as their standalone model-free and model-based versions. With the same hyper-parameters, we also implement the SR approach. The extra hyper-parameters introduced by Daw *et al.* are the following: an learning rate α for each stage, a temperature parameter β for each stage and a perseverance/switching tendency parameter p for the first-stage decision. The SR approach also needs a parameter capturing the learning rate of immediate reward value, we call it α_{reward}. In this set-up, the γ parameter is not used because it becomes redundant with the multiple α parameters. Each of the 4 models are fitted to the experimental data provided in [6] following a grid-search on 100 agents each performing 200 trials. With $\alpha_1, \alpha_2, \alpha_{reward} \in [0, 0.2, 0.4, 0.6, 0.8, 1]^3$, $\beta_1, \beta_2 \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]^2$, $\lambda \in [0, 0.16, 0.33, 0.5, 0, 67, 0.83, 1]$, w-Hybrid $\in [0, 10^{-3}, 3.10^{-3}, 10^{-2}, 3.10^{-2}, 10^{-1}, 3.10^{-1}, 1]$ and $p \in [-1, -0.3, -0.1, 0, 0.1, 0.3, 1]$. The fitting quality measure is the log-likelihood (LLH).

3.2.3 Results

The best fitting hyper-parameters obtained after the grid search procedure are detailed in Table 4, the resulting LLH measure in Table 5 and the best fit graphs are depicted in Figure 6. As anticipated, the model-free and model-based approach are unable to fit correctly the experimental data. Their resulting pattern are relatively similar with the anticipated one as shown in Figure 5. The dual and SR approaches fit the experiment data well, with an advantage for the SR one; interestingly, they do not require the perseveration/switching tendency parameter p to account for the experimental data.

Table 5: Fitting measure of Daw *et al.*'s experiment. Hyper: number of hyper-parameters, LLH: log-likelihood (the greater, the better).

Model	Hyper	LLH
Model-based	4	-22.06
Model-free(λ)	6	-29.13
Hybrid MB-MF(λ)	7	-10.83
Successor Representation(λ)	7	-36.57

3.3 Internal motivation (Leeper, 1935)

One of the main issue trying to model animal behaviour using RL is accounting for recurrent behaviour [41] such as the drinking/eating alternation. When you get replete but thirsty, you do not need to re-learn how to drink, and so on through an entire day. The model-free RL methods are unable to deal with this kind of natural problem, because of the entanglement between the "rewarding values of action outcomes and action outcomes *per se*" [21], that is between learning the contingencies of the environment and learning the value associated with its components.

Here we investigate whether simple and dual models relying on model-free and model-based approach can account for these daily observed capability. We intend to demonstrate that the Successor Representation approach is particularly compelling to account for multiple qualitative facets of such behaviour in addition to being computationally much less costly.

3.3.1 Outline of the experiment

The Leeper's experiment [24] is one the numerous works done on the question of the interaction between learning and internal motivation in the middle of the twentieth century [17, 24, 38, 45, 44], and considered as one of the more reliable set of results [45, 44]. It consists of 23 rats which were trained for 26 days on two mazes structurally similar to the one depicted in Figure 7. They benefited from 3 to 5 training sessions per day following a specific motivational schedule. They thus alternated between thirsty and starving conditions, respectively refereed as squares and circles in the graph of Figure 7B, that constitutes the motivational drive that eventually leads the rats to go alternately in the branch containing food or water. There was several

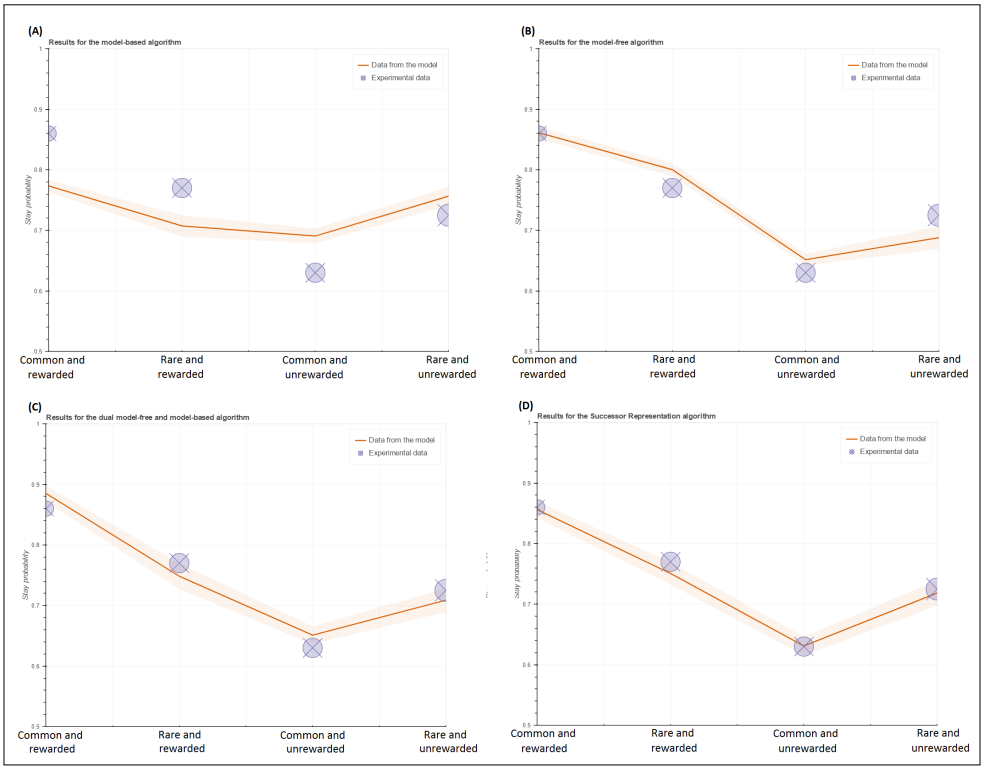


Figure 6: (A) Model-based algorithm. (B) Model-free algorithm with eligibility traces. (C) Hybrid model as in [6]: a linear combination of the model-free(λ) and model-based Q-tables. (D) Successor Representation Algorithm with eligibility traces. The crossed circle represent the experimental data adapted from [6] where the size of the circles reflect the SEM, the solid lines depict the different modelling (the transparent areas denote the confidence interval of 95%).

minor other procedural aspects. Though taking them into account in our modelling work, we do not describe them in detail here, see [24]. An error was counted if a rat went far enough to see if there was water or food at the end of the branch which did not correspond to the internal motivation of the day.

3.3.2 Modeling

We model the task as a Markov Decision Process (MDP) where the state space is seen as an Y-maze as shown in Figure 8. The final states are alternatively rewarded according to Leeper's schedule to simulate the internal motivation associated with each of them since they are respectively associated with the food and water positions.

The four models implemented are the same as in Section 3.1.2 (modeling Blodgett's experiment), except that there is here no additional hyper-parameter of pseudo-reward. As in Section 3.1.2, we proceed with a grid-search over the relevant hyper-parameters, but here on 50 agents.
With $\alpha, \lambda, \gamma, \gamma$-MB $\in [0, 0.2, 0.4, 0.6, 0.8, 1]^4$, $\beta \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10]$, w-Hybrid $\in [0, 10^{-3}, 3.10^{-3}, 10^{-2}, 3.10^{-2}, 10^{-1}, 3.10^{-1}, 1]$ and pseudo-Reward $\in [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$.
As before, we use the AIC(c) criterion to compare models.

3.3.3 Results

The results are depicted in Figure 9. They correspond to the best fitting of each model: the global AIC scores (and their corrected version) are shown in Table 7 and the best hyper-parameters are detailed in Table 6. The model-based algorithm immediately computes an optimal policy when it discovers the whole maze, its decision procedure is randomized to get closer to the experimental data: hence, there is no progressive

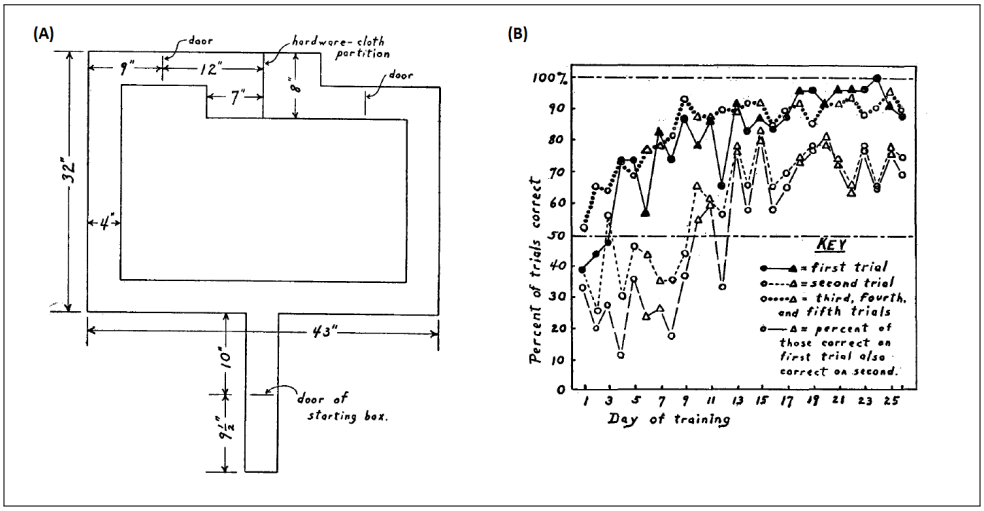


Figure 7: (A) One of the main mazes used by Leeper in his 1935 experiment and (B) the obtained results. We focus here on the solid line which is, according to Leeper himself, the main contribution of his work (from Leeper 1935).

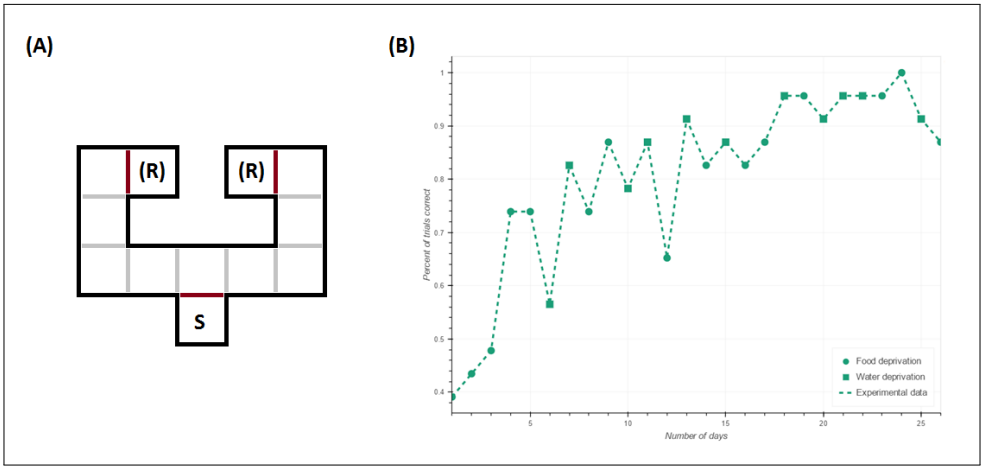


Figure 8: (A) Spatial grid-world representation of the underlying MDP. "S" is the start state, "(R)" the final and potentially rewarding states, the red divisions represent the one-way doors. (B) Experimental data of the Leeper's experiment in a modern fashion.

learning as observed in rats. The model-free approach is unable to take internal motivation into account directly. Furthermore, each time the internal motivational state changes, it needs to dismantle its previous acquired knowledge in order to laboriously re-build a new one from an adverse starting point.

The best fit of the dual algorithm is obtained with a value of 1 for the w-Hybrid parameter, which means that it relies on the model-based component only. The model-free hyper-parameters (α, λ and γ) are thus not taken into account by the algorithm. Then, it makes sense that the (A) and (C) curves of Figure 9 show the same dynamics; actually, as demonstrated in Table 6 the model-based hyper-parameters (β and γ-MB) are the same. Eventually, any model-free contribution in this dual model [15] is counter-productive in the context of this experimental setup.

Ultimately, the SR approach is the only considered model that is able to account for the behavioural flexibility expressed through these experimental data. It progressively learns to accurately choose the right branch of the maze, even though motivational drive alternates. Overall, we also observe that, as in the experimental data, there is slightly more errors the days of motivational switch than the others, which may

Table 6: Best hyper-parameters for each model fitting the Leeper's task.

Model	α	β	λ	γ	w-Hybrid	γ-MB
Model-based		0.001			(1)	0.2
Model-free(λ)	0.4	0.1	0	0.2	(0)	
Hybrid MB-MF(λ)	0.6	0.001	0.2	0.4	1	0.2
Successor Representation(λ)	0.2	0.1	0.2	0.6		

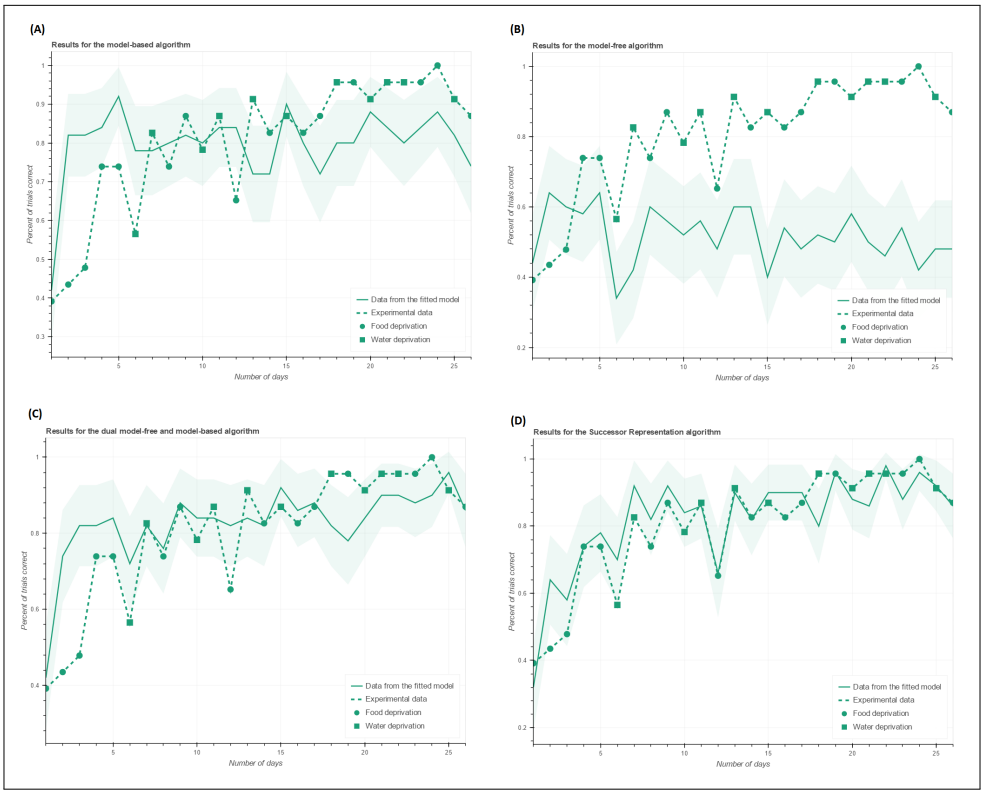


Figure 9: (A) Model-based algorithm. (B) Model-free algorithm with eligibility traces. (C) Hybrid model as in [15, 6]: a linear combination of the model-free(λ) and model-based Q-tables given the proportionality parameter w-Hybrid. (D) Successor Representation Algorithm with eligibility traces. The dotted lines represent the experimental data adapted from [24] with days of food deprivation specified with circle markers and days of water deprivation with square markers The solid lines depict the different models (the transparent areas denote the confidence interval of 95%).

be the expression of what is often refereed as model-free habitual control.

Table 7: Fitting measure of Leeper's experiment. Hyper: number of hyper-parameters, LLH: log-likelihood. AIC(c): Akaike Information Criterion (and its corrected version).

Model	Hyper	LLH	AIC	AICc
Model-based	2	-97.4	-93.4	-92.9
Model-free(λ)	4	-55.8	-47.8	-45.9
Hybrid MB-MF(λ)	6	-110.3	-98.3	-93.9
Successor Representation(λ)	**4**	**-134.2**	**-126.0**	**-124.3**

4 Related work

One of the closest works to ours is [48] in which they present their *SAwSu* algorithm. This work partly relies on the same experimental data such a [6] and [3] in order to evaluate several algorithms, but not always in the same manner as we pointed out in Section 3.1.2 and without modeling dual approaches such as [15] or [6]. Actually, the model they designed integrating associative learning and reinforcement learning may appear very similar to the SR. Indeed, it can be seen as a degeneracy of the SR model: at each time step it updates only one of the prediction component of the SR(λ) according to a delta rule instead of a TD rule. As a consequence, in a 3 states Y-MDP with two final and rewarded states in which one the final state has a greater reward although a tiny transition probability, if this unlikely state is visited just once by chance the policy will be biased toward it indefinitely. In all likelihood they did not know the RL Successor Representation introduced by Dayan in 1992 [8] as they say, speaking about their model, that "there are no other computational cognitive models that learn the spatiotemporal and the reward structures of the environment, and use both in decision-making processes".

