Soutenance de thèse

Résolution efficace de processus décisionnels de Markov par l'exploitation d'approches structurelles et algorithmiques tirant parti de l'architecture moderne des ordinateurs

Jaël Champagne Gareau

Doctorat en informatique
Université du Québec à Montréal
Directeurs: Éric Beaudry et Vladimir Makarenkov

6 décembre 2024

Plan

Plan

Planification automatique probabiliste

- La **planification automatique** est une branche de l'intelligence artificielle.
- Elle vise à trouver des plans permettant d'atteindre un objectif.
- Certains problèmes de planification sont **probabilistes** :
  - Incertitude endogène (actuateurs/senseurs de l'agent) ;
  - Incertitude exogène (environnement).
- Les **processus décisionnels de Markov** (MDP) permettent de modéliser certains de ces problèmes.

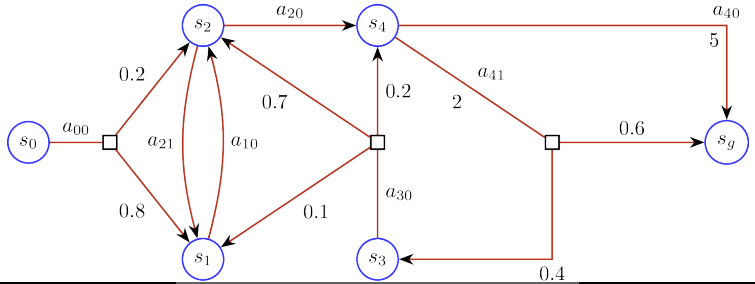
Exemples de domaines où les MDP sont utilisés

- Finance
- Neuroscience
- Jeux de hasard
- Marketing
- Théorie du contrôle
- Déplacement de robots
- Etc.

Processus décisionnel de Markov à plus court chemin stochastique (SSP-MDP)

Un **SSP-MDP** est un tuple (S, A, T, C, G) où :

- S est l'ensemble fini des états ;
- A est l'ensemble fini des actions que l'agent peut exécuter ;
- $T\colon S \times A \times S \to [0, 1]$ est la fonction de transition :
  - $T(s, a, s')$ donne la probabilité que l'agent atteigne l'état s' s'il exécute l'action a à l'état s ;
- $C\colon S \times A \times S \to \mathbb{R}^+$ est la fonction de coût :
  - $C(s, a, s')$ donne le coût qu'un agent doit payer s'il atteint l'état s' en exécutant l'action a à l'état s ;
- $G \subseteq S$ est l'ensemble des états buts.



Algorithmes existants

Objectif

Trouver une **politique** $\pi : S \to A$ qui minimise l'espérance du coût total pour atteindre un but.

Algorithmes classiques

- Value Iteration (VI), 1957 [1]
- Policy Iteration (PI), 1960 [2]

Approches par priorisation

- Generalized Prioritized Sweeping (genPS), 1998 [3]
- Partitioned, Prioritized, Parallel Value Iteration (P3VI), 2005 [4]

1. Bellman, R. (1957). Dynamic Programming. Prentice Hall.
2. Howard, R. A. (1960). Dynamic Programming and Markov Processes. John Wiley.
3. Andre, D. et al. (1998). Generalized prioritized sweeping. Proceedings of the 10th International Conference on Neural Information Processing Systems (p. 1001-1007). MIT Press.
4. Wingate, D. and Seppi, K. D. (2005). Prioritization methods for accelerating MDP solvers. Journal of Machine Learning Research, 6, 851-881.

Algorithmes existants

Objectif

Trouver une **politique** $\pi : S \to A$ qui minimise l'espérance du coût total pour atteindre un but.

Approches heuristiques

- Improved Looped And/Or* (ILAO*), 2001 [1]
- Labeled Real-Time Dynamic Programming (LRTDP), 2003 [2]

Approches topologiques

- Topological Value Iteration (TVI), 2011 [3]

1. Hansen, E. A. and Zilberstein, S. (2001). LAO* : A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 129(1-2), 35-62.
2. Bonet, B. and Geffner, H. (2003). Improving the Convergence of Real-Time Dynamic Programming. Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003) (vol. 3, p. 12-21).
3. Dai, P. et al. (2011). Topological value iteration algorithms. Journal of Artificial Intelligence Research, 42, 181-209.

Sujet de recherche

- Plusieurs problèmes réels nécessitent un grand nombre de variables d'état.
- Malédiction de la dimensionnalité : le nombre d'états est exponentiel en le nombre de variables d'état.
- Dans plusieurs circonstances, le temps alloué pour trouver une solution d'un MDP est limité.
- Les approches existantes ne suffisent pas pour résoudre certains MDP dans un temps raisonnable.

Idée

- Prise en compte d'éléments tels que la hiérarchie de mémoire et les différents niveaux de parallélisme lors de la conception et l'implémentation des algorithmes.
- Ces approches ont a plusieurs reprise (entre autre pour les tris, les matrices et les graphes) permises d'obtenir des amélioration de plusieurs ordre de grandeur.
- En ML : utilisation de matériel (GPU) et de types de données (bfloat) spécialisés.

Objectif

Proposer des approches structurelles et algorithmiques exploitant l'architecture moderne des ordinateurs pour résoudre des MDP de grande taille de manière plus efficace.

Plan

Implémentations existantes

- La différence de performance entre les algorithmes de résolution de MDP (VI, TVI, LRTDP, etc.) peut parfois être plus petite que l'accélération pouvant être obtenue par une implémentation plus efficace.
- Les structures utilisées pour stocker un MDP en mémoire sont rarement mentionnées dans la littérature.
- Survol de quelques implémentations disponibles en ligne publiquement :
  - AI Toolbox [4]
    - Utilise des matrices denses ou creuses pour stocker les MDPs (une pour les transitions, et une pour les coûts) ;
    - Matrices denses : mémoire gaspillée ;
    - Matrices creuses : utilisation sous-optimale de la mémoire cache.
  - L'implémentation des auteurs de TVI :
    - Liste chaînée d'états ;
    - Chaque état contient une liste chaînée d'actions applicables ;
    - Très mauvaise efficacité en cache et surcoût en mémoire.
  - MDP Engine Library [5] et G-Pack [6] (C++ Libraries)
    - Implémentation des auteurs de LRTDP et de Gourmand (2e place à la compétition ICAPS2014) ;
    - Table de hachage contenant des structures d'états ;
    - Deux niveaux d'indirection, ce qui empêche une utilisation optimale du cache.

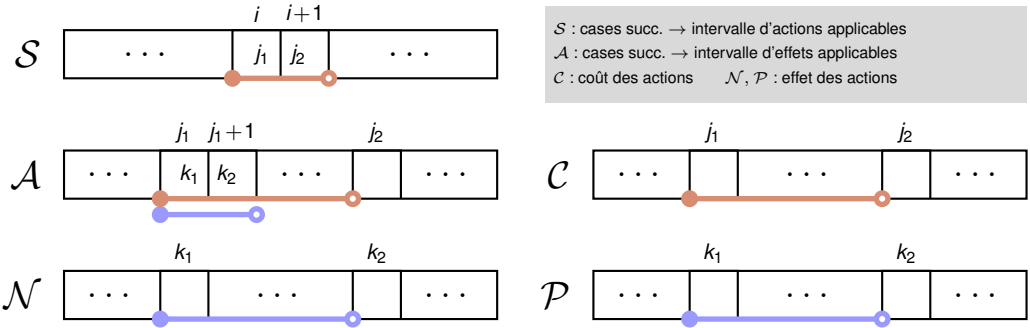
4. E. Bargiacchi, D. M. Roijers, and A. Nowé. AI-Toolbox : A C++ library for Reinforcement Learning and Planning (with Python Bindings). Journal of Machine Learning Research (2020), pp. 1-12.

5. `https://github.com/bonetblai/mdp-engine`

6. A. Kolobov, Mausam, and D. S. Weld, "LRTDP versus UCT for online probabilistic planning," in Proc. of the Natl. Conf. on AI, 2012, vol. 3, no. 1, pp. 1786-1792.

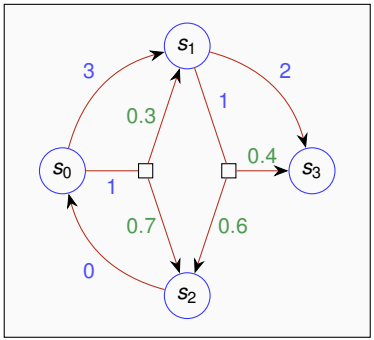
Représentation CSR-MDP

- CSR-MDP est inspirée par la représentation **Compressed Sparse Row** des graphes.
- Représentation des MDP “structure de tableaux” (SoA) plutôt que “tableau de structures” (AoS).
- Minimum de mémoire gaspillée : pas de pointeurs ni de padding nécessaire.
- Compact en mémoire : la plupart du contenu des lignes de cache chargées est utile au calcul en cours.
- Facilite l’utilisation d’opérations SIMD.



$\mathcal{S}$: cases succ. $\rightarrow$ intervalle d’actions applicables
$\mathcal{A}$: cases succ. $\rightarrow$ intervalle d’effets applicables
$\mathcal{C}$: coût des actions $\quad \mathcal{N}, \mathcal{P}$: effet des actions

Exemple d'une instance de MDP sous forme CSR-MDP



$\mathcal{S}$

0	1	2	3	4
0	2	4	5	5

$\mathcal{A}$

0	1	2	3	4	5
0	1	3	4	6	7

$\mathcal{C}$

0	1	2	3	4
3	1	2	1	0

$\mathcal{N}$

0	1	2	3	4	5	6
1	1	2	3	3	2	0

$\mathcal{P}$

0	1	2	3	4	5	6
$\frac{10}{10}$	$\frac{3}{10}$	$\frac{7}{10}$	$\frac{10}{10}$	$\frac{4}{10}$	$\frac{6}{10}$	$\frac{10}{10}$

Méthodologie expérimentale

- La performance de CSR-MDP est comparée à celle de la représentation utilisée dans l'implémentation originale de TVI sur :
  - VI (variante asynchrone *Round-Robin*) ;
  - LRTDP (avec l'heuristique admissible et indépendante du domaine h_{min}) ;
  - TVI.
- Chaque algorithme est exécuté jusqu'à convergence à $\epsilon = 10^{-6}$.
- Les algorithmes sont implémentés en C++ et compilés avec GNU g++ 11.2 (option `-O3`).
- Les tests ont été effectués sur un PC équipé d'un processeur Intel Core i5 7600k.
- La mémoire n'était pas un enjeu : aucun test n'a utilisé plus de 2 Go.
- Pour chaque configuration de test, nous avons généré aléatoirement 15 instances.
- Pour minimiser les facteurs aléatoires, on mesure les médianes des 15 résultats obtenus.

Description des domaines de planification testés

Domaine par couches (*Layered*)

- Introduit dans l'article original de TVI.
- Domaine générique qui modélise les situations où certains évènements sont irréversibles.
- Ex. : jeux de plateau où le nombre de pièces d'un joueur ne peut jamais augmenter.

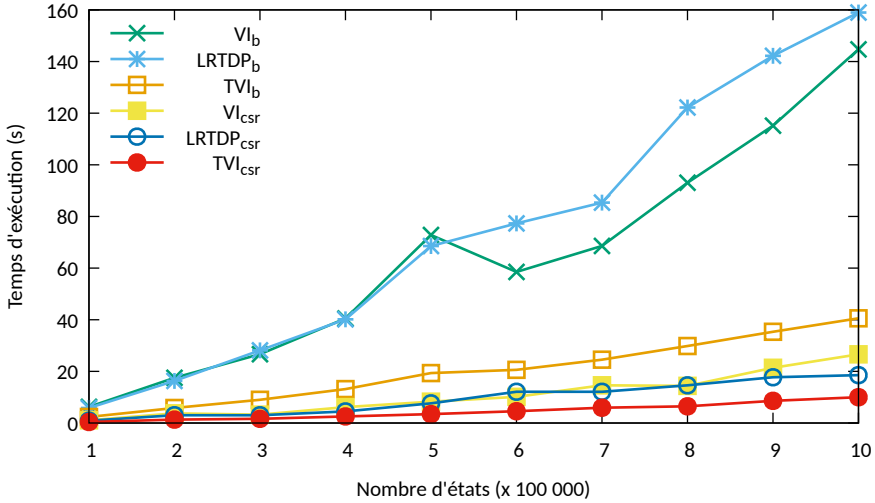
Domaine du pendule à un bras (SAP)

- Problème de contrôle optimal bidimensionnel à temps minimal.
- Deux actions possibles à chaque état : appliquer un couple positif ou négatif.
- Objectif : placer le balancier du pendule en équilibre à la verticale sur son axe.

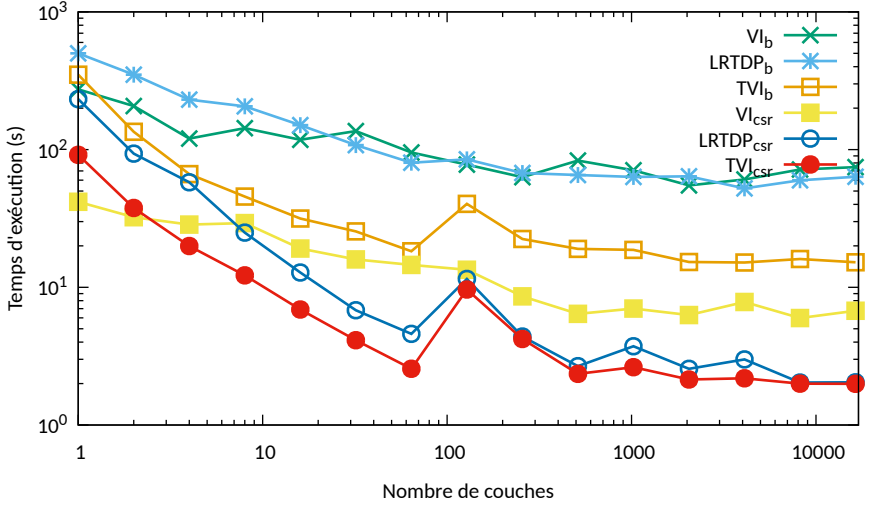
Domaine du plancher humide (*Wetfloor*)

- Plusieurs salles (grille de navigation carrée) sont connectées les unes aux autres.
- Chaque cellule d'une salle peut être sèche, légèrement humide, ou détrempée
- Objectif : trouver le chemin le plus court pour naviguer entre deux positions.