Some other works investigate different implementations of the SR approach, trying to bridge the gap of flexibility and computational cost between classical model-free and model-based framework in a very attractive manner [29, 34]. Our work instead focuses on the standalone SR approach using TD learning which is the computationally cheaper and more biologically plausible.

5 General discussion

In this section we recall the main results of the experimental section to draw conclusions and initiate discussions about the interest of the Successor Representation for animal modelling work as well as for its appealing algorithmic flexibility.

In the latent learning experiment (Blodgett, 1929), the results show a clear superiority of the SR model, whose AIC score is smaller than the other models, regardless of the considered AIC score (classical or corrected). This superiority is confirmed by the statistical analysis inspired by Blodgett's one [3]. Hence, the SR model is the only one which shows a significant error reduction after (and only after) latent learning phase, as rats did in the Blodgett's experiment. It thus provides a decisive evidence of the ability of the SR approach to account for the latent learning phenomenon in contrast to classical and hybrid models.

One remaining question might be the meaning of the small decrease observed in rats during the latent learning phase. We hypothesized a kind of indirect or pseudo-reward inferred by the rats at the end of the maze. This might not be entirely satisfactory as the latent learning concept assumes the absence of any direct reward or goal. Nonetheless, others explanations of this decrease such as an increasing discrimination ability of the rat between one-way doors and dead-ends.

In the policy revaluation experiment (Daw *et al.*, 2011), the SR approach is the only approach which always fit the experimental data sufficiently well to always be in the confidence interval. Nevertheless, contrarily to our work on Blodgett's experiment, we found no clear evidence of a qualitative and undeniable difference between the different approaches.

Nonetheless, this approach relies on model-free learning principles such as TD learning; hence, it is almost as computationally cheap as these model-free algorithm. The SR here shows the ability to immediately infer an efficient new policy after reward variation, even when the link between the reward pattern and the subsequent optimal policy is not straightforward. Classical model-free approaches are unable to do so, it is thus typically considered as a model-based capacity [41, 29, 34].

In the internal motivation experiment (Leeper, 1935), the SR approach is the only considered models that is able to decently fit the experimental data. In addition, it shows a qualitative property similar to the one observed during the experiment that seems to go further than just appropriate fitting, appearing to express a kind of model-free habitual control on the top of its great flexibility. It is thus interesting to go back to the vision which Leeper shared in his article about learning and the decisional process, it echoes some of the structural aspects of the SR algorithm [24].

The major phenomenon put forward by Leeper [24] was the behavioural ability of the animal to progressively learn about the structure of the environment, even without reinforcer. And once rewarded, the animal uses this knowledge to immediately adopt the appropriate strategy to navigate toward the goal whose internal value is currently predominant. The primary standpoint of Leeper was to advocate for a clear distinction between *acquisition* and *utilization* of knowledge, which were generally not clearly differentiated. He particularly refereed to previous experiments that showed that "the habits or knowledge in such a case [Ed.: rats were shocked either in the correct or in the incorrect pathway] are independent of the specific motivation which was related to their development" [24]. Thus, the SR learning procedure of the *successor states* matrix could be largely independent from the reward in the environment; it remains linked with the current policy, which is in turn dependent of the rewards, but in an indirect way.

However, Leeper went one step further, defining "the phenomenon of differential motivational control of habit utilization". Which can be explained as "if motivation is importantly related to utilization, one of the prime functions of motivation would seem to be that of determining which associations, or which habits, are to be utilized in any particular situation" [24]. The parallel with the algorithmic structure of the SR approach is striking. Indeed, in SR, motivation is encoded in the reward matrix which is multiplied - at the decision (or utilization) time with the SR matrix to evaluate which action is the more relevant. Furthermore, the SR matrix encodes the state occupancy dynamics learned under the agent policy; thus, it could be viewed as a particular way to encode habits.

6 Conclusion

In this paper, we have studied the capability of the approach to account for behavioural flexibility, fitting it to experimental data published from 3 classical experiments. For each experiment we implemented 3 classical RL algorithms besides the Successor Representation. Our main concern was to compare the SR approach to the model-free and model-based dual models. Thus, we implemented the hybrid mechanism introduced by Glscher *et al.* [15] and adapted by Daw *et al.* [6].

We were aiming to show that the Successor Representation implementation can on its own account for subtle and varied behavioural flexibilities, well beyond the raw reward revaluation paradigm. The results presented here demonstrate that the RL Successor Representation approach is of special interest to account for several flexibility abilities such as a latent learning, immediate policy revaluation and internal motivation variation. Thus, the SR seems to be particularly suited to model animal behaviour. Nevertheless, an important limitation of our study is that we did not have access to individual data to get a more constrained fit of the algorithms. Another point is that we could compare the SR to several other dual mechanisms such as [23, 20, 31]. Conversely, it would be interesting to determine whether the SR can address other behavioral phenomena that have been explained by dual mechanisms, such as the "sign-tracker" "versus goal-tracker" behaviors [25] or negative automaintenance in pigeons [26]. Finally, our study leaves open the question of central interest of the actual neurophysiological implementation of the learning mechanism [14, 29, 39]. Along a different line of thought, the model-free RL framework is getting increasingly used in real-world engineering context [27, 28, 32], sometimes using the SR approach [22, 18, 2]; hence our work provides decisive clues of which flexibility capabilities the SR could provide to these implementations.

References

[1] BALLEINE B. W., DAW N. D. & ODOHERTY J. P. (2008). Multiple forms of value learning and the function of dopamine. *Neuroeconomics: decision making and the brain*, **36**, 7–385.

[2] BARRETO A., MUNOS R., SCHAUL T. & SILVER D. (2016). Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*.

[3] BLODGETT H. C. (1929). The effect of the introduction of reward upon the maze performance of rats. *University of California publications in psychology*.

[4] COLLINS A. G. & FRANK M. J. (2016). Surprise! dopamine signals mix action, value and error. *nature neuroscience*, **19**(1), 3–5.

[5] DAW N. D. & DAYAN P. (2014). The algorithmic anatomy of model-based evaluation. *Phil. Trans. R. Soc. B*, **369**(1655), 20130478.

[6] DAW N. D., GERSHMAN S. J., SEYMOUR B., DAYAN P. & DOLAN R. J. (2011). Model-based influences on humans choices and striatal prediction errors. *Neuron*, **69**(6), 1204–1215.

[7] DAW N. D., NIV Y. & DAYAN P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, **8**(12), 1704–1711.

[8] DAYAN P. (1992). The convergence of td (λ) for general λ. *Machine learning*, **8**(3-4), 341–362.

[9] DAYAN P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, **5**(4), 613–624.

[10] DAYAN P. (2002). Motivated reinforcement learning. *Advances in neural information processing systems*, **1**, 11–18.

[11] DAYAN P. & DAW N. D. (2008). Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, **8**(4), 429–453.

[12] DESERNO L., HUYS Q. J., BOEHME R., BUCHERT R., HEINZE H.-J., GRACE A. A., DOLAN R. J., HEINZ A. & SCHLAGENHAUF F. (2015). Ventral striatal dopamine reflects behavioral and neural signatures of model-based control during sequential decision making. *Proceedings of the National Academy of Sciences*, **112**(5), 1595–1600.

[13] GERSHMAN S. J., MARKMAN A. B. & OTTO A. R. (2014). Retrospective revaluation in sequential decision making: A tale of two systems. *Journal of Experimental Psychology: General*, **143**(1), 182.

[14] GERSHMAN S. J., MOORE C. D., TODD M. T., NORMAN K. A. & SEDERBERG P. B. (2012). The successor representation and temporal context. *Neural Computation*, **24**(6), 1553–1568.

[15] GLÄSCHER J., DAW N., DAYAN P. & O'DOHERTY J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, **66**(4), 585–595.

[16] HOUK J. C., DAVIS J. L. & BEISER D. G. (1995). *Models of information processing in the basal ganglia*. MIT press.

[17] HULL C. L. (1933). Differential habituation to internal stimuli in the albino rat. *Journal of Comparative Psychology*, **16**(2), 255.

[18] JADERBERG M., MNIH V., CZARNECKI W. M., SCHAUL T., LEIBO J. Z., SILVER D. & KAVUKCUOGLU K. (2016). Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

[19] KARIMI Y. & BOLAND H. (1991). Experimental investigation of"latent learning"in mice. *The International Journal of Humanities*, **3**, 18–24.

[20] KERAMATI M., DEZFOULI A. & PIRAY P. (2011). Speed/accuracy trade-off between the habitual and the goal-directed processes. *PLoS Comput Biol*, **7**(5), e1002055.

[21] KOECHLIN E. (2016). Prefrontal executive function and adaptive behavior in complex environments. *Current opinion in neurobiology*, **37**, 1–6.

[22] KULKARNI T. D., SAEEDI A., GAUTAM S. & GERSHMAN S. J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

[23] LEE S. W., SHIMOJO S. & ODOHERTY J. P. (2014). Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, **81**(3), 687–699.

[24] LEEPER R. (1935). The role of motivation in learning: a study of the phenomenon of differential motivational control of the utilization of habits. *The Pedagogical Seminary and Journal of Genetic Psychology*, **46**(1), 3–40.

[25] LESAINT F., SIGAUD O., FLAGEL S. B., ROBINSON T. E. & KHAMASSI M. (2014a). Modelling individual differences in the form of pavlovian conditioned approach responses: a dual learning systems approach with factored representations. *PLoS Comput Biol*, **10**(2), e1003466.

[26] LESAINT F., SIGAUD O. & KHAMASSI M. (2014b). Accounting for negative automaintenance in pigeons: a dual learning systems approach and factored representations. *PloS one*, **9**(10), e111050.

[27] LILLICRAP T. P., HUNT J. J., PRITZEL A., HEESS N., EREZ T., TASSA Y., SILVER D. & WIERSTRA D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[28] MNIH V., KAVUKCUOGLU K., SILVER D., GRAVES A., ANTONOGLOU I., WIERSTRA D. & RIEDMILLER M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[29] MOMENNEJAD I., RUSSEK E. M., CHEONG J. H., BOTVINICK M. M., DAW N. & GERSHMAN S. J. (2016). The successor representation in human reinforcement learning. *bioRxiv*, p. 083824.

[30] MOTULSKY H. & CHRISTOPOULOS A. (2004). *Fitting models to biological data using linear and nonlinear regression: a practical guide to curve fitting*. Oxford University Press.

[31] PEZZULO G., RIGOLI F. & CHERSI F. (2013). The mixed instrumental controller: using value of information to combine habitual choice and mental simulation. *Frontiers in psychology*, **4**, 92.

[32] PRITZEL A., URIA B., SRINIVASAN S., PUIGDOMÈNECH A., VINYALS O., HASSABIS D., WIERSTRA D. & BLUNDELL C. (2017). Neural episodic control. *arXiv preprint arXiv:1703.01988*.

[33] REYNOLDS B. (1945). A repetition of the blodgett experiment on'latent learning.'. *Journal of Experimental Psychology*, **35**(6), 504.

[34] RUSSEK E. M., MOMENNEJAD I., BOTVINICK M. M., GERSHMAN S. J. & DAW N. D. (2017). Predictive representations can link model-based reinforcement learning to model-free mechanisms. *bioRxiv*, p. 083857.

[35] SCHULTZ W. (1998). Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, **80**(1), 1–27.

[36] SCHULTZ W. (2007). Multiple dopamine functions at different time courses. *Annu. Rev. Neurosci.*, **30**, 259–288.

[37] SIGAUD O. & BUFFET O. (2008). Processus décisionnels de Markov en intelligence artificielle.

[38] SPENCE K. W. & LIPPITT R. (1946). An experimental test of the sign-gestalt theory of trial and error learning. *Journal of Experimental Psychology*, **36**(6), 491.

[39] STACHENFELD K. L., BOTVINICK M. & GERSHMAN S. J. (2014). Design principles of the hippocampal cognitive map. In *Advances in neural information processing systems*, p. 2528–2536.

[40] SUTTON R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, **3**(1), 9–44.

[41] SUTTON R. S. & BARTO A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[42] TARTAGLIA E. M., CLARKE A. M. & HERZOG M. H. (2017). What to choose next? a paradigm for testing human sequential decision making. *Frontiers in Psychology*, **8**.

[43] THORNDIKE E. L. (1911). *Animal intelligence: Experimental studies*. Macmillan.

[44] TOLMAN E. C. (1949). There is more than one kind of learning. *Psychological review*, **56**(3), 144.

[45] TOLMAN E. C. *et al.* (1948). Cognitive maps in rats and men.

[46] TOLMAN E. C. & HONZIK C. H. (1930a). ínsightsín rats. **4**, 215–232.

[47] TOLMAN E. C. & HONZIK C. H. (1930b). Introduction and removal of reward, and maze performance in rats. *University of California publications in psychology*.

[48] VEKSLER V. D., MYERS C. W. & GLUCK K. A. (2014). Sawsu: An integrated model of associative and reinforcement learning. *Cognitive science*, **38**(3), 580–598.

[49] WALSH M. M. & ANDERSON J. R. (2014). Navigating complex decision spaces: Problems and paradigms in sequential choice. *Psychological bulletin*, **140**(2), 466.

[50] WATKINS C. J. & DAYAN P. (1992). Q-learning. *Machine learning*, **8**(3-4), 279–292.

[51] WHITE L. M. (1995). *Temporal difference learning: Eligibility traces and the successor representation for actions*. PhD thesis, Citeseer.

Faut-il minimiser le résidu de Bellman ou maximiser la valeur moyenne?

Matthieu Geist[1,2,3], Bilal Piot[4 †], Olivier Pietquin[4 †]

[1] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France.

[2] CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France.

[3] LORIA, CentraleSupélec, Université Paris-Saclay, 57070 Metz, France.

[4] Univ. Lille, CNRS, Centrale Lille, Inria UMR 9189 - CRIStAL, F-59000 Lille, France.

Résumé : Cet article a pour objectif la comparaison tant théorique qu'empirique de deux critères d'optimisation classiques en apprentissage par renforcement : (i) la maximisation de la valeur moyenne et (ii) la minimisation du résidu de Bellman. Pour cela, nous nous plaçons dans le cadre de la recherche directe dans un espace de politiques, le cadre naturel pour la maximisation de la valeur moyenne, et nous proposons une méthode minimisant le résidu $\|T_* v_\pi - v_\pi\|_{1,\nu}$ sur un espace de politiques. Une analyse théorique montre que cette approche bénéficie d'une borne de performance meilleure que la seule connue pour la maximisation de la valeur moyenne, et également meilleure que les bornes de programmation dynamique approchée, respectivement en termes de concentrabilité et d'horizon impliqués. Toutefois, des expériences sur des processus décisionnels de Markov générés aléatoirement, conçues pour étudier l'influence du coefficient de concentrabilité, montrent que le résidu de Bellman est généralement un mauvais substitut à l'optimisation de politique. Comparativement, maximiser la valeur moyenne semble insensible à ce problème. Ces résultats suggèrent que bien que la minimisation du résidu de Bellman permette d'obtenir de bonnes bornes de performance, maximiser directement la valeur moyenne est plus susceptible de produire des algorithmes d'apprentissage par renforcement robustes et efficaces, malgré le manque de compréhension théorique actuelle.

L'article complet, en anglais, est disponible en ligne sur arXiv (Geist *et al.*, 2016).

Remerciements

Matthieu Geist remercie le programme européen FEDER INTERREG VA (projet GRONE) et la Région Grand-Est pour leur soutien financier.

Références

GEIST M., PIOT B. & PIETQUIN O. (2016). Should one minimize the bellman residual or maximize the mean value ? *arXiv preprint arXiv :1606.07636*.

†. Bilal Piot et Olivier Pietquin sont actuellement chez Deepmind, Londres.

Empirical evaluation of a Q-Learning Algorithm for Model-free Autonomous Soaring

Erwan Lecarpentier[1], Sebastian Rapp[2], Marc Melo, Emmanuel Rachelson[3]

[1] ONERA – DTIS (Traitement de l'Information et Systèmes)
2 avenue Edouard Belin, 31000 Toulouse, France
erwan.lecarpentier@isae.fr

[2] TU Delft – Department of Aerodynamics, Wind Energy & Propulsion
Building 62, room B62-5.07, Kluyverweg 1, 2629 HS Delft, Netherlands
s.rapp@tudelft.nl

[3] ISAE Supaero – DISC (Département d'Ingénierie des Systèmes Complexes)
10 avenue Edouard Belin, 31055 Toulouse, France
emmanuel.rachelson@isae.fr

Abstract : Autonomous unpowered flight is a challenge for control and guidance systems: all the energy the aircraft might use during flight has to be harvested directly from the atmosphere. We investigate the design of an algorithm that optimizes the closed-loop control of a glider's bank and sideslip angles, while flying in the lower convective layer of the atmosphere in order to increase its mission endurance. Using a Reinforcement Learning approach, we demonstrate the possibility for real-time adaptation of the glider's behaviour to the time-varying and noisy conditions associated with thermal soaring flight. Our approach is online, data-based and model-free, hence avoids the pitfalls of aerological and aircraft modelling and allow us to deal with uncertainties and non-stationarity. Additionally, we put a particular emphasis on keeping low computational requirements in order to make on-board execution feasible. This article presents the stochastic, time-dependent aerological model used for simulation, together with a standard aircraft model. Then we introduce an adaptation of a Q-learning algorithm and demonstrate its ability to control the aircraft and improve its endurance by exploiting updrafts in non-stationary scenarios.