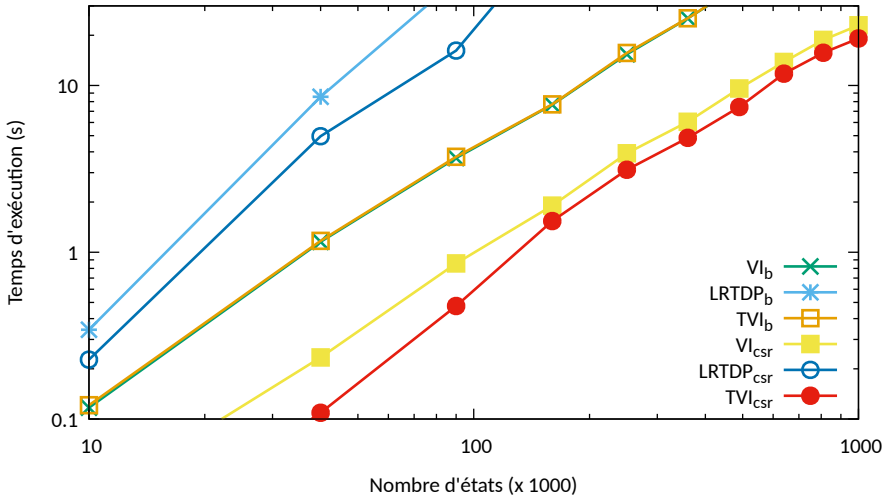
Domaine par couches (*Layered*) en faisant varier le nombre d'états (10 couches)



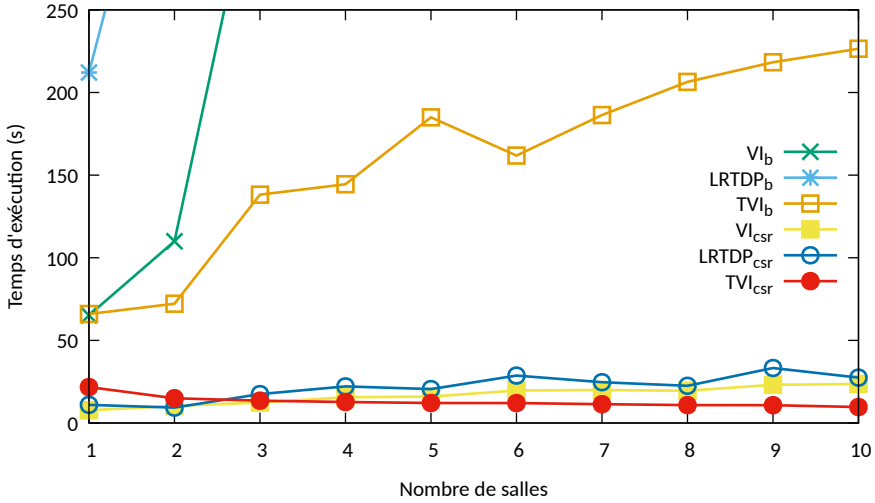
Domaine par couches (*Layered*) en faisant varier le nombre de couches (1M d'états)



Domaine du pendule à un bras (*SAP*)



Domaine du plancher humide (500k états)



Résultats

Domaine	**VI**	**LRTDP**	**TVI**
Layered (en faisant varier le nombre d'états)	5.87	7.92	4.47
Layered (en faisant varier le nombre de couches)	6.77	$>$ 4.08	$>$ 3.88
SAP	4.36	$>$ 1.05	5.34
Wetfloor	$>$ 15.33	$>$ 13.81	12.36
Moyenne	$>$ 8.70	$>$ 2.86	$>$ 6.65

Table – Facteurs d'accélération moyens obtenus par chaque solveur sur chaque domaine en utilisant la représentation mémoire proposée, CSR-MDP, lorsque comparée à l'implémentation de base. Les nombres avec un symbole "$>$" sont des bornes inférieures du vrai facteur d'accélération.

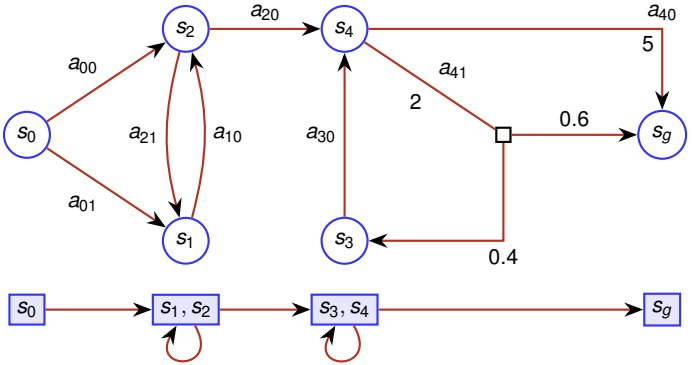
Plan

Algorithmes parallèles existants

- **P3VI** : *Partitionned, Prioritized, Parallel Value Iteration*
  - Partitionne l'espace d'états en plusieurs sous-parties ;
  - Assignation d'une priorité à chaque partie ;
  - Résolution de plusieurs parties en parallèle dans l'ordre donné par les priorités.
  - Désavantages :
    - Partitionnement fait au cas par cas selon le domaine de planification ;
    - Communication des valeurs d'états entre les fils d'exécution résolvant les parties : surcoût en temps de calcul.
- **Parallel CECA** : *Cache-Efficient with Clustering and Annealing*
  - CECA partitionne l'espace d'états et résout les parties une par une selon un ordre déterminé par un algorithme de recuit simulé.
  - Parallel CECA résout plusieurs parties en parallèle.
  - Désavantages :
    - L'algorithme final est plus complexe à comprendre/implementer que la plupart des algorithmes de MDP ;
    - Les gains de performance sont assez faibles (facteur 2.59 sur un CPU à 10 cœurs).

Algorithme TVI : *Topological Value Iteration*

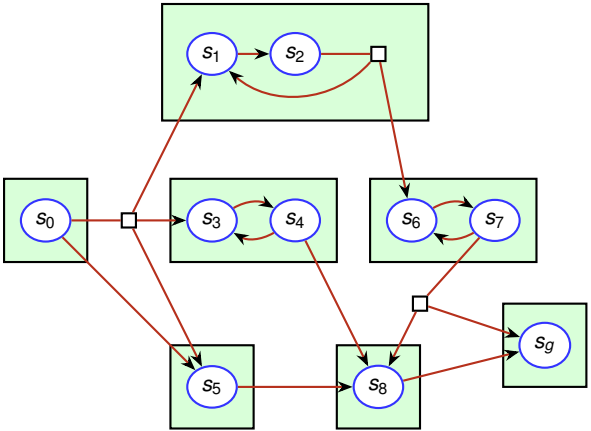
- Considère le graphe correspondant à la structure topologique du MDP ;
- Utilise l'algorithme de Tarjan pour décomposer le graphe en composantes fortement connexes (CFC) ;
- Si on considère les CFC dans l'ordre topologique inverse, on peut résoudre le MDP en un seul balayage.



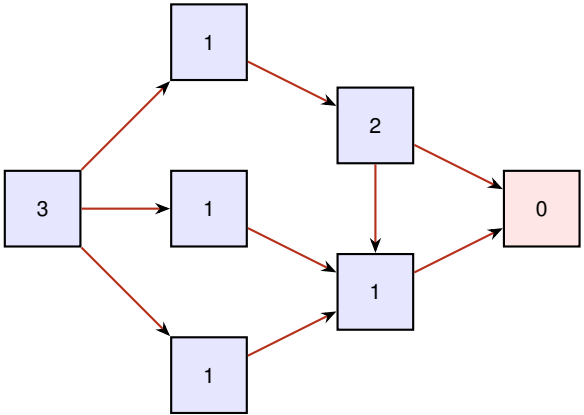
Algorithme pcTVI : *Parallel-Chained Topological Value Iteration*

- **pcTVI** est basé sur TVI (*Topological Value Iteration*). ;
- Plutôt que de choisir les CFC à résoudre en parallèle de manière aléatoire ou avec une métrique de priorité, il utilise plutôt les dépendances entre les CFC.
- Les CFC sans dépendances communes peuvent être résolues en parallèle.
- Pour trouver les dépendances : on peut faire un parcours en largeur inversée (du but) dans le graphe des CFC du MDP pour trouver des chaînes de CFC indépendantes.
- Durant l'exécution, une nouvelle tâche parallèle est créée à chaque fois que les dépendances d'une CFC ont toutes été calculées.

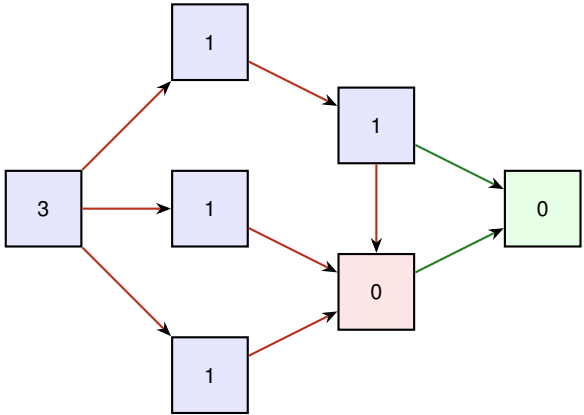
Exemple d'exécution de pcTVI



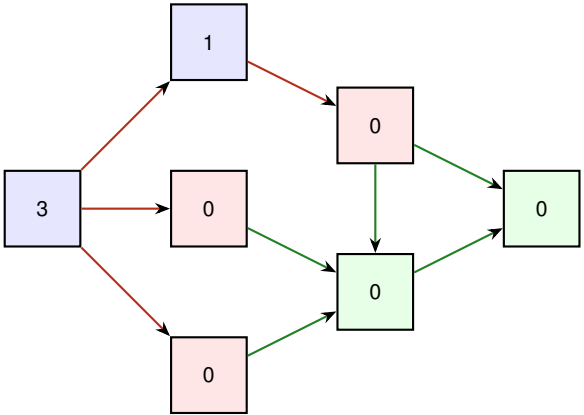
Exemple d'exécution de pcTVI



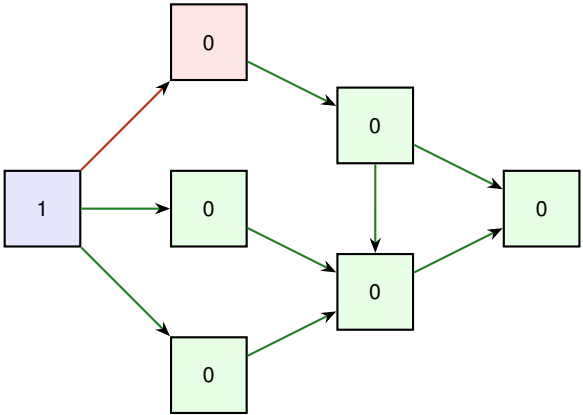
Exemple d'exécution de pcTVI



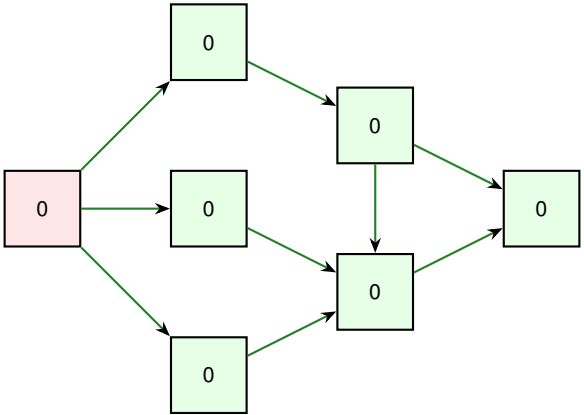
Exemple d'exécution de pcTVI



Exemple d'exécution de pcTVI



Exemple d'exécution de pcTVI



Domaine chaîné

- Aucun domaine standard de planification probabiliste dans la littérature n'a été conçu pour évaluer un solveur parallèle de MDP.
- Nouveau domaine paramétrique proposée : le **domaine chaîné**.
- Les paramètres sont :
  - k : le nombre de chaînes indépendantes $c_1, c_2, \ldots, c_k$;
  - n_C : le nombre de CFC $C_{i,1}, Ci, 2, \ldots, Ci, n_C$ dans chaque chaîne c_i ;
  - n_{cpc} : le nombre d'états par CFC ;
  - n_a : le nombre d'actions applicables par état ;
  - n_e : le nombre d'effets probabilistes par action.
- Les successeurs d'un état s dans $C_{i,j}$ peuvent être n'importe quel état dans $C_{i,j}$ ou dans $C_{i+1,j}$ (ou, si ce dernier n'existe pas, l'état but).

Exemple d'une instance du domaine *chaîné*

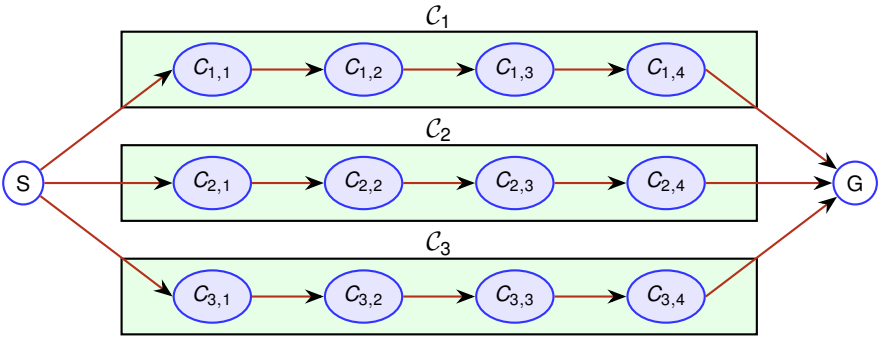
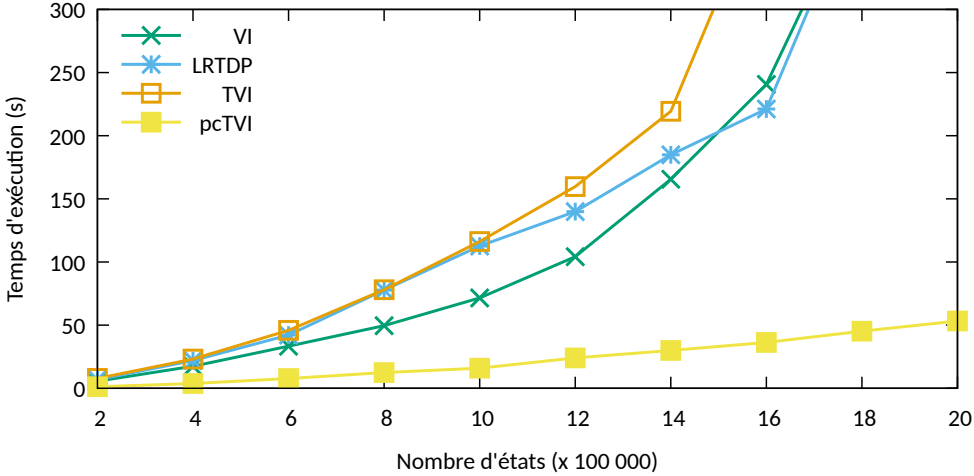