Mots-clés : Reinforcement learning control, Adaptive control applications, Adaptation and learning in physical agents, UAVs.

1 INTRODUCTION

The number of both civil and military applications of small unmanned aerial vehicles (UAVs) has augmented during the past few years. However, as the complexity of their tasks is increasing, extending the range and flight duration of UAVs becomes a key issue. Since the size, and thus the energy storage capacity, is a crucial limiting factor, other means to increase the flight duration have to be examined. A promising alternative is the use of atmospheric energy in the form of gusts and updrafts. This could significantly augment the mission duration while simultaneously save fuel or electrical energy. For this reason, there is a great interest in the development of algorithms that optimize the trajectories of soaring UAVs by harvesting the energy of the atmosphere. Since the atmospheric conditions are changing over time, it is crucial to develop an algorithm able to find an optimal compromise between exploring and exploiting convective thermal regions, while constantly adapting itself to the changing environment.

In this work we adapt a Q-learning (Watkins & Dayan, 1992) algorithm for this task. Our method is model-free, therefore suitable for a large range of environments and aircraft. Additionally, it does not need pre-optimization or pre-training, works in real-time, and can be applied online. Although the gap towards a fully autonomous physical demonstrator has not been bridged yet, our main contribution in this work is the *proof of concept* that a model-free reinforcement learning approach can efficiently enhance a glider's endurance. We start by reviewing the state of the art in UAV static soaring and thermal modelling in Section 2 and position our contributions within previous related work. Then, in Section 3, we present

the specific atmospheric model we used and its improvements over previous contributions, along with the thermals scenario used in later experiments. Section 4 details the aircraft dynamics model. We introduce our implementation of the Q-learning algorithm in Section 5 and discuss its strengths, weaknesses and specific features. Simulation results are presented in Section 6. We finally discuss the limitations of our approach and conclude in Section 7.

2 RELATED WORK

During the last decade, several possibilities to efficiently utilize atmospheric energy for soaring aircraft have been proposed. For a general introduction to static and dynamic soaring, refer to Chen & McMasters (1981) for instance. For a more specific review on thermal centring and soaring in practice, see Reichmann (1993).

Most approaches to thermal soaring rely on the identification of some model of the wind field surrounding the aircraft. This estimated wind field is then used to track an optimized trajectory inside the thermal or between thermals, using various methods for identification and path planning (Allen, 2005; Allen & Lin, 2007; Lawrance & Sukkarieh, 2011; Lawrance, 2011; Bencatel *et al.*, 2013; Chen & Clarke, 2011; Chakrabarty & Langelaan, 2010). Such approaches demonstrated important energy savings (up to 90% in simulation (Chakrabarty & Langelaan, 2010)) compared to conventional flight. An alternative robust control algorithm (Kahveci & Mirmirani, 2008), based again on a pre-identification of a thermal model showed good results also.

In this paper, we reconsider the possibility to use a *Reinforcement Learning* (RL, Sutton & Barto, 1998) approach to optimize the trajectory. Using RL to exploit thermals has already been examined by Wharington (1998). In this work, a neural-based thermal centre locator for the optimal autonomous exploitation of the thermals is developed. After each completed circle, the algorithm memorizes the heading where the lift was the strongest and moves the circling trajectory towards the lift. However, this thermal locator is too time consuming for real-time on-board applications.

We introduce a *Q-learning* algorithm using a *linear function approximation*, which is simple to implement, demands less computational resources and does not rely on the identification of a thermal model. We empirically evaluate this online learning algorithm (Section 5) by interfacing it with a simulation model that couples the aircraft dynamics (Section 4) with an improved local aerological model (Section 3). We use the model to test our algorithm in several scenarios and show that it yields a significant endurance improvement. Our algorithm's main feature lies in its complete independence of the characteristics of the aerological environment, which makes it robust against model inaccuracy and estimation noise. Moreover, not explicitly estimating the thermal centre position and updraft magnitude saves valuable computational time.

3 ATMOSPHERIC MODEL

Our updraft model expands on that of Allen (2006). His model possesses three desirable features: dependence of the updraft distribution in the vertical direction, explicit modelling of downdrafts at the thermal's border and at every altitude, and finally the use of an environmental sink rate to ensure conservation of mass. Although a complete literature review on modelling the convective boundary layer is beyond the scope of this paper, it should be noted that Allen (2006) is the first reference that includes these three modelling aspects.

We describe a thermal updraft as a symmetrical, bell-shaped distribution as illustrated in Figure 1. This distribution is characterized by two radii r_1 and r_2. At a given altitude z, if r denotes the distance to the thermal center, for $r < r_1$ the updraft has a quasi-constant value of w_{peak}, then for $r_1 < r < r_2$ this value drops smoothly to zero, and between r_2 and $2r_2$ appears a downdraft. The thermal has no influence further than $2r_2$.

The maximum updraft velocity w_{peak} evolves altitude-wise proportionally to $w^* \left(\frac{z}{z_i}\right)^{\frac{1}{3}} \left(1 - 1.1\,\frac{z}{z_i}\right)$, where w^* is an average updraft velocity and z_i is a scaling factor indicating the convective boundary layer thickness. Above $0.9z_i$ all velocities are assumed to be zero.

Finally, based on the conservation of mass principle, an altitude-dependent global environmental sink rate is calculated and applied everywhere outside the thermals. For specific equations, we refer the reader to

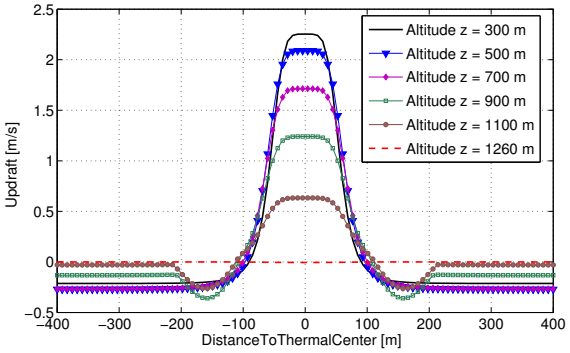


Figure 1: Updraft distribution with altitude

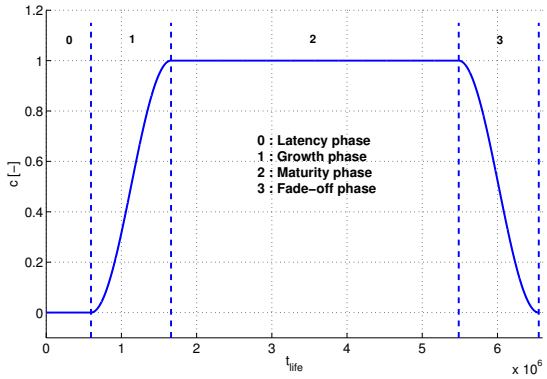


Figure 2: Evolution of the updraft coefficient $c_\xi(t)$

Allen (2006).

We introduce three additional features that bring our simulation model closer to a real-life description, namely thermal drift, life-cycle and noise. First, in order to account for local winds, we let the thermals drift in the horizontal plane with a velocity $(\bar{v}_x, \bar{v}_z)$. Usually, the root point of a thermal is a fixed location and the thermal leans with the wind, so introducing a thermal drift is a poor description of this phenomenon. Nevertheless, for our simulations, it approximates the practical phenomenon of drift given that the aircraft model is reduced to a single point-mass. Thermals also have a finite life. We decompose a thermal's life in a latency phase of duration t_{off} and a growth, maturity and fade-off phase of duration t_{life}. After $t_{off} + t_{life}$ the thermal dies. The life-cycle of a thermal is described by the updraft coefficient $c_\xi(t)$ shown in Figure 2, using a shape parameter ξ. This $c_\xi(t)$ coefficient is used as a multiplier on the total updraft. Finally, it is well-known among cross-country pilots that thermals are rarely round and present a great variety of shapes and much noise. In order to account for this fact and to model real-life uncertainties we added a Gaussian distributed noise n to the wind velocity.

We maintain a constant number N of thermals in the flight area, although some might be in their latency phase. Consequently, whenever a thermal dies, a new thermal is generated with randomly drawn parameters $\{x_{th}, y_{th}, w^*, z_i, \bar{v}_x, \bar{v}_y, t_{off}, t_{life}, \xi\}$.

4 AIRCRAFT MODEL

To model the dynamical behaviour of our aircraft, we used the equations derived by Beeler *et al.* (2003), which consider the aircraft as a point-mass, 6 degrees of freedom system, and take into account the three

dimensional wind velocity vector of the atmosphere as well as a parametric model for the aircraft's aerodynamics. Let m be the glider's mass and g the gravity acceleration. The used variables are:

- x, y, z the coordinates in the earth frame;
- V the absolute value of the aircraft's velocity in the earth frame;
- γ the angle of climb;
- χ the course angle;
- α the angle of attack;
- β the sideslip angle;
- μ the bank angle;
- L, D and C the lift, drag and lateral force.

The corresponding equations are described below:

$$\begin{aligned}
\dot{x} &= V\cos(\chi)\cos(\gamma) \\
\dot{z} &= V\sin(\gamma) \\
\dot{y} &= V\sin(\chi)\cos(\gamma) \\
\dot{V} &= -\frac{D}{m} - g\sin(\gamma) \\
\dot{\gamma} &= \frac{1}{mV}\left(L\cos(\mu) + C\sin(\mu) - \frac{g}{V}\cos(\gamma)\right) \\
\dot{\chi} &= \frac{1}{mV\cos(\gamma)}\left(L\sin(\mu) - C\cos(\mu)\right)
\end{aligned}$$

The first three equations describe the kinematics and position rates in the earth frame. The last three equations define the dynamics of the glider aircraft. For a detailed presentation of the aerodynamic parameters and forces, we refer the reader to Beeler *et al.* (2003). Adopting this modelling directly implies taking the three angles α, β and μ as control variables. Indeed the lift force depends on the bank angle, while the drag and lateral force depend on the three angles. For simplicity of notations we omitted to write this dependency in the model's equations. The choice of the state and action spaces considered by the controller is discussed in Section 5.2.

5 ADAPTIVE CONTROLLER

5.1 Q-learning

RL (Sutton & Barto, 1998) is a branch of Discrete-time Stochastic Optimal Control that aims at designing optimal controllers for non-linear, noisy systems, using only interaction data and no *a priori* model. The only hypothesis underlying RL algorithms is that the system to control can be modelled as a Markov Decision Process (MDP, Puterman, 2005), even if this model is not available. An MDP is given by a set of system states $s \in S$, a set of control actions $a \in A$, a discrete-time transition function $p(s'|s,a)$ denoting the probability of reaching state s' given that action a was undertaken in state s, and finally a reward model $r(s,a,s')$ indicating how valuable the (s,a,s') transition was with respect to the criterion one wants to maximize.

The overall goal of an RL algorithm is to derive an optimal control policy $\pi^*(s) = a$ that maximizes the expected cumulative sum of rewards $\mathbb{E}\left(\sum_{t=0}^{\infty}\eta^t r_t\right)$ from any starting state s ($\eta \in [0;1[$ being a discount factor over future rewards). We focus on model-free RL algorithms that do not commit to the knowledge of the transition and reward models of the underlying MDP but use *samples* of the form (s,a,r,s') to learn an optimal policy. In our case, that means that an RL algorithm controlling the glider with an overall goal of gaining energy will use sensor data to build π^* online, without relying on a (possibly approximate) model of the atmosphere, or the aircraft's flight dynamics.

Q-learning, introduced by Watkins & Dayan (1992), is one of the most simple and popular online RL algorithms. It aims at estimating the optimal action-value function $Q^*(s,a)$ in order to asymptotically act optimally. This function denotes the expected gain of applying action a from state s, and then applying an optimal control policy π^*:

$$Q^*(s,a) = \mathbb{E}\left(\sum_{t=0}^{\infty} \eta^t r_t | s_0 = s, a_0 = a, a_t = \pi^*(s_t)\right)$$

The key idea behind Q-learning is that the optimal action in state s is the one that maximizes $Q^*(s,a)$. Thus the optimal policy is greedy with respect to Q^* in every state. Estimating Q^* from (s,a,r,s') samples is a stochastic approximation problem which can be solved with a procedure known as *temporal differences*. The Q-learning algorithm is summarized in Algorithm 1.

Algorithm 1: Q-learning

Initialize $Q(s,a)$ for all $(s,a) \in S \times A$,
$s_t \leftarrow s_0$.
repeat
  Apply $a_t = \arg\max_{a \in A} Q(s_t,a)$ with probability $1-\epsilon_t$, otherwise apply a random action a_t
  Observe s_{t+1} and r_t
  $\delta_t = r_t + \eta \max_{a' \in A} \left(Q\left(s_{t+1}, a'\right)\right) - Q\left(s_t, a_t\right)$
  Update $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t \delta_t$
  $s_t \leftarrow s_{t+1}$
until *simulation end*

Notice that Q-learning is an *off-policy* method, that is, it estimates Q^* assuming that a greedy policy w.r.t. Q is followed. However, the undertaken action at time t is not necessarily greedy and can be randomly chosen with probability ϵ_t. This strategy, so-called ϵ-*greedy*, allows a wider exploration of the state-action space granting a better estimation of the Q-function. As ϵ_t tends towards zero, if the learnt Q-function has converged to Q^*, the agent tends to act optimally. As long as all state-action pairs are visited infinitely often when $t \to \infty$, Q is guaranteed to converge to Q^* if the sequence of learning rates α_t satisfies the conditions of Robbins & Monro (1951):

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

In the remainder of this section, we discuss how our problem differs from the vanilla MDP and Q-learning frameworks, and the design choices we made to accommodate these differences.

5.2 State and action spaces

Recall that the state of the aircraft, as defined in Section 4, or the state of the atmospheric model (Section 3) are not fully observable to our learning agent. So it would be unrealistic to define the state space S as the observations of these values. Instead, we suppose that a state only defined by $(\dot{z}, \dot{\gamma}, \mu, \beta)$ is accessible and that its dynamics still define an MDP. Such a state is easily measurable with reliable sensors such as pressure sensors, accelerometers or gyrometers. This key assumption is crucial to the success of our method since it reduces the size of the state space, easing the approximation of $Q^*(s,a)$. We shall see later that this choice of state variables has other advantages.

The considered actions consist in directly controlling the aircraft's bank and sideslip angles increments so that the action space is $A = \{-\delta\mu, 0, \delta\mu\} \times \{-\delta\beta, 0, \delta\beta\}$, resulting in $|A| = 9$ different possible actions. We chose the values of $\delta\mu$ and $\delta\beta$ so that, given a certain control frequency, the cumulated effect of a constant action does not exceed the admissible dynamics of the aircraft. This results in a steady state change, representative of the actual behaviour of the actuators.

5.3 Reward model

The goal of our learning algorithm is to maximize the glider's endurance. This boils down to maximizing the expected total energy gain, so we wish that $Q(s,a) = \mathbb{E}\{\text{total energy at } t = \infty\}$. To achieve this, we

choose:

$$r_t = \dot{E}_{aircraft} = \frac{d}{dt}\left(z + \frac{V^2}{2g}\right) \tag{1}$$

Thus we assume that this reward signal r_t is provided to the learning algorithm at each time step, representing the (possibly noisy) total energy rate of the aircraft. Note that the variables $\dot{z}$, V and $\dot{V}$ can be measured online with classical sensors such as a GPS and an accelerometer.

5.4 Convergence in unsteady environments

The previous requirements on ϵ_t and α_t for convergence of Q to Q^* hold if the environment can indeed be modeled as an MDP. However, in the studied case, the environment is non-stationary since the thermals have a time-varying magnitude (thermal coefficient) and location (drift). Moreover, given the choice of state variables, since the agent is blind to its localization, the distribution $p(s'|s,a)$ is not stationary and changes from a time step to the other. Consequently, our learning agent evolves in a constantly changing environment which is *not* a stationary MDP and we actually need to rely on its ability to learn and adapt quickly to changing conditions if we wish to approximate these conditions as quasi-stationary. In order to allow this quick adaptation, we need to force a permanent exploration of the state-action space and to constantly question the reliability of Q. This corresponds to making use of constant α_t and ϵ_t values, which need to be well-chosen in order to retain a close-to-optimal behaviour while quickly adapting to the changes in the environment.

The choice of a simplified low-dimensional state space makes the adaptation to a non-stationary environment feasible. In fact, with our specific choice of state variables, in the short term, the learning agent observes a quasi-constant state $(\dot{z}, \dot{\gamma}, \mu, \beta)$ and the optimal action in this state is almost constant as well. Indeed, the chosen variables evolve slowly through the time, making the evolution of the optimal action value slow as well. This allows to make maximal use of the collected samples since only a local approximation around the current state is required to compute the current optimal action. The success of the method is therefore due to the capacity of the Q-learning algorithm to track the optimal action quickly enough in comparison to the environment's dynamics.