Figure – Instance du domaine *chaîné* où $n_c = 3$ et $n_C = 4$. Chaque ellipse représente une CFC.

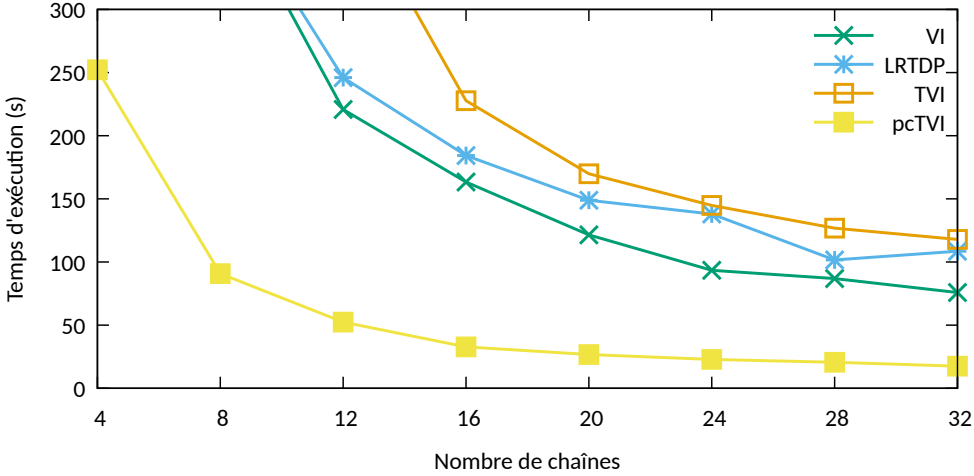
Méthodologie expérimentale

- La performance de pcTVI est comparée à celle de :
  - VI (variante asynchrone *Round-Robin*) ;
  - LRTDP (avec l'heuristique admissible et indépendante du domaine h_{min}) ;
  - TVI.
- Chaque algorithme est exécuté jusqu'à convergence à $\epsilon = 10^{-6}$.
- Les algorithmes sont implémentés en C++ et compilés avec GNU g++ 11.2 (option `-O3`).
- Les tests ont été effectués sur un PC équipé de **4 processeurs Intel Xeon E5-2620V4**.
- Chacun de ces processeurs a 8 cœurs (à 2.1 GHz), pour un total de 32 cœurs.
- La mémoire n'était pas un enjeu : aucun test n'a utilisé plus de 2 Go.
- Pour chaque configuration de test, nous avons généré aléatoirement 15 instances.
- Pour minimiser les facteurs aléatoires, on mesure les médianes des 15 résultats obtenus.

Domaine *Chained-MDP* avec nombre variable d'états et 32 chaînes



Domaine *Chained-MDP* avec 1M d'états et un nombre variable de chaînes



Plan

Considération de la hiérarchie de mémoire

Problème ouvert

« We believe that more research on cache efficiency of MDP algorithms is desirable and could lead to substantial payoffs — similar to the literature in the algorithms community, one may gain huge speedups due to better cache performance of the algorithms. » — Mausam et Kolobov (2012) [7]

Algorithme CEC : *Cache-efficient with clustering*

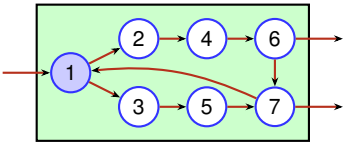
- L'algorithme **CEC** [8] est basé sur l'algorithme **FTVI** (*Focused Topological Value Iteration*).
- Il divise l'espace d'états en partis à l'aide d'un algorithme de partitionnement :
  - prendre un état aléatoire s dans la CFC courante ;
  - faire une recherche largeur depuis s ;
  - arrête lorsque la taille de la CFC dépasse un seuil (p. ex., taille du cache L3).
- CEC résout les partis de chaque CFC cycliquement en faisant des balayages de Bellman sur chaque partie jusqu'à convergence de la CFC.

7. Mausam et Kolobov, A. (2012). Planning with Markov Decision Processes : An AI Perspective. Synthesis Lectures on Artificial Intelligence and Machine Learning (vol. 6). Morgan & Claypool. https://doi.org/10.2200/S00426ED1V01Y201206AIM017

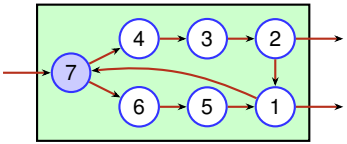
8. Jain, A. et Sahni, S. (2020). Cache efficient Value Iteration using clustering and annealing. Computer Communications, 159, 186-197. https://doi.org/10.1016/j.comcom.2020.04.058

Analyse de la performance de TVI

- TVI peut améliorer la performance par rapport à VI de trois façons :
  1. il maximise l'efficacité de chaque mise à jour de Bellman en s'assurant que seules les valeurs d'état qui ont convergé sont propagées d'une CFC à l'autre ;
  2. l'ordre de balayage est donné par un parcours en profondeur postordre ce qui améliore le flux d'information par rapport à l'ordre arbitraire utilisé par VI ;
  3. il y a moins d'états à la fois par balayage qui sont considérés, et donc il y a moins de défauts de cache.



VI : ordre arbitraire



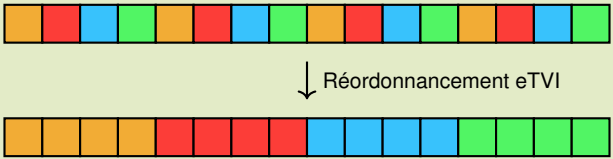
TVI : DFS postordre

eTVI : Réordonnancement des états en fonction de leur ordre externe aux CFC

- Une façon d'améliorer la performance du cache est de réordonner le MDP en mémoire de sorte que les données relatives à chaque CFC soient contiguës.
- **eTVI** utilise cette idée et réordonne les états en fonction de la composante à laquelle ils appartiennent.

Exemple de eTVI

- On suppose chaque état prend 16 octets, et chaque CFC contient 4 états.
- Avant : chaque CFC est répartie sur quatre lignes de cache.
- Après : chaque CFC est contenue dans une seule ligne de cache.