5.5 Linear Q-function approximation

In order to avoid the discretisation of the state space in the description of Q, we adopt a linear function approximation of $Q(s,a)$. We introduce sigmoid-normalized versions of the state space variables and define our basis functions ϕ as the monomials of these normalized variables of order zero to two (15 basis functions). Then, by writing $Q(s,a) = \theta^T \phi(s,a)$, the update equation of Q-learning becomes $\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(s_t, a_t)$. There is abundant literature on choice of feature functions in RL, we refer the reader to Parr *et al.* (2008), Hachiya & Sugiyama (2010), or Nguyen *et al.* (2013) for more details.

To summarize, our glider is controlled by a Q-learning algorithm with fixed learning and exploration rates (α and ϵ) to account for the unsteadiness of the environment. The optimal action-value function Q^* is approximated with a linear architecture of quadratic features defined over a set of observation variables $(\dot{z}, \dot{\gamma}, \mu, \beta)$. Finally, at each time step, the chosen action is picked among a set of 9 possible increments on the (μ, β) current values.

6 SIMULATION RESULTS

We identify three scenarios designed to empirically evaluate the convergence rate of the algorithm and the overall behaviour of the glider. These scenarios take place within a 1100m wide circular flight arena. Whenever the glider exits the arena, an autopilot steers it back in. The aircraft is initialized at $z = 300$m and $V = 15$m/s. According to Allen (2006), we set $w^* = 2.56$m/s and $z_i = 1401$m. The algorithm parameters were $\epsilon = 0.01$; $\alpha = 0.001$; $\eta = 0.99$; $\delta\beta = 0.003\,\text{deg}$; $\delta\mu = 0.003\,\text{deg}$; $\beta_{max} = 45\,\text{deg}$; $\mu_{max} = 25\,\text{deg}$ and the observation frequency is $1kHz$.

The three scenarios are the following: flight in still air without thermal but a noisy downdraft; birth of a thermal along the trajectory; death of a thermal into which the UAV was flying. Qualitatively, the optimal policy in each case is respectively to adopt a straight flight configuration; to circle up within the thermal; and to switch from the circular trajectory to a straight one as in the first case. In each scenario, we refer to the

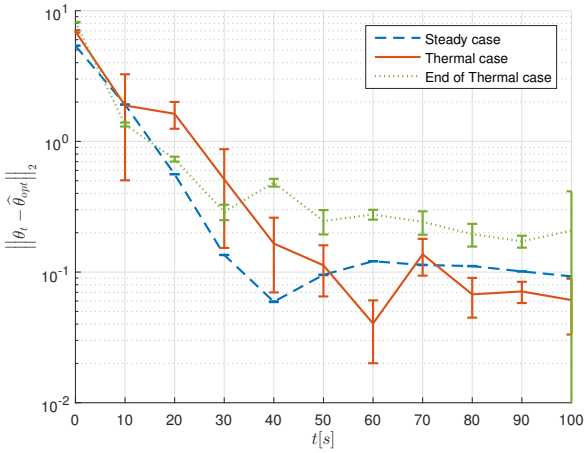


Figure 3: Convergence of the action-value function

optimal action-value function parameters as θ_{opt}. In order to analyse the convergence rate of the algorithm, we built an empirical estimate $\widehat{\theta}_{opt}$ of those parameters with the value they take after the convergence of the algorithm and then compute the quantity $\|\theta_t - \widehat{\theta}_{opt}\|_2$ along 50 roll-outs of the system. The convergence results are reported in Figure 3 where the error bars indicate the standard deviation. One can see that the time required to adjust the parameters to each situation ranges between 30 and 40 seconds, which is compatible with the change rates of the glider's environment. Note in particular that the glider's behaviour might be optimal long before θ converges to θ_{opt} since from a certain state s the optimal action might be selected even if the parameters did not converge. Indeed, what matters is the ranking of the Q-values of the different actions rather than the Q-values themselves. Practically, the configurations vary between the three studied cases and the exploratory feature of the ϵ-greedy policy allows to permanently adapt the Q-function to the situation.

The performance reached by the control algorithm can be measured via the total energy of the aircraft, capturing the reached altitude and the velocity. In the three aforementioned scenarios, the expected results are not the same. Indeed, in a steady atmosphere, the optimal policy only allows to minimize the loss of altitude by setting $\beta = \mu = 0$. Such a configuration is optimal since no thermals can be found and the glider can only maximize its long term energy by flying straight and avoiding sharp manoeuvres. Then, when a thermal is reached, the algorithm's exploratory behaviour allows to captures the information that it is worth changing β and μ, and adapts the trajectory to maximize the long-term return. In the third situation, when the glider flies inside a dying thermal, the algorithm brings back the parameters to a steady atmosphere configuration and again minimizes the expected loss of energy.

Figure 4 shows the evolution of altitude and instantaneous rewards through time in a typical long-term scenario with multiple thermal crossings. Each altitude pike shows the entry of the aircraft into a thermal. First the trajectory is bent in order to maximize the altitude gain and when the thermal dies, the glider goes back to the steady flight configuration. Clearly, each gain-of-altitude phase corresponds to a positive reward and, conversely, a loss-of-altitude phase to a negative one. A 3D display of the trajectory inside a thermal is presented in Figure 5.

The Q-learning controller yields an overall behaviour close to the one of a human pilot while being totally unaware of its own location and of local wind field models. When flying in still air, the glider remains in "flat" flight attitude, thus maximizing its flight time expectancy. Whenever an updraft is spotted, it engages in a spiral, as shown in Figure 4. If the updraft dies, the aircraft comes back to the first configuration. This results in an overall trajectory composed with straight lines and circles as displayed in Figure 6.

Figure 4 also illustrates the reaction times of the glider and the overall command behaviour. It appears that the glider starts to circle up the thermals long before the value function has converged. Similarly, the convergence to a steady air optimal behaviour is faster than the Q-function convergence illustrated on Figure 3. When the glider reaches the thermal's top, the updraft naturally decreases. Consequently one can

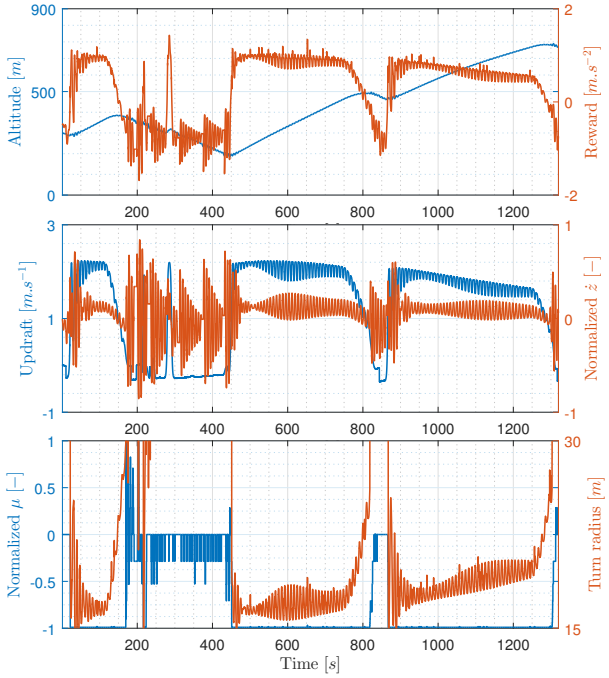


Figure 4: Evolution of the aircraft variables with time

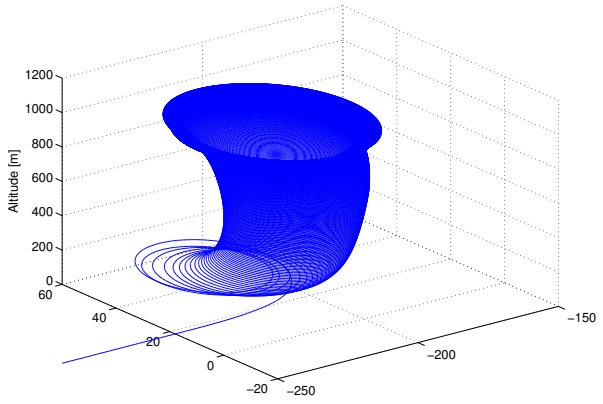


Figure 5: Trajectory of the aircraft inside a thermal

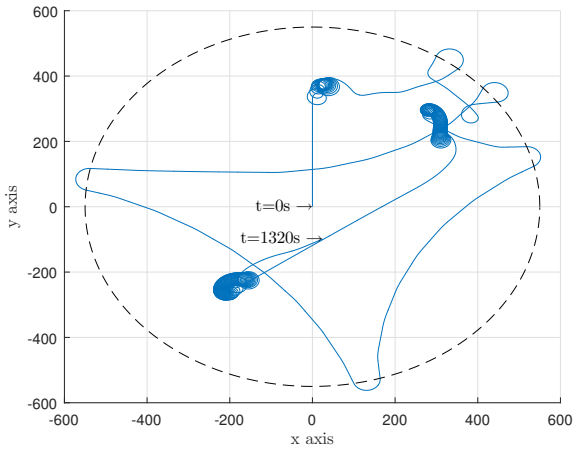


Figure 6: An example of trajectory

notice the reduction of the bank angle (enlargement of the turning radius) computed by the algorithm in order to stay in the thermal while reaching a zero vertical velocity.

7 DISCUSSION AND CONCLUSION

In this paragraph, we discuss the limitations of our contribution, highlight directions for improvements and underline how our results make a difference compared to related work in the literature presented in Section 2. To summarize, we implemented a proof of concept that a computationally light algorithm like Q-learning could be adapted to take into account the time-varying conditions of thermal soaring flight and could make efficient online changes to the control behaviour of an autonomous glider. We take a critical look at this contribution.

First of all, we did not introduce a new RL algorithm *per se*, even though we shortly discuss the question of learning in unsteady environments. The choice of Q-learning is justified by its low computational footprint, despite the existence of a vast literature of efficient algorithms in online RL. Our contributions on the RL side are application-specific: first we justify the need for constant α and ϵ parameters to account for permanent exploration and adaptation in our unsteady environments. Secondly, we make a particular choice of state and action variables, such that, under an optimal policy, the system remains in a quasi-constant state (it would not be the case if the coordinates x, y, z were part of the state space for instance), thus limiting the need for exploration and making the learning process faster. Finally, we introduced a reward model based explicitly on the maximization of the long term energy of the aircraft, thus linking energetic considerations with the definition of the Q-function.

From a low-level control point of view, the hypothesis of a control frequency of 1kHz is somehow questionable and it should be decreased in further developments. We argue however that this frequency is representative of a measurement frequency and should thus still be used to update the Q-function. Exploratory actions artificially account for the information collected due to the noise in wind conditions felt by the aircraft.

The 6 degrees of freedom aircraft model used in the simulation is a classical flight dynamics model that does not take into account the wind gradient in the wingspan direction. This gradient however is known to be a crucial information for human pilots, since it disambiguates whether a thermal centre is on the left or right hand side of the glider. Exploiting such information could bring more efficiency to the glider's control and avoid missing some thermals because the turn was initiated in the wrong direction.

Lastly, in this proof of concept, we based the action space on the aerodynamic angles μ and β as it was done by Beeler *et al.* (2003). Since the Q-learning algorithm aims at maximizing the average energy gain in the long term, it does not improve the short-term stabilization of the longitudinal modes of the aircraft, leading to the oscillations shown in Figure 4. Even though this does not affect the overall long-term energy

gains, a desirable improvement would consist in implementing a low-level stabilization loop (with a PID controller for instance), thus allowing to define the action space using aircraft attitude set points, rather than aerodynamic angles.

Overall, our contribution is three-fold. First we report on how to efficiently adapt a Q-learning algorithm to the non-steady, partially observable, control problem of thermal soaring. Then we empirically evaluate the performance of this algorithm in a rich simulation environment, illustrating how it can be used to improve the energy autonomy of soaring planes. Finally we discuss the strengths and limitations of this approach, thus opening research perspectives on this topic and providing first insights on these perspectives.

References

ALLEN M. J. (2005). *Autonmous Soaring for Improved Endurance of a Small Uninhabited Air Vehicle*. Rapport interne, NASA Dryden Research Center.

ALLEN M. J. (2006). *Updraft Model for development of Autonomous Soaring Uninhabited Air Vehicles*. Rapport interne, NASA Dryden Flight Research Center.

ALLEN M. J. & LIN V. (2007). *Guidance and Control of an Autonomous Soaring UAV*. Rapport interne, NASA Dryden Flight Research Center.

BEELER S., MOERDER D. & COX D. (2003). *A Flight Dynamics Model for a Small Glider in Ambient Winds*. Rapport interne, NASA.

BENCATEL R., DE SOUSA J. T. & GIRARD A. (2013). Atmospheric flow field models applicable for aircraft endurance extension. *Prog. in Aerospace Sciences*, **61**.

CHAKRABARTY A. & LANGELAAN J. (2010). Flight path planning for UAV atmospheric energy harvesting using heuristic search. In *AIAA Guidance, Navigation, and Control Conference*.

CHEN M. & MCMASTERS J. (1981). From paleoaeronautics to altostratus - a technical history of soaring. In *AIAA Aircraft Systems and Technology Conference*.

CHEN W. & CLARKE J. H. A. (2011). Trajectory generation for autonomous soaring UAS. In *17th International Conference on Automation and Computing*.

HACHIYA H. & SUGIYAMA M. (2010). Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, p. 474–489.

KAHVECI N. & MIRMIRANI M. (2008). Adaptive LQ control with anti-windup augmentation to optimize UAV performance in autonomous soaring application. In *IEEE Transactions on Control System Technology*.

LAWRANCE N. (2011). *Autonomous Soaring Flight for Unmanned Aerial Vehicle*. PhD thesis, The University Of Sydney.

LAWRANCE N. & SUKKARIEH S. (2011). Path planning for autonomous soaring flight in dynamic wind. In *IEEE International Conference on Robotics and Automation*.

NGUYEN T., LI Z., SILANDER T. & LEONG T. Y. (2013). Online feature selection for model-based reinforcement learning. In *Int. Conf. on Machine Learning*.

PARR R., LI L., TAYLOR G., PAINTER-WAKEFIELD C. & LITTMAN M. L. (2008). An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*.

PUTERMAN M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

REICHMANN H. (1993). *Cross-Country Soaring*. Soaring Society of America.

ROBBINS H. & MONRO S. (1951). A stochastic approximation method. *Ann. Math. Statist.*, **22**(3), 400–407.

SUTTON R. S. & BARTO A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

WATKINS C. J. C. & DAYAN P. (1992). Q-learning. *Machine Learning*, **8**, 279–292.

WHARINGTON J. (1998). *Autonmous Control of Soaring Aircraft by Reinforcement Learning*. PhD thesis, Royal Melbourne Institute of Technology.

Quadcopter modeling and control

Zoltán Nagy[1], Zsófia Lendek[1]

Technical University of Cluj Napoca, Department of Automation
Memorandumului 28, 400114, Cluj Napoca, Romania {Zoltan.Nagy, Zsofia.Lendek}@aut.utcluj.ro

1 Introduction

Quadcopters have recently been used in many application domains, like surveillance, security or even in transport. One of their usual tasks is to track a trajectory. For example to avoid certain obstacles the drone should follow a well defined path. However tracking any kind of trajectory is a difficult control problem, in many cases a simple linear controller will provide acceptable results. In our project we investigate wether advanced control design technologies can ensure better performances. This paper compares a linear and a fuzzy controller.

Our mathematical model for the UAV is based on the AR Drone 2 quadcopter. Using the Euler-Lagrange approach the dynamic model of the quadcopter can be obtained. Based on (Dydek *et al.*, 2013), the dynamics have the form in (1). In the model, x, y and z represent the position of the drone, while ϕ, θ and ψ represent the Euler angles; the first derivative is used for the linear and angular velocities, while the second derivative for the linear and angular accelerations. We have the state vector $X = \begin{bmatrix}\zeta & \dot{\zeta} & \eta & \dot{\eta}\end{bmatrix}$, where $\zeta = \begin{bmatrix}x & y & z\end{bmatrix}$ and $\eta = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ and the input vector $U = \begin{bmatrix}U_{coll} & U_\phi & U_\theta & U_\psi\end{bmatrix}^T$.

Based on the model in (1), our goal is to track a certain trajectory. The main focus for this tracking was the latitude and the longitude, while for the altitude a fixed reference was used. In order to see the capability of the controllers the trajectory used is the one in Fig. 1. Thus the main task is to follow this trajectory and at the same time keep the altitude at a certain level.

2 Linear quadratic tracking

The first option which can be used is a linear quadratic tracking control. We first linearize the system around an equilibrium point, where the drone is hovering; the controller has the form :

$$u(t) = -KX(t)$$

where K is the controller gain computed by minimizing a quadratic performance function. Because we want to achieve a tracking control, the derivative of the tracking errors are introduced as auxiliary states. Using the linear controller on the nonlinear system (1) a good tracking was achieved for x, y and z, as can be seen in Fig. 1.

3 Takagi-Sugeno (TS) fuzzy control

The dynamics of the UAV described in (1) are highly nonlinear. To actually take into account the nonlinearities we use a nonlinear TS controller (Takagi & Sugeno, 1985). For this, we assume that ϕ, θ, and ψ are varying on the range $\left[-\frac{\pi}{6}, \quad \frac{\pi}{6}\right]$. We have the following approximations :

$$\begin{aligned}\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) &\approx 0.9\theta\\ \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) &\approx -0.95\phi,\\ \cos(\phi)\cos(\theta) &\approx 0.9\end{aligned}$$

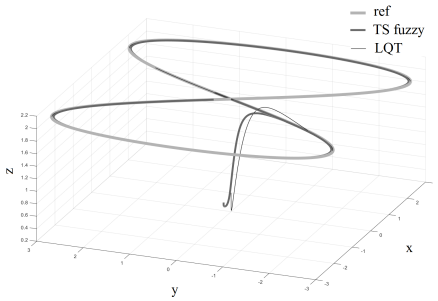


FIGURE 1 – Tracking with TS fuzzy control and LQR

$$
\begin{aligned}
\ddot{x} =& [\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)]\frac{U_{coll}}{m} \\
\ddot{y} =& [\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi))\frac{U_{coll}}{m} \\
\ddot{z} =& -g + \cos(\phi)\cos(\theta)\frac{U_{coll}}{m} \\
\ddot{\phi} =& \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} + \frac{1}{I_x}U_\phi \\
\ddot{\theta} =& \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{1}{I_y}U_\theta \\
\ddot{\psi} =& \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{1}{I_z}U_\psi
\end{aligned} \tag{1}
$$

Even with these approximations the system remains highly nonlinear. Using the sector nonlinearity approach (Ohtake *et al.*, 2001), the approximate system can be converted into a TS fuzzy model. The controller used has the form :

$$u(t) = -K(X(t))X(t)$$

It is a nonlinear control, because K depends on $X(t)$. Based on this, a tracking controller (Lendek *et al.*, 2011) has been computed, which achieved the results in Fig. 1.

4 Conclusion and future work

As it can be seen both options are working, for our trajectory. However the linear one has a strange behaviour in the transient regime, which can be partially seen in Fig. 1. That can be due to the fact that the system is highly nonlinear. Furthermore, no theoretical guarantees exist on the region where the linear controller is working. In our future work we will include in the control also the angle with respect to the z axis, i.e. we want to point in the direction of the trajectory with the UAV.

5 Acknowledgment

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number PN-II-RU-TE-2014-4-0942, contract number 88/01.10.2015, as well as by the Agence Universitaire de la Francophonie (AUF) and the Romanian Institute for Atomic Physics (IFA) under the AUF-RO project NETASSIST.

Références

DYDEK Z., ANNASWAMY A. & LAVRETSKY E. (2013). Adaptive control of quadrotor UAVs : A design trade study with flight evaluations. *IEEE Trans on Control Systems Technology*, **21**(4), 1400–1406.

LENDEK ZS., GUERRA T. M., BABUŠKA R. & DE SCHUTTER B. (2011). *Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models*. Springer Berlin Heidelberg.

OHTAKE H., TANAKA K. & WANG H. (2001). Fuzzy modeling via sector nonlinearity concept. In *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 1, p. 127–132, Vancouver, Canada.

TAKAGI T. & SUGENO M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15**(1), 116–132.

Multi-cycle Coverage for Multi-robot Patrolling - application to data collection in WSNs -

Mihai-Ioan Popescu, Hervé Rivano, Olivier Simonin

University of Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France
firstname.lastname@insa-lyon.fr

Résumé : Patrolling is mainly used in situations where the need of repeatedly visiting certain places is critical. In this paper, we consider a deployment of a wireless sensor network (WSN) that cannot be fully meshed because of the distance or obstacles. Several robots are then in charge of getting close enough to the nodes in order to connect to them, and perform a patrol to collect all the data in time. We discuss the problem of multi-robot patrolling within the constrained wireless networking settings. We show that this is fundamentally a problem of vertex coverage with bounded simple cycles (CBSC). We offer a formalization of the CBSC problem and prove it is NP-hard and at least as hard as the Traveling Salesman Problem (TSP). Then, we provide and analyze heuristics relying on clusterings and geometric techniques. The performances of our solutions are assessed in regards to networking parameters, robot energy, but also to random and particular graph models.

1 INTRODUCTION

Multi-agent patrolling consists in organizing the continuous coverage of an area by several agents, e.g. drones, autonomous vehicles, etc.. Patrolling is predominately used in tasks and applications which require to repeatedly visit certain places. This is the case of coverage, surveillance (e.g. intruder and anomaly detection), data collection, or rescue operations (e.g. after natural disasters) (Rescue, 2001) (Daniel *et al.*, 2009). In this regard, each agent follows its own trajectory. The main objective in patrolling optimization is to minimize the time between two consecutive visits of the same place, called *idleness*. In the multi-agent case, the problem is known to be NP-hard, and for most of the scenarios, efficient solutions are periodic and the trajectories are cycles (Chevaleyre, 2004) (Glad *et al.*, 2008).

In this paper, we consider the problem of multi-robot patrolling constrained by maximum required idleness, with application to collection of data produced by a wireless sensor network (WSN), as in Figure 1. We assume that the WSN cannot be fully meshed because of the inter-node distance or because of obstacles. The challenge is therefore to provide solutions to the general problem of patrolling while taking into account performance metrics induced by the networking constraints.

Having agents patrolling a wireless sensor networks in order to collect the data has been extensively studied in the networking literature, agents being denoted *mobile sinks* (Tunca *et al.*, 2014). These solutions have been introduced for coping with poor connectivity among fixed sensors in difficult radio conditions. The main objective is to improve the network lifetime, which is related to the energy consumed by the nodes for communicating. While most of the literature focuses on networking issues such as address allocation and routing protocols, the mobility of a set of sinks (Marta & Cardei, 2009) and the optimization of their trajectories (Basagni *et al.*, 2008) have also yielded some interest. There is a consensus on the efficiency of using mobile sinks. In our work, we therefore abstract the networking issues into metrics and objectives for the multi-agent patrols with communication capabilities.

In this work, we express the problem of multi-agent patrolling in WSNs with respect to networking requirements and real life resource limits. We take into account issues like limited sensor memory, sensor throughput, data delivery time, limited robot energy and memory, and the finite number of robots. Our main contribution consists in offering, analyzing and testing solutions for the multi-robot patrolling, which respect the requirements of the network as well as robots' limitations, while trying to minimize the number

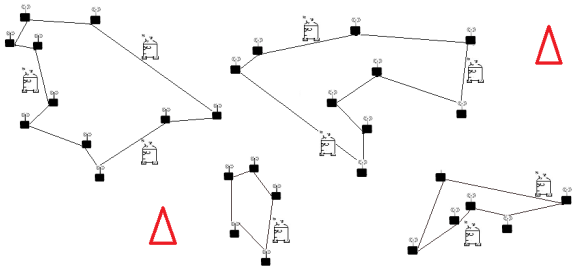


FIGURE 1 – Multi-agent patrolling in a wireless sensor network, for collecting the sensor data. Delivery of the data is made at the red end stations.

of robots to use. We consider that robots are drones, which means they can fly directly between two sensors, so we use Euclidean distances in the sensor graph. First, we prove the necessity of partitioning the area, such that each robot will patrol its own region of the sensor graph. Then, we apply a cyclic patrolling strategy for each region of the graph partition and we explain this choice. For this, we define the CBSC problem (*Covering with Bounded Simple Cycles*) which assumes the covering of the graph nodes with a minimum number of bounded Hamiltonian cycles. We prove this problem is NP-hard and at least as hard as the Traveling Salesman Problem (TSP), which is NP-hard in turn (Papadimitriou, 1977). We also address the following questions : how to partition (or how to cover) the nodes of the graph, to improve the solution for the CBSC ? How to tell whether or not a set of nodes can be covered by an upper bounded Hamiltonian cycle (using a certain TSP heuristic) ? The heuristics that we analyze are state-of-the-art clusterings and our proposed *North-Eastern Neighbor* heuristic, based on geometric computations.

In Section 2, we present the problem setting and related notions, together with the work motivation. After introducing related work on patrolling, graph partitioning and covering in Section 3, we introduce the formalization of the CBSC problem and give insights on its complexity in Section 4. Section 5 presents the adaptation of classic clusterings for CBSC and introduces the North-Eastern Neighbor heuristic. In Section 6, we analyze the performances of these algorithms based on the TSP cycles computed by Christofides heuristics (Christofides, 1976). The results for CBSC on an extensive set of instances are analyzed with respect to the practical networking parameters and different graph models.

2 PROBLEM STATEMENT

In our work, we focus on proposing and analyzing solutions for multi-agent patrolling under WSN constraints, while trying to minimize the number of robots to use. The studied system assumes a Wireless Sensor Network (WSN), with fixed sensors, whose positions are known (though approximate positions are sufficient, within the limit of the radio range). We let $G = (V, E)$ be the complete graph of sensors, where a sensor $s \in V$ is represented by its (x, y) coordinates in the $2D$ plane, and E contains the $\frac{n\times(n-1)}{2}$ edges. The WSN produces data in real time, and some robots are in charge of performing a patrol to collect it in time, while taking into account the limited storage and energy they have. The robots possess a limited wireless communication range, so they can exchange data if it is the case. We assume the robots being drones, which can easily fly directly between any two sensors. Hence, we consider Euclidean distances between any two nodes of G, which is therefore metric. The collected data has to be delivered to an end station as soon as possible (see Fig. 1). Throughout the paper, we use the terminology of *Hamiltonian cycle* instead of *simple cycle*, when it does not cause any confusion. From now on, we use the definitions of (Chevaleyre, 2004).

Definition 1. *Each agent that performs a patrol follows a strategy. An agent strategy is defined as a function $f : \mathbb{N} \to V$, where each value $f(i)$ represents the i^{th} node visited by the agent. A multi-agent strategy is simply defined as a set of single-agent strategies.*

Definition 2. *Assuming a patrolling strategy on G, the idleness of a node represents the time elapsed since the last visit of an agent. For a sensor $s \in V$, its maximum idleness is denoted by $MaxIdl(s)$ (over all performed visits). The worst idleness is the maximum idleness recorded during the patrolling, among all*

the graph nodes, and it is denoted $WI(G)$. The average idleness of a graph is the average of its nodes' maximum idlenesses, and it is denoted $AvgIdl(G)$.

There exist multiple types of idleness that we can consider as evaluation metric (Machado *et al.*, 2003) (Crites & Barto, 1996) (e.g. average idleness, polynomial idleness), but for our sensor graph, we will use the criteria of the worst idleness, introduced in (Machado *et al.*, 2003). Now we state a formal definition of the patrolling problem.

Definition 3. *For a connected graph $G = (V, E)$, the patrolling problem consists in finding a multi-agent strategy for visiting the nodes in V, which minimizes $WI(G)$.*

We now discuss the requirements and the constraints of the multi-robot patrolling in a WSN, and explain how we can solve the related issues. We argue why defining and addressing the CBSC problem is needed.

In practice, sensors have limited storage capacity (usually several kilobytes), so by producing data, its memory gets full at some point. Hence, the sensor storage capacity (denoted S_{mem}) and the throughput of data production by a sensor (denoted S_{thr}) give us an upper bound on the maximum idleness that the patrol should respect, in order to avoid memory overflow and not lose packets : $\frac{S_{mem}}{S_{thr}} \geq MaxIdl(s)$ (the sensor throughput is measured in bytes/sec). Having the bound for $MaxIdl(s)$ and the robot speed (denoted R_{speed}), we can compute the maximum length of the path the robots should walk between two consecutive visits of the same sensor : $\frac{S_{mem}}{S_{thr}} \times R_{speed}$. We could also consider the time spent by an agent for collecting the sensor data, but the sensor radio range permits the robot to collect the data while passing by the sensor and not stopping. Otherwise, it could just be a constant added to each edge (or node) in the graph, which can be neglected therefore.

Partitioning motivation

Two constraints which affect the patrolling are related to data delivery time and to robot storage capacity. We want the data to be delivered as soon as possible to the end station. When dealing with large WSNs, the delay would be high if delivered after patrolling the majority of the sensors. Hence, it is convenient to perform this after patrolling a region of the area. Moreover, robots have limited storage capacity, so they have to discharge the data at some point (through other robots or directly at the end station). This constraint places a bound on the number of sensors a robot should visit, and together with the delivery time constraint, it pushes towards the partitioning of the sensor graph. Since we do not want agents to leave their patrolling region, the data has to be passed between regions, to finally reach the end station. This can be done via inter-region agent rendezvous, which represent the trade-off when partitioning the area (see Fig. 2) : having too many rendezvous does not offer any advantage for data delivering, since they require time also. Hence, we aim at minimizing the number of regions, which in turn minimizes the number of inter-region rendezvous.

Energy consumption is another resource constraint which impacts the patrolling. Robots have limited energy and they need to recharge periodically. On the other hand, we want them to be able to perform at least one patrolling iteration of the sensor graph, before they recharge (i.e. to pass at least one time through each sensor). This constraint may be as well requiring to partition the graph, since the energy may not be sufficient enough for one iteration of the whole sensor graph. We could also consider energy consumption in the communication with the sensor, but it can be a constant added to each edge (or node) of the graph, and therefore can be neglected.

The above arguments express the need of partitioning the sensor graph into regions where the agent patrolling can respect the imposed constraints. For this, we first state a definition of partitioning :

Definition 4. *By partitioning the graph G, we mean covering the set of vertices V with a family of disjoint sets $\{P_i\}_{i=\overline{1,k}}$, where $P_i \subseteq V$, and $\bigcup_{i=1}^{k} P_i = V$. Here, P is called partition for the graph G (see Fig. 2).*

In our setting, we do not benefit of an infinite number of agents. Hence, we minimize resource usage by requiring a single agent for each region of the graph. Our objective is to use as few robots as possible for the patrol. A proof of the following result can be found in (Chevaleyre, 2005).

Theorem 1. *The optimal patrolling strategy in terms of worst idleness, for the single-agent case, is the strategy based on the shortest Hamiltonian cycle of the complete graph.*

Given this theorem, in the context of our setting, the optimal patrolling strategy to apply inside each graph region is the shortest Hamiltonian cycle of the subgraph. So we try to cover the sensor graph with cycles

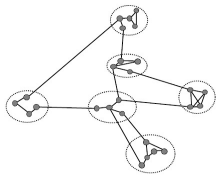


FIGURE 2 – Partition of a graph, with possible rendezvous paths between its regions (Chevaleyre, 2004).

for patrolling, cycles which respect networking and robot constraints. The bound for $MaxIdl(s)$ and the limited robot energy impose a bound on the length of the patrolling cycle. This translates into a problem of vertex covering with bounded simple cycles (CBSC), which we formalize in Section 4.

Before moving forward to related work and problem formalization, we briefly state our contributions : we address the problem of multi-agent patrolling in WSN by defining the problem of vertex Covering with Bounded Simple Cycles (CBSC). We prove it is NP-hard, and as a consequence we use heuristics to partition the graph and to cover it with bounded cycles. In this sense, we consider the *Hierarchical Agglomerative Clustering* and show a way it can be adequately used for specific requirements yielded by the networking metrics and robot limitations. We propose another heuristic which exploits the geometric structure of the graph, the *North-Eastern Neighbour* heuristic. We assess the performance of the partitioning algorithms with respect to the networking metrics, robot energy, and different graph models.

3 RELATED WORK

As presented in the literature (Chevaleyre, 2004) (Portugal & Rocha, 2011), the patrolling of each agent can be performed over the whole area, or with respect to some regions of the area (as in Fig. 3). Cycling strategies and partition based strategies have been studied by Chevaleyre (Chevaleyre, 2004). The experimental results showed that the ones based on cycles performed better than other strategies (such as Cognitive Coordination, or Reinforcement Learning). In addition, the solution improved when the cycling strategies were combined with the partitioning of the longest edges. We make use of this observation in our solution of Section 5. However, no constraints were considered on the patrolling cycles and no bounds on the node idleness.

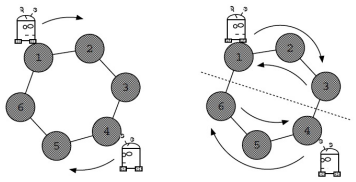


FIGURE 3 – Agents patrolling across the fixed sensors, in two different scenarios : patrolling of the whole graph, patrolling of individual regions (Chevaleyre, 2004).

To the best of our knowledge, there is no preferred partitioning algorithm for the multi-agent patrolling problem, as it is also pointed out in (Chevaleyre, 2004). For particular graphs like grids, Glad et al. (Glad *et al.*, 2008) used their ant-based EVAW algorithm to prove that when the agents self-organize in cycles (i.e. each agent patrols a cycle on the grid), these cycles are necessarily of the same length. Area patrolling under frequency constraints has been studied in (Elmaliach *et al.*, 2009), where all the agents follow one Hamiltonian cycle on the grid. The frequency constraint is respected by placing the agents at equidistant positions on the cycle, covering the whole environment. These papers do not consider a partitioned area, nor a weighted area graph. Partitioning of the area for agent patrolling has been proposed in (Portugal & Rocha, 2010) using the so-called Multilevel Subgraph Patrolling (MSP) algorithm. The authors use the Multilevel Graph Partitioning (Karypis & Kumar, 1998) to section the area graph and offer a decisional scheme which gradually proposes patrolling strategies in the order of their performance (e.g. Eulerian cycle, Hamiltonian cycle, etc.). Here, the partitioning decisions consider the total edge cost of the subgraphs, while trying to

preserve their dimensions. Hence, the approach is not suitable when trying to avoid merging isolated regions of the graph (and long edges), like we initially prefer to do.