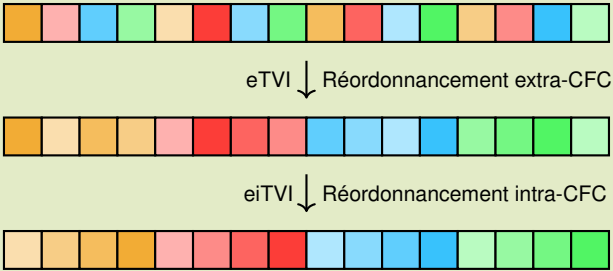


eiTVI : Réordonnancement des états en fonction de leur ordre interne aux CFC

- eTVI ne considère que l'ordre des états par rapport à la CFC à laquelle ils appartiennent.
- Que dire de l'ordre des états **à l'intérieur** d'une CFC ?
- **eiTVI** réordonne les états en fonction de l'ordre considéré par les balayages internes des CFC.

Exemple de eiTVI

- La teinte d'une couleur représente l'ordre de considération des états à l'intérieur d'une CFC lors d'un balayage (les teintes plus claires sont considérées avant les teintes plus foncées).

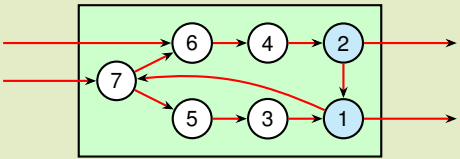


Quel est le meilleur ordre de balayage des états dans une CFC ?

- TVI utilise l'ordre de découverte des états lors de la recherche des CFC (parcours profondeur postordre).
- Idéalement, il faudrait utiliser l'ordre qui maximise la propagation des valeurs d'état.
- Une solution possible : trouver un ordre dynamique de balayage des états à l'intérieur d'une CFC de manière similaire à PVI :
  - nécessite une file de priorité, et a un surcoût important.
- Autre solution : ordre statique de balayage donné par un parcours en largeur inversé partant des états frontières vers l'extérieur de la CFC.

Exemple de l'ordre proposée

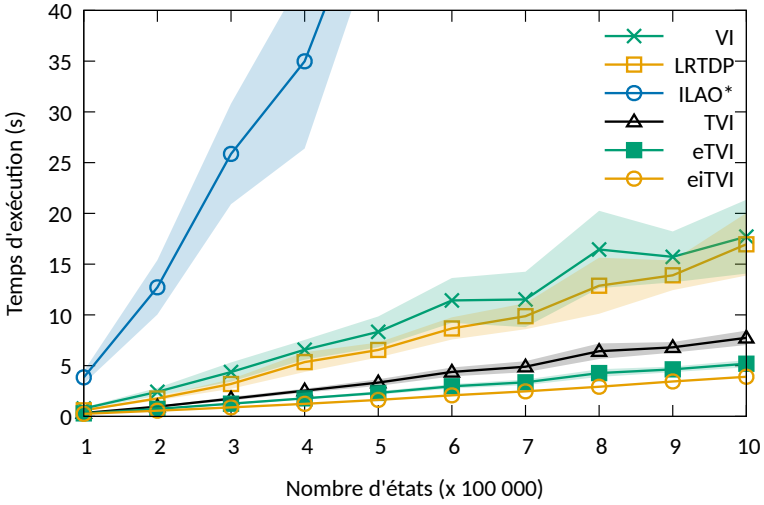
- Les états en bleu sont les états frontières de la CFC.
- Les états sont numérotés selon leur ordre de balayage.



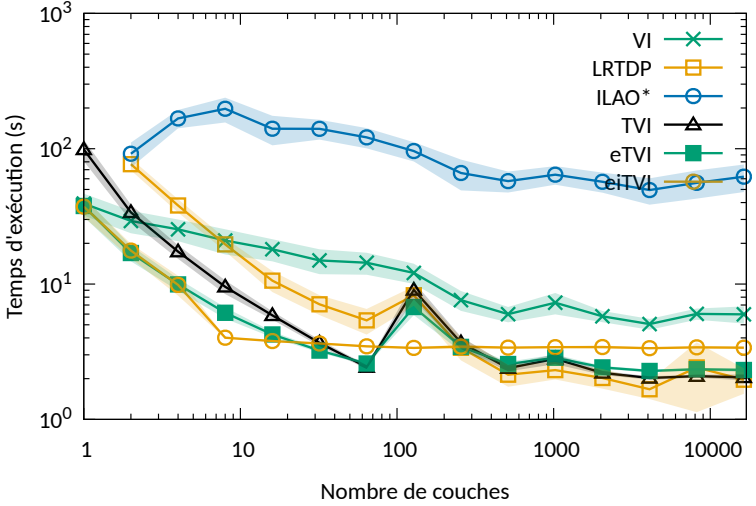
Méthodologie expérimentale

- La performance de eTVI et de eiTVI est comparée à celle de :
  - VI (variante asynchrone *Round-Robin*) ;
  - LRTDP (avec l'heuristique admissible et indépendante du domaine h_{min}) ;
  - **ILAO*** ;
  - TVI.
- Chaque algorithme est exécuté jusqu'à convergence à $\epsilon = 10^{-6}$.
- Les algorithmes sont implémentés en C++ et compilés avec GNU g++ 11.2 (option `-O3`).
- Les tests ont été effectués sur un PC équipé d'un processeur Intel Core i5 7600k.
- La mémoire n'était pas un enjeu : aucun test n'a utilisé plus de 2 Go.
- Pour chaque configuration de test, nous avons généré aléatoirement 15 instances.
- Pour minimiser les facteurs aléatoires, on mesure les médianes des 15 résultats obtenus.
- Domaines de planification testés :
  - Domaine par couches (*Layered*) ;
  - Domaine du pendule à un bras (*SAP*) ;
  - Domaine du plancher humide (*Wetfloor*).

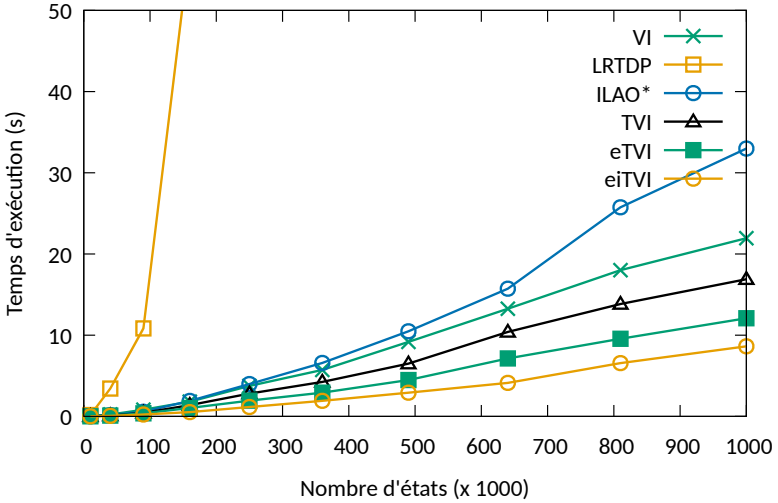
Domaine par couches en faisant varier le nombre d'états (10 couches)