The computation of the shortest Hamiltonian cycle of a graph is a hard combinatorial optimization problem, known as the Travelling Salesman Problem (TSP). It is proven to be NP-hard (NP-complete for integer distances (Papadimitriou, 1977)). For this, Christofides (Christofides, 1976) offered a heuristic for a 1.5-approximation of the TSP solution. Vertex covering with a minimum number of bounded cycles has been studied for undirected unweighted graphs (Bekkai *et al.*, 2009). The authors provided an upper bound as a function of graph's order and graph's independence number. The problem becomes much harder, since our setting fundamentally assumes weighted links between nodes (i.e. Euclidean distances).

4 FORMALIZATION (CBSC PROBLEM)

In this section we formalize the CBSC problem and we address it in order to offer an efficient solution for multi-agent patrolling under WSN constraints. We discuss the main elements of the problem, give a hardness proof and point out another two variants of CBSC (minimum-weight CBSC and bounded-size CBSC).

Definition 5. *Let $G = (V, E)$ be a complete undirected graph. A real positive value (of finite precision) is assigned to each edge $e \in E$, representing its length L_e. The graph G is assumed to be metric* [1]*. The problem of Covering with Bounded Simple Cycles requires to cover the set of graph vertices V, with a minimum number of simple cycles, whose lengths are less then or equal to a bound B.*

There is no point in covering the same vertex twice, so we assume the cycles to be vertex-disjoint. Since we want to minimize the number of robots needed for the patrolling and the number of rendezvous between regions, we try to minimize the number of cycles. We evaluate the patrolling efficiency in terms of sensor idleness (concretely, the average graph idleness). In this regard, minimizing the idleness corresponds to the *minimum-weight* version of CBSC (MW-CBSC), where, among all the solutions of the CBSC, we search for the covering with the minimum total length. Related to the finite storage of the robot, avoiding the overflow corresponds to what we call the *bounded-size* CBSC (BS-CBSC). This version of the problem imposes a second bound on the size of each cycle, i.e. the number of sensors (or edges). This constraint is not making the general problem easier, since the cycle length does not depend on it, but the problem becomes trivial if we do not consider anymore the length bound B (or it is big enough to be ignored) : the solution of the BS-CBSC would be $\left\lceil \frac{|V|}{k} \right\rceil$, where $k \in \mathbb{N}$ is the size bound.

TSP and CBSC complexity

The problem of finding the shortest Hamiltonian cycle in a complete weighted graph is NP-hard, and is known as a combinatorial optimization problem, named *Traveling Salesman Problem* (TSP). The Euclidean TSP (i.e. TSP with Euclidean distances between points in the plane) is proven to be NP-complete (Papadimitriou, 1977). It corresponds to our setting (which we could call Euclidean CBSC), since we are aware of the sensors' positions. To denote the TSP solution length, we use the Opt_{TSP} notation.

TSP being NP-hard, in practice, people use certain heuristics to finally obtain desired running times (polynomial time algorithms). Christofides heuristic (Christofides, 1976) runs in time $O(|V|^3)$ and it is proven to deliver Hamiltonian cycles with length no more than $\frac{3}{2} \times Opt_{TSP}$. This heuristic will be used throughout the experimental process.

We now state a theorem regarding the hardness of CBSC. The proof can be found in Annex A, and it is based on the reduction of TSP to CBSC. Its basic idea lies on a "binary search" for the TSP solution. The decision at every step of the search is made with respect to a value of the bound B. The search time is $O(\log Opt_{TSP})$ (linear in the precision of the input). Since the reduction time is polynomial in the size of the input and Euclidean TSP is NP-complete, CBSC is therefore NP-hard.

Theorem 2. *CBSC is NP-hard, being at least as hard as the Traveling Salesman Problem (TSP).*

Corollary 1. *The BS-CBSC and MW-CBSC problems are at least as hard as the CBSC.*

1. graph whose edges respect the triangle inequality ($L_{ij} + L_{jk} \geq L_{ik}$, for all $i, j, k \in V$)

The proof of Corollary 1 can be made easily, since the reduction is straightforward in both cases.

A phase in the solution of CBSC requires that we cover a set of nodes with a bounded simple cycle. For the computation of a simple cycle on the set, we use the Christofides heuristic with polynomial complexity. But because of the hardness of CBSC (Theorem 2), the question is how to partition the graph in a reasonable amount of time (e.g. polynomial time), and consequently, how to tell if a set of nodes can be covered with a bounded simple cycle. We address the questions through heuristics, in the next section.

5 HEURISTICS FOR CBSC PROBLEM

We now present heuristics for the CBSC problem, which we test and compare in Section 6. We focus on the core issue of the problem : the partitioning of the sensor graph for cycle coverage. State-of-the-art heuristics for vertex partitioning include clustering algorithms, such as the hierarchical clusterings. We describe a heuristic of this type and show a way it can be adapted to our needs. Then, we present also our proposed N-EN heuristic.

5.1 Clustering Heuristics

For the partitioning of the sensor graph we consider clustering algorithms, such as the ones based on hierarchies, densities or centroids (e.g. k-Means clustering). Here, we choose the HAC heuristic because of the quadratic running time and the linkage properties it assumes. It does not assume predefined cluster centers (like the k-Means clustering), nor taking exponential time to run (like the divisive hierarchical clustering).

Hierarchical Agglomerative Clustering (HAC)

One popular clustering strategy is the hierarchical one (Jain *et al.*, 1999), which forms the cluster hierarchy either using the top-down (divisive) approach, or the bottom-up (agglomerative) one. HAC uses a proximity matrix and forms clusters incrementally based on the inter-node and inter-cluster distances. Hence, the clusters are formed according to the node connectivity (linkage), which is appropriate when trying to cover as many nodes as possible with an upper bounded cycle. HAC is known to be an efficient clustering algorithm, whose complexity is $O(|V|^2 \log |V|)$. We now show a possible way of using this heuristic for bounded cycle coverage.

HAC for CBSC problem

In order to use HAC clustering for the CBSC problem, we need to tell when a cluster is pertinent to form a cycle on it. So we introduce a condition for the evaluation of clustering : when a set of nodes is evaluated as sufficient enough for obtaining a Hamiltonian cycle with length close to the bound B, then we "freeze" that set (we do not continue to enlarge it). Consequently, it will be considered as a region of the final graph partition. The heuristic used for the evaluation of a set of nodes is a constant time function, which computes on the fly a Hamiltonian cycle on the nodes that are sequentially added to the set : if i and j are the first, respectively the last node added to the cluster, and the current cluster is evaluated to $s \in \mathbb{R}$, then for every new node k, the heuristic will test if $s + L_{jk} + L_{ki} \leq B$. In case the inequality is true, the set of nodes is considered adequate to be covered by a Hamiltonian cycle upper bounded by B, and s is set to $s + L_{jk}$. We use the same technique to evaluate a set of nodes in the case of the N-EN partitioning heuristic.

Cluster evaluation : For the realization of this cluster evaluation, we perform a depth-first search (DFS) of the cluster tree containing the hierarchy. Before we backtrack, we evaluate the cluster lying at that node of the tree. If it is considered adequate by the evaluation heuristic, then we move to its parent node (which represents its merging with another cluster). If the parent node is evaluated as inadequate, then we freeze the previous visited cluster (its child node) and move on to the unvisited children of the parent node. Since the search is a DFS, the algorithm stops when reaching the root of the cluster tree.

5.2 N-EN Heuristic Proposal

In the case of HAC, the clusters are formed in parallel based on the proximity matrix, which can result in too small or too big clusters (before a merging step, and respectively after). In addition, a part of the graph

nodes do not get aggregated in any cluster until late in the clustering process. If before they do, the cluster is frozen by the partitioning heuristic, then the nodes will remain isolated, resulting in one-node regions. If we want to avoid these events, we need another type of heuristic. For this, we propose the *North-Eastern Neighbor* (N-EN) heuristic.

The N-EN heuristic assumes another parameter (L_{max}) which represents the maximum length allowed of any edge in the graph, hence defining the neighbors in the heuristic. We propose the following algorithm (Alg. 1) that consists of the main phases used in computation :

Algorithm 1: The proposed approach for CBSC

1 **Do not consider edges** $e \in E$**, with** $e > L_{max}$**, when forming the neighbor lists**
2 **Obtain the connected components of the graph** G
3 **for** *every connected component of* G **do**
4 Run the N-EN heuristic for vertex partitioning
5 Run Christofides heuristic on every region of the partition

In the first phase of the algorithm, obtaining the neighbor lists according to L_{max} is done in quadratic time $O(|V|^2)$, which makes the proposed heuristic have the same time complexity as HAC. To obtain the connected components of G, we use the Tarjan SCC algorithm (Tarjan, 1971), which runs in linear time $O(|V|)$. After this, the nodes will be partitioned according to the N-EN heuristic, so that we finally obtain Hamiltonian cycles within our constraints. What follows regards the lines 4 and 5 of the Algorithm 1.

Algorithm 2: North-Eastern Neighbor Heuristic (+ Christofides execution)

1 **for** *all connected components (C) of the graph* **do**
2 Sort the nodes in C by their $(x + y)$ sum
3 $S = \emptyset$
4 **while** *there are unvisited nodes inside C* **do**
5 $v =$ choose the most North-Eastern node from C (not visited), which is neighbor to S
6 (mark it as visited)
7 $S = S \cup v$;
8 **for** *all the unvisited neighbors* n *of* v **do**
9 **if** $eval(S \cup \{n\}) \leq B$ **then**
10 $S = S \cup n$ (mark n as visited)
11 // n = last neighbor of v
12 **if** $eval(S \cup \{n\}) > B$ **then**
13 run Christofides heuristic on S, and store the resulted cycle
14 $S = \emptyset$
15 run Christofides heuristic on S, and store the resulted cycle

The choice of starting from the north-eastern part of the area and processing it towards the south-west is due to the incremental coverage that we want to perform, in a rectangular environment. The diagonal coverage allows us to incrementally form cycles that do not stop growing once they hit the borders of the area (this is likely to happen in the case of north departure, when the coverage extends to western or eastern extremities). Concretely, that is minimizing the risk of covering the nodes at the edges of area with small cycles ($<< B$), when they could actually be avoided. The north-east ordering criteria simply considers the sum of the (x,y) coordinates of every sensor in the area. In Algorithm 2, every time the heuristic freezes a set of nodes or finishes visiting the neighbors of a node, it will choose the most north-eastern node to continue. Moreover, in the latter case, it will continue to form the set S starting with the most north-eastern node (from the connected component) that has a neighbor inside S. In this case, we start a new simple cycle on the newly added nodes. Its length is added to the evaluation. This is why we may obtain cycles that exceed B. Otherwise, the heuristic will deliver too many small cycles, while the good ones are the closest to B (to maximize the ratio %BC / TC, as in Section 6). The evaluation function $eval(S)$ is the same function as for HAC, used to evaluate a set of nodes according to the bound B. Sorting the nodes can be done in

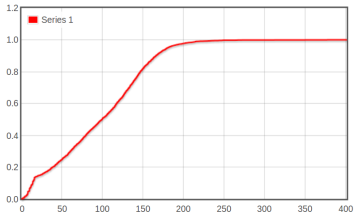


FIGURE 4 – The cumulative distribution function (CDF) of the cycle lengths, obtained for $B = 150$ and 500 random graphs of 100 nodes, using the N-EN heuristic. X-axis : the lengths of the cycles.

	N-EN Heuristic	HAC clustering
%BC	82.27	92.87
#BC | TC	2679 | 3256	3571 | 3845
%BC / TC $\times 10$	0.25	0.24
worst idleness $WI(G)$	401	343
avg idleness $AvgIdl(G)$	96.2	80.1
avg # of iterations	1.5	1.8

TABLE 1 – The results obtained with HAC and N-EN heuristics, on an extensive set of 500 random graphs of 100 nodes, for $B = 150$; %BC = the percentage of bounded cycles (over TC) ; TC = the total number of cycles.

$O(|V| \log |V|)$ and choosing the most North-Eastern node, neighbor to set S, takes at most $O(|V|^2)$ steps for the whole execution of the algorithm. Visiting the neighbors of each node takes also $O(|V|^2)$ time, so the worst case performance of the algorithm is $O(|V|^2)$.

6 PERFORMANCE ANALYSIS

In the experiments, we used the heuristics presented in the previous section (HAC and N-EN), for the partitioning of the sensor graph. For the computation of a Hamiltonian cycle in each region of the graph partition, we used the Christofides heuristic. We also implemented the procedures necessary to generate random positions for the graph vertices in the $2D$ plane of any size or to consider any particular graph as input. The vertex positions have been generated using the random uniform distribution. Moreover, all the experiments have been conducted for an area size of 100×100 meters. For HAC, we used the criteria of complete-linkage clustering (the distance between two clusters $C1$ and $C2$ is the maximum distance between any node in $C1$ and any node in $C2$). We consider that all lengths are measured in meters.

Table 1 presents the results obtained with HAC and N-EN on an extensive set of random graph instances. The results are compared with respect to the percentage of cycles with length less than B (denoted BC), with respect to the number of cycles in the coverage (TC), the worst idleness of the patrolling ($WI(G)$), the average graph idleness ($AvgIdl(G)$) and the average number of iterations a robot would have to patrol (which is actually $B/AvgIdl(G)$). We also evaluate the ratio %BC / TC (since the desired %BC would be 100, while minimizing TC).

While HAC delivers more cycles respecting the bound B, it also delivers a greater number of cycles than the N-EN heuristic. In this case, $\approx 93\%$ of the cycles respect the maximum idleness and energy, but their number is much higher than in the case of N-EN. Hence, we could argue that N-EN performs better in this sense, but if computing the ratio of these metrics (%BC / TC), we observe that the heuristics actually have almost equal performances. The $WI(G)$ and $AvgIdl(G)$ of the N-EN are bigger than the ones of HAC, which may be explained by the fact that N-EN delivered a smaller number of cycles, but with higher lengths. The average number of iterations a robot would patrol is higher in the case of HAC, which implies a lower amount of energy consumption (if recharging after 1 iteration), so HAC performs better here. Yet, regarding the minimization of the number of robots, one would like to obtain a value closer to 1 (but not less), since this means getting cycles close to B (but not higher), and for this, N-EN behaves better (see

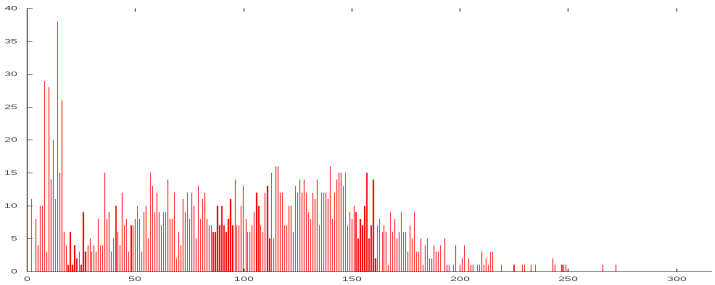


FIGURE 5 – X-axis : the cycle length. Y-axis : the frequency of the cycle length (in terms of number of cycles). The frequency of the cycle lengths, obtained for $B = 150$ and 250 random graphs of 100 nodes, using the N-EN heuristic.

"avg # of iterations" in Table 1). Overall, though it is convenient to obtain more patrolling cycles which respect the data collecting frequency ($\frac{S_{mem}}{S_{thr}}$) and the energy constraint, obtaining a higher number of such cycles means a higher number of graph regions, robots to use and inter-region rendezvous, which does not optimize the objective of CBSC.

In the CDF of Figure 4, one can see the probability of a cycle to have length (C_l) less than the bound $B : P(C_l < B = 150) \approx 80\%$, which corresponds to the data in the Table 1. The CDF increases linearly uniform up to a threshold around the bound $B = 150$. This means that the cycle lengths are nearly uniformly distributed across the interval of $[0, 150]$ (as it can be also seen in Fig. 5). A peak of the CDF is observed for cycles of small lengths, this being a cause of the random graphs, which often contain groups of isolated node.

We now discuss in detail the behavior of the N-EN heuristic. We describe the observations with respect to parameters B and L_{max}.

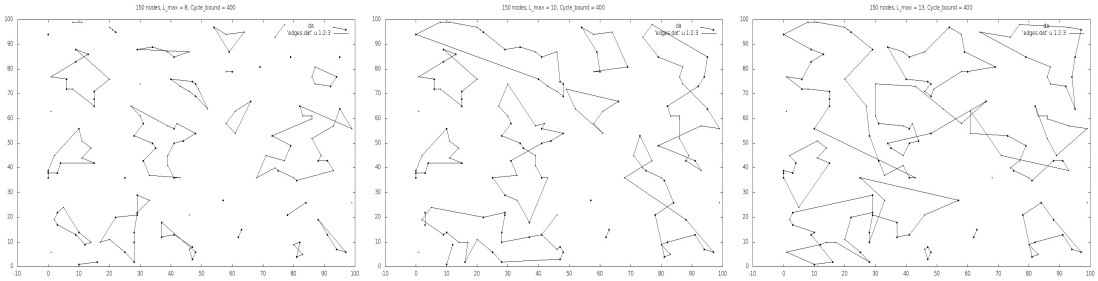


FIGURE 6 – The evolution of the cycle coverage, as L_{max} parameter increases (8, 10 and 13). Scenario for 150 nodes and B fixed to 400.