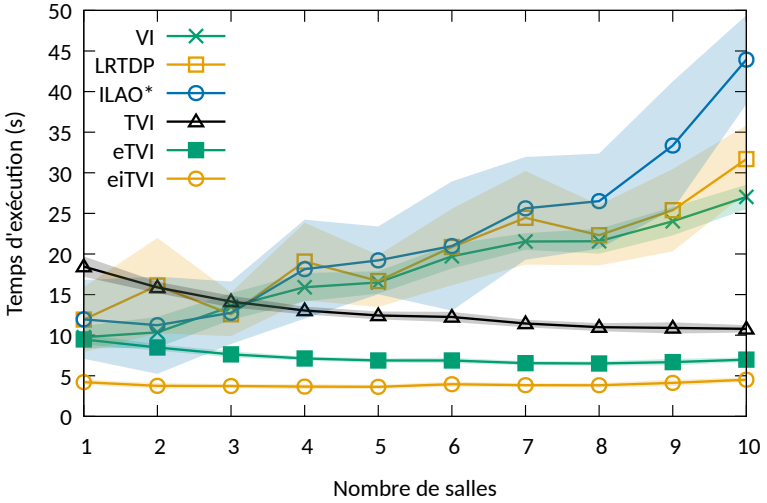
Domaine par couches en faisant varier le nombre de couches (1M d'états)



Domaine du pendule à un bras (SAP)



Domaine du plancher humide (500k états)



Résultats

Domaine	TVI / VI	eTVI / TVI	eiTVI / eTVI	eiTVI / TVI
Layered (en faisant varier le nombre d'états)	2.50	1.43	1.40	2.00
Layered (en faisant varier le nombre de couches)	1.81	1.45	0.98	1.42
SAP	1.40	1.37	1.74	2.39
Wetfloor	1.38	1.78	1.86	3.31
Moyenne	1.63	1.60	1.31	2.10

Table – Facteurs d'accélération moyens obtenus entre VI, TVI, eTVI et eiTVI.

Algorithme	Accès au cache	Défauts de cache	Ratio de défauts
TVI	2.87G	0.86G	29.96 %
eTVI	2.39G	0.41G	17.28 %
eiTVI	1.59G	0.33G	20.62 %

Table – Métriques de la mémoire cache sur TVI, eTVI et eiTVI sur le domaine par couches (instance avec 1M d'états, 10 couches).

Plan

Étant donné un domaine de planification, quel algorithme est le plus rapide ?

- Pour certains domaines, nous connaissons déjà la réponse :
  - MDP denses (les actions peuvent mener à un grand nombre d'états) : algorithmes classiques (ex. : VI et PI) ;
  - MDP ayant un grand nombre d'états buts : approches heuristiques (ex. : LRTDP et ILAO*) ;
  - MDP ayant un grand nombre de composantes fortement connexes : approches topologiques (ex. : TVI)
- Qu'en est-il des domaines ayant une combinaison de ces caractéristiques ?

Problème ouvert

« More theory is needed to guide the development and selection of such enhancements. The most useful would be problem features and optimality definitions that would indicate which metric, reordering method and partitioning scheme are maximally effective, and which would guide the development of new enhancements. These may include distributional properties of the reward functions, distributional properties of transition matrices, strongly/weakly connected component analyses, etc. » — Wingate et Seppi (2005)[9]

9. Wingate, D. and Seppi, K. D. (2005). Prioritization methods for accelerating MDP solvers. Journal of Machine Learning Research, 6, 851-881.

Manque de domaines standardisés dans la littérature

- Pour avoir une meilleure idée à priori de l'algorithme le plus approprié à chaque situation, il faudrait disposer d'un plus grand nombre de domaines de planification standardisés.
- Les domaines ce rapprochant le plus de ce besoin sont ceux utilisés dans la compétition internationale de planification probabiliste (IPPC) ayant lieu à la conférence ICAPS.
  - Toutefois, leur nombre est relativement faible.
  - Les domaines décrivent principalement des problèmes à horizon fini ou infini, plutôt que des problèmes de plus court chemin stochastique.
  - Ils manquent de diversité dans les caractéristiques topologiques impactant les performances des algorithmes.

Objectif

- Identifier les caractéristiques des MDP qui influencent la performance des algorithmes.
- Générer un grand nombre de MDP synthétiques avec différentes caractéristiques.
  - Ils pourraient servir de données d'entraînement pour entraîner un classificateur.
  - Ils pourraient servir de bancs d'essais pour évaluer de nouveaux algorithmes.

Caractéristiques proposées

- Le **nombre d'états** $|S|$.
- Le **nombre d'actions** $|A|$.
- Le **nombre d'états buts** $|G|$.

Caractéristiques proposées

- Le **nombre d'états** $|\mathcal{S}|$.
- Le **nombre d'actions** $|A|$.
- Le **nombre d'états buts** $|G|$.

- Le **nombre de composantes fortement connexes (CFC)** $|\mathfrak{S}|$.
- Le **nombre d'états dans la plus grande CFC** $\max_{\mathcal{S} \in \mathfrak{S}} |\mathcal{S}|$.

Caractéristiques proposées

- Le **nombre d'états** $|S|$.
- Le **nombre d'actions** $|A|$.
- Le **nombre d'états buts** $|G|$.

- Le **nombre de composantes fortement connexes (CFC)** $|\mathfrak{S}|$.
- Le **nombre d'états dans la plus grande CFC** $\max_{\mathcal{S}\in\mathfrak{S}} |\mathcal{S}|$.

- Le **coefficient de clustering** : $\mathfrak{C} := \frac{1}{|S|} \sum_{s\in S} \frac{e_s}{k_s(k_s-1)}$, où e_s est le nombre de paires d'états directement atteignables à partir de s qui sont également directement atteignables l'un de l'autre, et k_s est le nombre d'états directement atteignables à partir de s. De plus, $\mathfrak{C}$ est fixé à 0 lorsque $k_s < 2$.
- La **distribution des actions** : $\forall k, P_k^a :=$ proportion des états qui ont k actions applicables.

Caractéristiques proposées

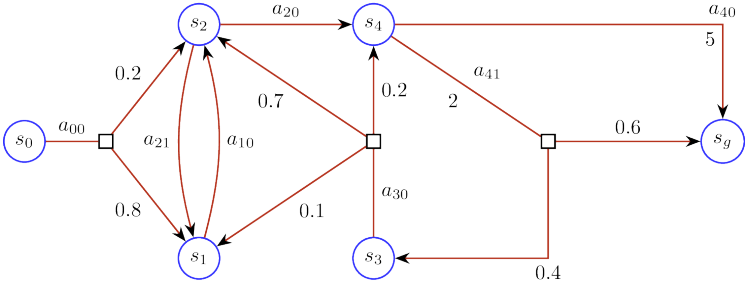
- Le **nombre d'états** $|S|$.
- Le **nombre d'actions** $|A|$.
- Le **nombre d'états buts** $|G|$.

- Le **nombre de composantes fortement connexes (CFC)** $|\mathfrak{S}|$.
- Le **nombre d'états dans la plus grande CFC** $\max_{\mathcal{S}\in\mathfrak{S}} |\mathcal{S}|$.

- Le **coefficient de clustering** : $\mathfrak{C} := \frac{1}{|S|} \sum_{s\in S} \frac{e_s}{k_s(k_s-1)}$, où e_s est le nombre de paires d'états directement atteignables à partir de s qui sont également directement atteignables l'un de l'autre, et k_s est le nombre d'états directement atteignables à partir de s. De plus, $\mathfrak{C}$ est fixé à 0 lorsque $k_s < 2$.
- La **distribution des actions** : $\forall k, P_k^a :=$ proportion des états qui ont k actions applicables.