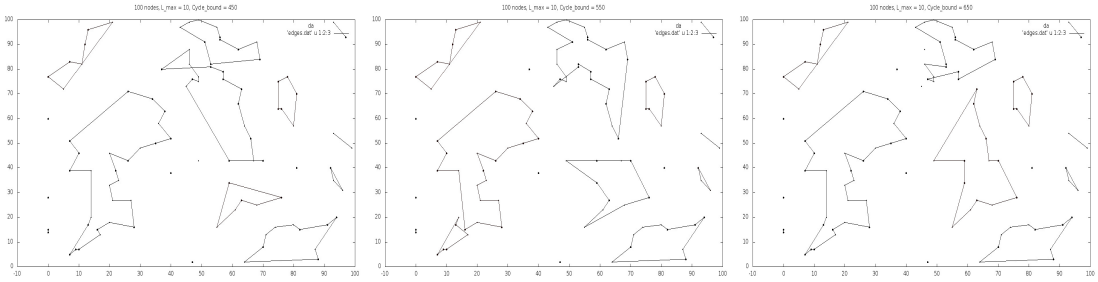


FIGURE 7 – The stabilization of the cycle coverage, as B increases (450, 550 and 650). Scenario for 100 nodes and L_{max} fixed to 10.

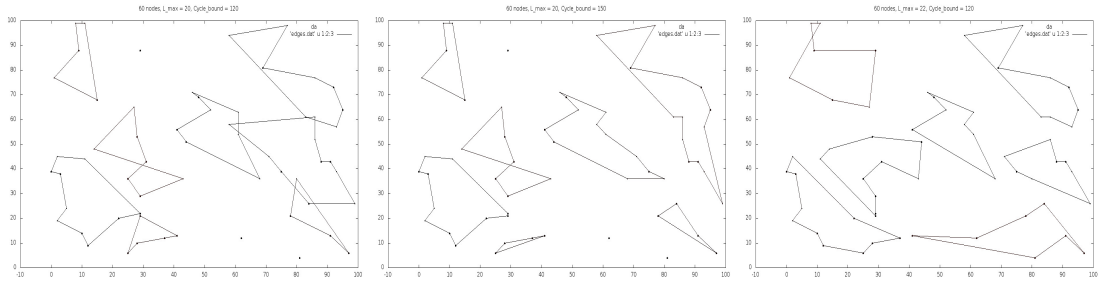


FIGURE 8 – The improvement of the cycle coverage, as B and L_{max} are modified together ($120 - 20$, $150 - 20$ and $120 - 22$). Scenario for 100 nodes.

L_{max} *parameter variation*

In the graphics of Figure 6, one can observe how the solution changes with respect to the modification of L_{max}. Throughout the experiments, we observed that when L_{max} is inside a specific interval, cycles do not overlap. In the random graph model it is difficult to say which interval is the best. When L_{max} increases above a specific limit, the cycles start to overlap. The complementary situation is true : when the parameter decreases below a specific value, the cycles start to be smaller or to disappear. The motivation behind this aspect lies in the N-EN heuristic, where, if L_{max} increases, the nodes neighborhood gets larger and the number of neighbors increases.

B *parameter variation*

In another scenario, we fix the L_{max}. Then, if only the B parameter modifies, the resulted cycles will get larger and larger on the graph, but at some time they will stabilize (see Figure 7). That is because the L_{max} parameter fixes the set of neighbors for a node, and a set of the partition cannot get larger, even if the bound B allows this. We can say that the solution stabilizes, as no big differences occur from that point on.

B *and* L_{max} *variation*

Another thing observed during the experiments with N-EN heuristic is that the variation of B within a threshold, can lead to better coverage if several values for L_{max} are tried. So, by adjusting the two parameters together, B and L_{max}, one may find a set of cycles that do not overlap and that cover all the sensors (i.e. containing no isolated sensors ; see Figure 8). However, how to properly choose the parameters remains an open question.

One problem that still persists with the N-EN heuristic is that sometimes a cycle of the coverage is consistently bigger than all the other ones. From the experimental observations, this exception occurs only in some graphs, and it may happen because of the way we form the set of nodes inside the heuristic. In some cases, a post-covering optimization could be done, to redefine the shape of cycles whose edges overlap. In this sense, other TSP heuristics could be tried as well.

7 CONCLUSION

In the conducted study, we started from the general problem of multi-agent patrolling under the constraints of wireless sensor networks, problem which has not been studied before in the context that we do. The novelty that the problem setting brings is the consideration of networking aspects and constraints in the strategies used for multi-agent patrolling.

This paper addressed the problem of multi-agent patrolling in WSNs by defining and formalizing the problem of vertex Covering with Bounded Simple Cycles (CBSC). We proved it is NP-hard, and consequently considered polynomial-time algorithms to offer solutions for CBSC. Our algorithms, adapted HAC and the proposed N-EN heuristic, were evaluated with respect to networking metrics, robot limitations, and different graph models. Our results show that HAC performs better when it comes of cycles that respect the networking idleness ($WI(G)$ or $AvgIdl(G)$), but N-EN tends to form fewer cycles in each coverage. The

ratio between these metrics finally shows that overall the heuristics have almost equal performance, with a slight advantage for N-EN (due to the number of cycles it delivers).

Annex A

Proof (Theorem 2). We show that TSP can be reduced in polynomial time to CBSC. Let $G = (V, E)$ be the graph for which we want to solve the TSP, where E contains the $\frac{n\times(n-1)}{2}$ edge lengths in finite precision. For simplicity, we round the values to integers (i.e. multiplication by $10^{|s|}$, where $|s|$ is the numeric scale of the lengths). First, let H be the length of a Hamiltonian cycle on G (obtained with Christofides heuristic or any other algorithm). This gives us an upper bound on the Opt_{TSP}. Then, we set the interval for the binary search : $[Left, Right] \leftarrow [0, H]$ (or $[\frac{2}{3} \times H, H]$ suffices in the case of Christofides heuristic). We start the search by setting the bound for CBSC to the middle of the interval : $B \leftarrow \lfloor \frac{Left+Right}{2} \rfloor$. We check the solution of the CBSC, to see if the minimum number of cycles is 1 or greater than 1. We set $Right \leftarrow B$ in the first case, or $Left \leftarrow B$ in the second. The search is performed until $L = R$, which will represent the actual Opt_{TSP}. The running time of Christofides heuristic is $O(|V|^3)$, while the binary search is performed in $O(\log H)$. Hence, the time of the reduction is polynomial in the size of the input. □

Acknowledgments

This work was partly supported by Region Rhône-Alpes (France).

Références

BASAGNI S., CAROSI A., MELACHRINOUDIS E., PETRIOLI C. & WANG Z. M. (2008). Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wirel. Netw.*, **14**(6), 831–858.

BEKKAI S., FORGE D. & KOUIDER M. (2009). Covering the vertices of a graph with cycles of bounded length. *Discrete Mathematics*, **309**(8), 1963 – 1966.

CHEVALEYRE Y. (2004). Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, p. 302–308.

CHEVALEYRE Y. (2005). *The patrolling problem*. Technical report, Univ. Paris 9. Available in french at `http://www.lamsade.dauphine.fr/~chevaley/papers/anna_patro.pdf`.

CHRISTOFIDES N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem*. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University.

CRITES R. & BARTO A. (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, p. 1017–1023 : MIT Press.

DANIEL K., DUSZA B., LEWANDOWSKI A. & WIETFELD C. (2009). Airshield : A system-of-systems muav remote sensing architecture for disaster response. In *Systems Conference, 2009 3rd Annual IEEE*, p. 196–200.

ELMALIACH Y., AGMON N. & KAMINKA G. A. (2009). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, **57**(3-4), 293–320.

GLAD A., SIMONIN O., BUFFET O. & CHARPILLET F. (2008). Theoretical study of ant-based algorithms for multi-agent patrolling. In *Proceedings of the 2008 Conference on ECAI 2008 : 18th European Conference on Artificial Intelligence*, p. 626–630.

JAIN A. K., MURTY M. N. & FLYNN P. J. (1999). Data clustering : A review. *ACM Comput. Surv.*, **31**(3), 264–323.

KARYPIS G. & KUMAR V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, **20**(1), 359–392.

MACHADO A., RAMALHO G., ZUCKER J.-D. & DROGOUL A. (2003). Multi-agent patrolling : An empirical analysis of alternative architectures. In *Proceedings of the 3rd International Conference on Multi-agent-based Simulation II*, MABS'02, p. 155–170.

MARTA M. & CARDEI M. (2009). Improved sensor network lifetime with multiple mobile sinks. *Pervasive and Mobile Computing*, **5**(5), 542 – 555.

PAPADIMITRIOU C. H. (1977). The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, **4**(3), 237 – 244.

PORTUGAL D. & ROCHA R. (2010). Msp algorithm : Multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, p. 1271–1276.

PORTUGAL D. & ROCHA R. (2011). A survey on multi-robot patrolling algorithms. In L. CAMARINHA-MATOS, Ed., *Technological Innovation for Sustainability*, volume 349 of *IFIP Advances in Information and Communication Technology*, p. 139–146. Springer Berlin Heidelberg.

RESCUE R. (2001). home page : `http://www.robocuprescue.org`.

TARJAN R. (1971). Depth-first search and linear graph algorithms. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*, p. 114–121.

TUNCA C., ISIK S., DONMEZ M. & ERSOY C. (2014). Distributed mobile sink routing for wireless sensor networks : A survey. *Communications Surveys Tutorials, IEEE*, **16**(2), 877–897.

Contrôle de Vol d'un Planeur Basé sur une Logique Non-monotone

José-Luis Vilchis Medina[1] Pierre Siegel[1] Andrei Doncescu[2]

[1 2] Aix Marseille Université, CNRS, LIF, Marseille, France
[2] LAAS, CNRS, Toulouse, France

joseluis.vilchismedina@lif.univ-mrs.fr

Résumé

Le pilotage d'un avion consiste à faire divers choix d'actions pour réaliser une trajectoire. Mais les différentes phases de vol ont un environnement changeant, en conséquence on peut avoir des informations incertaines. Les situations peuvent changer en fonction des situations externes, telles que les perturbations atmosphériques, ou situations de sécurité et urgence du pilote. Le problème d'un pilote est alors contrôler l'aéronef dans des situations incertaines. Donc la commande de vol est un problème non-monotone. Nous présentons une méthode pour contrôler un planeur prenant en compte des facteurs d'incertitude et de contradiction.

Mots Clef

Logique non monotone, Logique des défauts, Représentation des connaissances, Planification, Moteur-planeur, Règles de pilotage.

Abstract

Aircraft piloting consist of making various choices of actions to realize a trajectory. But different phases of flight have a changing environment, therefore, We can have uncertain information. Situations may change depending on external conditions, such as atmospheric disturbances or safety and emergency situations of the pilot. The problem of a pilot is then to control the aircraft in uncertain situations. So flight control is a non-monotonic problem. We present a method for controlling a glider taking into account factors of uncertainty and contradiction.

Keywords

Non monotonic logic, Default logic, Knowledge representation, Planning, Motor-Glider, Rules of piloting.

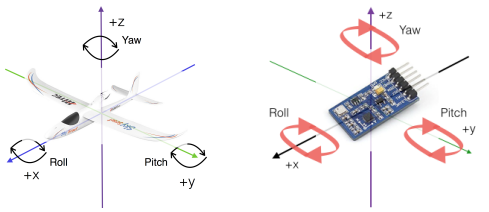
1 Introduction

La stabilisation des aéronefs, à différentes étapes du vol, est une partie importante pour une trajectoire bien définie. Il y a quatre forces qui maintiennent l'avion dans l'air : La force de portance, de traînée, le poids et la traction [2]. Une fois que l'aéronef est dans l'air, il doit s'orienter et se déplacer dans l'espace pour réaliser la trajectoire désirée. Généralment, les angles «roll», «pitch» et «yaw» sont utilisées pour mesurer l'orientation et l'inclinaison. Les avions ont besoin des moteurs comme agents propulseurs, cela permet d'avoir la vitesse nécessaire pour décoller et rester dans l'air [2]. Dans la plupart des cas, le contrôle de l'aéronef sont modélisés par des équations différentielles et des matrices de rotation [8]. Nous abordons le problème du point de vue de l'intelligence artificielle. Pour aborder cette problématique nous utilisons une théorie de la logique non-monotones. Nous utilisons un motoplaneur (planeur disposant d'un moteur d'appoint). L'avantage d'utiliser ce type d'aéronef, c'est parce que, il est l'un des meilleurs en termes de sa finesse (rapport entre la portance et la traînée). La finesse est liée aussi au rapport entre à la distance au sol et l'altitude perdue. Par exemple, s'il s'est déplacé de 40 km avec une perte d'altitude de 1 km, alors sa finesse est de 40. Dans les règles de pilotage, il y a des cas contradictoires ou incertains, si un pilote a une urgence (soit un problème technique, soit pour des raisons de sécurité, etc.), il doit violer certaines règles pour résoudre la problématique. Dans cet article nous allons présenter une méthode basée sur une logique non monotone pour la conduite d'un vol stable, prenant en compte l'incertitude et l'imprécision de l'information[1][7]. Nous présentons dans la section suivante la représentation des connaissances et la logique classique. La définition et les caractéristiques de la logique non monotone sont comprises dans la section 3. La représentation du système en logique des défauts est dans la section 4. Dans la section 5, nous décrivons une simulation d'une situation de vol avec toutes les solutions possibles et finalement les conclusions dans la section 6.

2 Représentation des Connaissances

La représentation des connaissances est une partie importante en intelligence artificielle car elle donne une description de l'environnement qui sera interpréter par un ordinateur. Dans ce contexte, grâce au capteur IMU [1] (Fig. 1), nous disposons de la mesure de différentes variables physiques (vitesses et accélérations, Fig. 2), avec lesquelles nous pouvons représenter les conditions spatio-temporelles du planeur. Ces informations vont permettre d'inférer les

1. Inertial Measurement Unit.



(a) Motoplaneur utilisé (b) Capteur IMU avec 10 degrés de liberté

FIGURE 1 – Axes de reférence du planeur(a) et du capteur(b), les dessins n'ont pas à l'échelle.

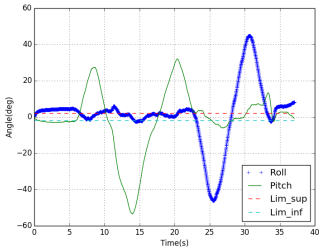


FIGURE 2 – Représentation dynamique du vol.

actions pour atteindre un objectif : décollage, atterrissage, tourner à droite ou à gauche, etc.

2.1 En logique classique

Avec la logique classique nous pouvons décrire des situations qui se passent dans la vie réelle ou encore représenter le monde. Dans la vie réelle, il y a des situations ou règles contradictoires. Si on prend l'exemple suivant, on peut représenter en logique classique le constat : "Lles ovipares pondent des œufs", avec la règle : $layEggs(X) \rightarrow ovipare(X)$. Maintenant, si nous considérons $X = Ornithorynque$, par définition, l'ornithorynque est un mammifère, or il pond des oeufs. En conséquence, la règle précédente ne peut pas être satisfaite. La logique classique est donc limitée, car elle ne prend en compte ni les aspects d'imprécisions ni ceux d'exceptions. Afin de traiter ce type de problématique, nous proposons d'utiliser la logique non monotone, plus particulièrement, la logique des défauts.

3 Logique Non-monotone

J. Mc-Carthy [5], D. Mc-Dermott et J. Doyle, et R. Reiter [6] ont été les premiers auteurs à travailler avec la logique non monotone dès la fin dès années 1970. Une logique monotone ne peut pas gérer les incertitudes (comme vu l'exemple présenté dans la section 2.1). La logique non-monotone est une logique formelle qui ne respecte pas la propriété de la monotonie, définie comme : $A \vdash w$, $A \bigcup B \vdash w$. Si on a un modèle A, et on ajoute des informations de B, on ne peut pas réduire les conclusions de w. Ainsi, elle permet de capturer et représenter le raisonnement par défauts (faits incertains et contradictoires). C'est un problème classique en Intelligence Artificielle. Une des logiques les plus connue et étudiée pour traiter cette problématique est la logique des défauts. C'est cette logique que nous présenterons dans la suite de ces travaux.

3.1 Logique des défauts

Une théorie des défauts T consiste en un ensemble de faits W, qui sont des formules en logique de premier ordre et un ensemble de défauts D, qui sont des règles d'inférence [6]. L'outil de représentation principal est la règle de défaut. Un défaut est une règle d'inférence sous la forme : $\frac{A(X) : B(X)}{C(X)}$, où $A(X), B(X), C(X)$ sont des formules bien formées (en logique de premier ordre). Où $X = (x_1, x_2, x_3, \ldots, x_n)$ est un vecteur de variables libres (non quantifiées). $A(X)$ est le prérequis, $B(X)$ est la justification et $C(X)$ est la conséquence. Il est possible d'utiliser la notation suivante, $A(X) : B(X) \rightarrow C(X)$. Intuitivement, une règle de défaut signifie : "Si $A(X)$ est vrai, et il n'y a aucune preuve que $B(X)$ soit faux, alors $C(X)$ est vrai ". Il est défini comme un défaut normal, si $B(X) = C(X)$. Les règles des défaut sont utilisées pour calculer des extensions.

3.2 Définition d'extension

Une extension de la théorie des défauts $\Delta = (D, W)$, est un ensemble E des formules logiques [6]. Une extension doit vérifier la propriété suivante : Si d est un défaut de D, dont le prérequis est dans E, sans que la négation de sa justification ne soit dans E, alors la conséquence de d est dans E. Formellement, E est une extension de Δ si et seulement si :

— $E = \cup_{i=0}^{\infty} E_i$ avec :
— $E_0 = W$ et pour $i \geq 0$

$$E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : B(X)}{C(X)} \in D, A(X) \in E_i \text{ et } \neg B(X) \notin E\}$$

Où $Th(E_i)$ est l'ensemble de formules dérivés de E_i. La définition précédente est difficile à appliquer dans la pratique. Parce que $\neg B(X) \notin E$ suppose E est connu, mais E n'est pas encore calculé. Nous allons prendre la définition des défauts normaux, c'est-à-dire, $B(X) = C(X)$. Une extension est définie : E est une extension de Δ si et seulement si :

— $E = \cup_{i=0}^{\infty} E_i$ avec :
— $E_0 = W$ et pour $i \geq 0$

$$E_{i+1} = Th(E_i) \cup \{C(X) | \frac{A(X) : C(X)}{C(X)} \in D, A(X) \in E_i \text{ et } \neg C(X) \notin E_i\}$$

Où $Th(E_i)$ est l'ensemble de formules dérivés de E_i. Selon Reiter [6], si tous les défauts sont normaux, il existe au moins une extension. Les extensions sont définies comme points fixes ou solutions.

4 Représentation en Logique des Défauts

Dans cette section nous présentons les ensembles de faits W et de défauts D. Nous donnons également un exemple de calcul des extensions.

4.1 L'ensemble des faits W

W est un ensemble de faits vrais. Par exemple, $glider(decrease, t)$, signifie que le planeur a perdu de l'altitude au temps t par rapport au temps $t-1$. Dans cet ensemble, nous pouvons aussi décrire des exclusions mutuelles, ce sont des situations qui ne peuvent pas se produire en même temps. Par exemple pour les actions, le planeur ne peut pas tourner à gauche et à droite simultanément au même temps t. Formellement cette exclusion mutuelle est représentée de la façon suivante : $\forall t, \neg(yoke(pull, t+1) \wedge yoke(push, t+1))$.

4.2 L'ensemble des défauts D

L'ensemble des règles d'inférence D décrit des actions possibles en prenant en compte les informations incomplètes et contradictoires de l'environnement. Chaque règle est représenté sous la forme de défaut normal (Section 3.1). Par exemple, $\frac{glider(decrease,t) : yoke(pull,t+1)}{yoke(pull,t+1)}$. Intuitivement, cela signifie, "Si le $glider(decrease, t)$ est vrai, et si c'est possible que $yoke(pull, t+1)$, alors $yoke(pull, t+1)$". Autrement dit, si c'est possible de faire une action, on la ferra.

4.3 Exemple de calcul

Voici, nous avons une fonction F qui représente les faits vrais. Cette représentation d'écrit la situation du planeur au temps t (pour la fonction $glider(X, t)$, elle sera notée comme $g(X, t)$).

$$F : g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t).$$

Nous pouvons dire en français que : le planeur perd de l'altitude, il tourne à gauche, il a le moteur coupé et une vitesse basse au moment t. Nous nous posons la question suivante : Quelles actions devons-nous faire pour avoir un vol stable ? Avec cette information, nous pouvons calculer les extensions (ensemble des actions) possibles pour atteindre l'objectif : vol stable. Ensuite, nous montrons les extensions avec les différents défauts obtenus (Fig. 3) :

$$E_0 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_6 = \frac{g(decrease, t) : yoke(pull, t+1)}{yoke(pull, t+1)} \quad (1)$$

$$d_8 = \frac{g(decrease, t) : motor(on, t+1)}{motor(on, t+1)} \quad (2)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (3)$$

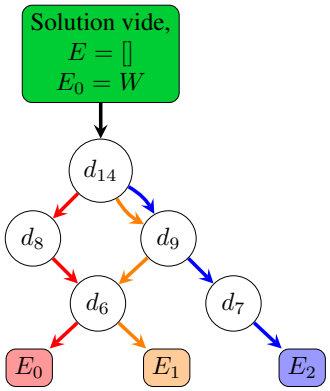


FIGURE 3 – Représentation graphique de calcul des extensions.

$$E_1 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_6 = \frac{g(decrease, t) : yoke(pull, t+1)}{yoke(pull, t+1)} \quad (4)$$

$$d_9 = \frac{g(decrease, t) : motor(off, t+1)}{motor(off, t+1)} \quad (5)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (6)$$

$$E_2 = \{g(decrease, t), g(turn\_left, t), g(motor\_off, t), g(low\_altitude, t), g(low\_airspeed, t)\}$$

$$d_7 = \frac{g(decrease, t) : yoke(push, t+1)}{yoke(push, t+1)} \quad (7)$$

$$d_9 = \frac{g(decrease, t) : motor(off, t+1)}{motor(off, t+1)} \quad (8)$$

$$d_{14} = \frac{g(turn\_left, t) : yoke(right, t+1)}{yoke(right, t+1)} \quad (9)$$

Nous pouvons constater que dans chaque extension calculée, nous avons des sous-systèmes cohérents [3][4]. Maintenant, nous avons résolu le problème de contradiction, présenté au début (Section 1). Nous pouvons ajouter des règles plus complexes pour une base de connaissances plus complète.

5 Simulation : Décollage

Dans cette section nous simulons une situation de décollage et nous calculons toutes les solutions possibles pour cette phase de pilotage. Les principaux éléments qui conduisent à l'objectif souhaité sont :

- État : C'est la représentation de la situation de l'objet (planeur) à chaque instant du temps.
- Temps : C'est l'instant entre les états.
- Action : C'est l'activité à réaliser afin d'atteindre l'objectif.

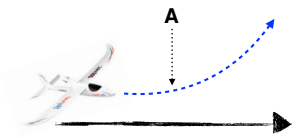


FIGURE 4 – Situation A.

Ces éléments sont clés pour inférer les actions suivantes. Dans le calcul sont considérés les trois aspects précédents. Dans cette situation (Fig. 4) nous avons les informations suivantes (pour la fonction $glider(X,t)$ vue précédemment, elle sera notée comme $g(X,t)$) :

$$G : g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t).$$

Si nous appliquons la théorie des défauts, plus particulièrement la logique des défauts (Section 3.1), nous obtenons 5 différentes extensions ou solutions.

$$E_0 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (10)$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (11)$$

$$d_{20} = \frac{g(low\_altitude,t) : motor(on,t+1)}{motor(on,t+1)} \quad (12)$$

$$E_1 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (13)$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (14)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (15)$$

$$E_2 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{16} = \frac{g(roll\_stable,t) : yoke\_roll(neutral,t+1)}{yoke\_roll(neutral,t+1)} \quad (16)$$

$$d_{19} = \frac{g(low\_altitude,t) : yoke(push,t+1)}{yoke(push,t+1)} \quad (17)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (18)$$

$$E_3 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

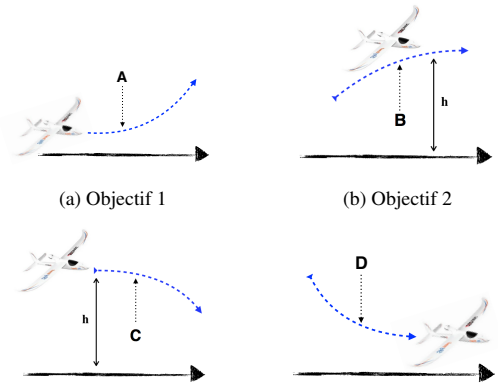
$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (19)$$

$$d_{18} = \frac{g(low\_altitude,t) : yoke(pull,t+1)}{yoke(pull,t+1)} \quad (20)$$

$$d_{20} = \frac{g(low\_altitude,t) : motor(on,t+1)}{motor(on,t+1)} \quad (21)$$

$$E_4 = \{g(pitch\_stable,t), g(roll\_stable,t), g(motor\_off,t), g(low\_altitude,t), g(low\_airspeed,t)\}$$

$$d_{17} = \frac{g(pitch\_stable,t) : yoke\_pitch(neutral,t+1)}{yoke\_pitch(neutral,t+1)} \quad (22)$$



(a) Objectif 1 (b) Objectif 2

(c) Objectif 3 (d) Objectif 4

FIGURE 5 – Différents objectifs pendant le vol.

$$d_{18} = \frac{g(low\_altitude,t) : yoke(pull,t+1)}{yoke(pull,t+1)} \quad (23)$$

$$d_{21} = \frac{g(low\_altitude,t) : motor(off,t+1)}{motor(off,t+1)} \quad (24)$$

La meilleure extension et/ou solution de la situation A simulée précédemment (Fig. 4) est l'extension E_3 qui a les combinaisons des actions $\{yoke(pull,t+1), yoke\_pitch(neutral,t+1), motor(on,t+1)\}$ nécessaires pour atteindre l'objectif : **décollage**. Puisque l'extension E_0 a : $\{yoke\_roll(neutral,t+1), yoke\_pitch(neutral,t+1), motor(on,t+1)\}$, le résultat est un vol droit que ce n'est pas l'objectif. L'extension E_1 a : $\{yoke\_roll(neutral,t+1), yoke\_pitch(neutral,t+1), motor(off,t+1)\}$, il n'y a pas de mouvement sur le planeur car le moteur est coupé. L'extension E_2 a : $\{yoke\_roll(neutral,t+1), yoke(push,t+1), motor(off,t+1)\}$, même résultat que l'extension précédente, pas de mouvement car le moteur est coupé. Et finalement, l'extension E_4 a : $\{yoke(pull,t+1), yoke\_pitch(neutral,t+1), motor(off,t+1)\}$, même résultat que l'extension précédente, pas de mouvement car le moteur est coupé. Le choix d'une extension (ensemble des actions) est basé sur une analyse multicritère [3]. Nous avons choisi le modèle de produits pondérés car il permet de faire une analyse adimensionnelle sur les alternatives. Ensuite, nous montrons la définition :

$$P(A_K/A_L) = \prod_{j=1}^{n}(a_{Kj}/a_{Lj})^{w_j} \text{ pour } K, L = 1, 2, 3, ..., m.$$

Nous montrons sur la Fig. 5 les différents objectifs qui sont les différents étapes de pilotage.

6 Conclusion

La logique non-monotone est un outil important en Intelligence Artificielle car elle permet de capturer le raisonnement incertain. Nous avons fait une simulation d'un cas contradictoire (décollage) et nous avons constaté que la logique des défauts peut gérer les situations d'incertitudes et

des contradictions, de cette façon, nous avons obtenu 5 extensions ou solutions qui sont sous-systèmes cohérentes. Nous sommes en train d'améliorer les aspects des décisions, pour cela nous étudions l'optimum de Pareto. Le théorème de Pareto n'est pas sensible aux déséquilibres dans la répartition des ressources. Ce point est important pour résoudre le problème de gestion de l'énergie du système.

Références

[1] A. Doncescu, P. Siegel, DNA Double-Strand Break-Based Nonmonotonic Logic, *Computational Biology, Bioinformatics and Systems Biology*, 2015.

[2] Direction Général de l'Aviation Civile, *Manuel du pilote d'avion Vol à vue,* Direction Général de l'Aviation Civile, 1992.

[3] I. Toulgoat, Modélisation du comportement humain dans les simulations de combat naval, *Thèse de doctorat*, 2011.

[4] I. Toulgoat, P. Siegel, A. Doncescu, Modelling of Submarine Navigation by Nonmonotonic Logic, *BWCCA*, 2011.

[5] J. McCarthy, Circumscription - A form of non-monotonic reasoning, *Artificial Intelligence*, 1980.

[6] R. Reiter, A logic for Default Reasonig, *Artificial Intelligence*, pp. 81-132, 1980.

[7] T. Le, A. Doncescu, P. Siegel, Utilization of Default Logic for Analyzing a Metabolic System in Discrete Time, *ICCSA*, 2013.

[8] K. Terrell, S. Zein-Sabatto, Intelligent reconfigurable control system for aircraft flight control, *Southeast-Con, 2017, IEEE*, 2017.

La logique facile avec TouIST

Formalisez et résolvez facilement des problèmes avec des solveurs SAT, SMT ou QBF

Olivier Gasquet[1], Andreas Herzig[2], Dominique Longin[2], Frédéric Maris[1], Maël Valais[1]
IRIT (Institut de Recherche en Informatique de Toulouse)
[2]CNRS, [1]Université Paul Sabatier, Toulouse, France
{Olivier.Gasquet,Andreas.Herzig,Dominique.Longin,Frederic.Maris,Mael.Valais}@irit.fr

TouIST

(Toulouse Integrated Satisfiability Tool)

Un compilateur de langages logiques étendus et de haut niveau vers des prouveurs efficaces et indépendants

```
0_a_perdu <=>
bigor $t in $T when $t>0:
  tour_de_0($t) and reste($t,0)
          and not reste($t-1,1)
end
```

Problème en langage TouIST

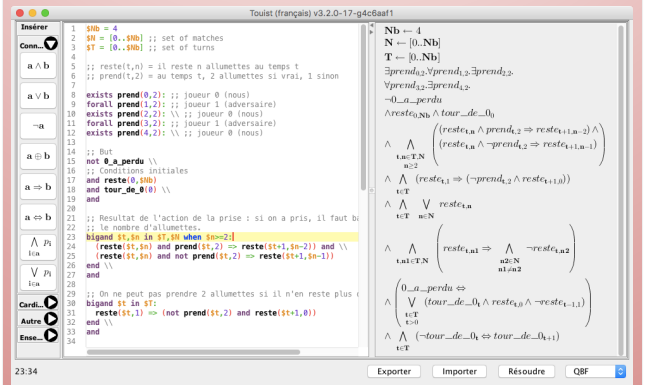
Programme TouIST

```
reste(5,0) -> vrai
0_a_perdu  -> faux
```

Solution

Spécificités

- Langage riche
- Édition visuelle
- Multiples solveurs (SAT, SMT, QBF)
- Facilement extensible
- Projet github, licence MIT



Exemple : le jeu de Nim

- Chacun son tour ($t \in T$), un joueur prend 1 ou 2 allumettes ($n \in A$)
- Celui qui ne peut plus prendre d'allumettes a perdu
- Nous jouons le rôle du **joueur 0** et l'adversaire est le **joueur 1**

Étape 1. Définir les règles du jeu

(Formalisation des règles en logique)

Jouer. Selon son choix, le joueur laisse **une** ou **deux** allumettes en moins :

$$\bigwedge_{\substack{t\in T\\ n\in A\\ n\geq 2}} \begin{pmatrix}(reste(t,n) \wedge \neg prend\_2(t) \rightarrow reste(t+1, n-\mathbf{1}))\\ \wedge\ (reste(t,n) \wedge prend\_2(t) \rightarrow reste(t+1, n-\mathbf{2}))\end{pmatrix}$$

Dernier coup. S'il ne reste plus qu'**une** allumette, le joueur n'a pas le choix, il doit la prendre et ne laisse **aucune** allumette :

$$\bigwedge_{t\in T} (reste(t,\mathbf{1}) \rightarrow \neg prend\_2(t) \wedge reste(t+1,\mathbf{0}))$$

Perdre. Le **joueur 0** a perdu ssi à un tour t, c'est à lui de jouer et il ne reste **aucune** allumette :

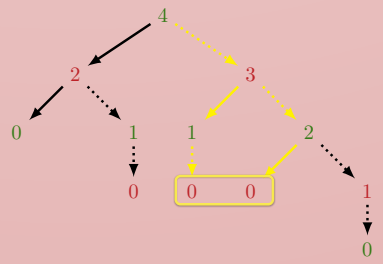
$$\mathbf{0}\_perd \leftrightarrow \bigvee_{\substack{t\in T\\ t>0}} (tour\_de\_\mathbf{0}(t) \wedge reste(t,\mathbf{0}))$$

Étape 2. Jouer et gagner à tous les coups !

(Formalisation d'une stratégie gagnante à l'aide de QBF)

Quels coups le **joueur 0** doit-il faire pour **être sûr de gagner** quels que soient les coups du **joueur 1** ?

$$\begin{bmatrix}\exists prend\_2(0).\\ \forall prend\_2(1).\\ \exists prend\_2(2).\\ \forall prend\_2(3).\\ \boxed{\neg \mathbf{0}\_perd} \wedge \boldsymbol{\Phi}\end{bmatrix}$$



où $\boldsymbol{\Phi}$ représente les règles du jeu présentées en étape 1

www.irit.fr/touist