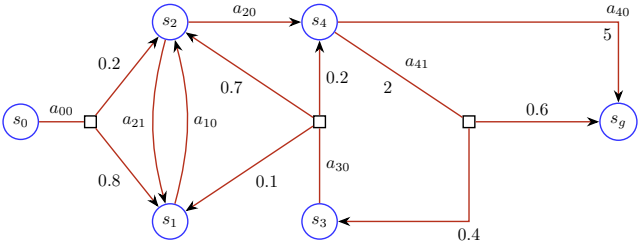
- La **distribution des transitions** : $\forall k, P_k^t :=$ proportion des actions qui ont k transitions probabilistes.
- L'**excentricité des états buts** : $\mathcal{G} := \min_{g\in G} \max_{s\in S} \bar{d}(s,g)$, où $\bar{d}(s,g)$ est le nombre minimum d'actions (le coût de chaque action n'est pas considéré) qui doivent être exécutées pour atteindre g à partir de s.

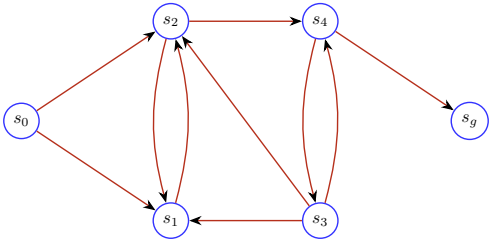
Exemple



- $|S| = 6, |A| = 7, |G| = 1$;
- $\mathfrak{S} = \{\{s_0\}, \{s_1, s_2, s_3, s_4\}, \{s_g\}\}$;
- $\mathbf{p^a} = [\frac{1}{6}, \frac{3}{6}, \frac{2}{6}]$;
- $\mathbf{p^t} = [0, \frac{4}{7}, \frac{2}{7}, \frac{1}{7}]$;
- $\mathfrak{C} = \frac{1}{6}(\frac{2}{2 \cdot 1} + 0 + \frac{0}{2 \cdot 1} + \frac{3}{3 \cdot 2} + 0 + 0) = \frac{1}{4}$;
- $\mathcal{G} = 3$.

Déterminisation d'un MDP





Génération synthétique de graphes

- Un petit nombre de domaines de planifications synthétiques existent :
  - Domaine par couches (contrôle le nombre de CFC) ;
  - Domaine chaîné (contrôle le nombre de chaînes indépendantes de CFC).
- En comparaison, il existe beaucoup plus de méthodes de génération de graphes synthétiques.

Modèle	**Distribution degrés**	**Coefficient de clustering**	**Diamètre**
Erdös-Rényi	binomiale	petit ($\bar{k}/n$)	petit : $\mathcal{O}(\log n)$
Watts-Strogatz	quasi constante	grand	petit
Barabási–Albert	invariant d'échelle ($\bar{k}^{-3}$)	grand ($\bar{k}^{-1}$)	petit : $\mathcal{O}(\frac{\log n}{\log(\log n)})$
Kronecker	multinomiale	flexible	flexible

Table – Méthodes de génération de graphes synthétiques et liste de certaines des propriétés topologiques leur étant associées, où $\bar{k}$ est le degré moyen des sommets dans le graphe, et n est le nombre de sommets.

Génération de MDP synthétiques

Algorithme Génération de MDP synthétiques

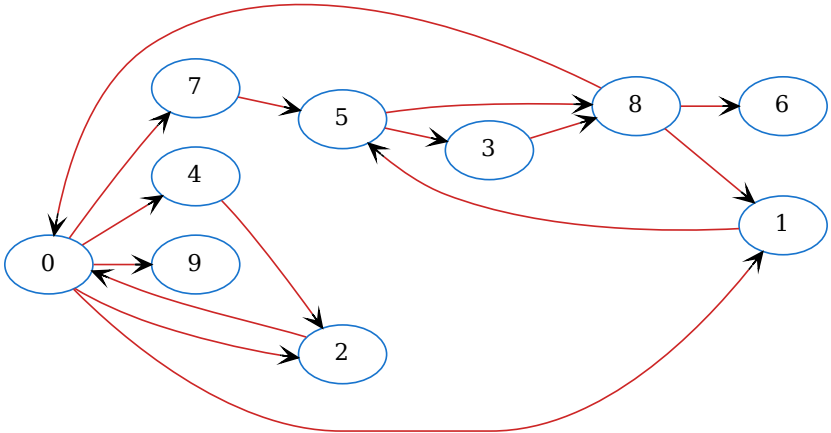
Entrée: Une liste de propriétés topologiques désirées (n : nombre d'états ; k : nombre de buts, etc.)
Sortie: Un PDM-PCCS (S, A, T, C, G)

▷ Utiliser la méthode de génération de graphes la plus appropriée par rapport aux propriétés voulues
$\Gamma \leftarrow$ GénérerGrapheSynthétique(n) ▷ p. ex., une des méthodes de la table 4
$S \leftarrow \Gamma$.obtenirNœuds() ▷ $|S| = n$

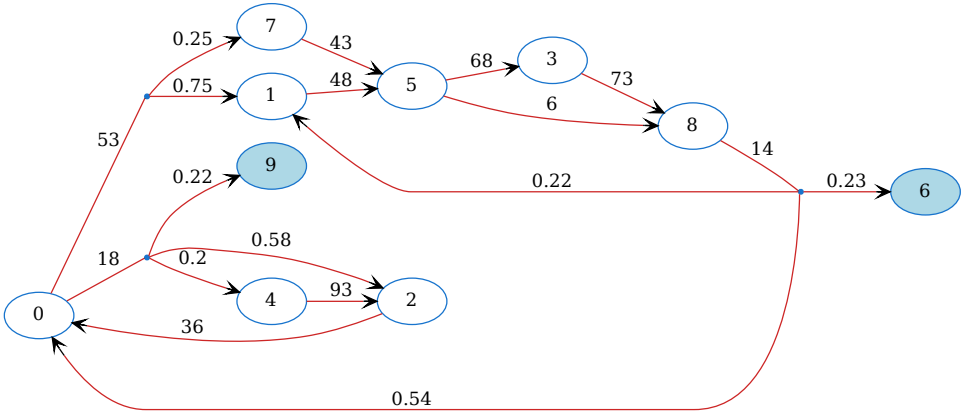
pour tout $s \in S$ **faire**
    $a_s \leftarrow$ EntierAléatoireUniformeEntre(1, k_s) ▷ Générer le nombre d'actions ; k_s est le degré de s
    $A_s \leftarrow$ DécomposerEnSomme(k_s, a_s) ▷ A_s est un tableau de a_s éléments où $\sum_{n_a \in A_s} n_a = k_s$
    pour tout $n_a \in A_s$ **faire** ▷ n_a est le nombre de transitions possibles de l'action courante
        $a \leftarrow$ nouvel identifiant d'action
        $A \leftarrow A \cup \{a\}$
        $C(s, a) \leftarrow$ CoûtAléatoire() ▷ Distribution uniforme ou autre, selon le besoin
        $P_a \leftarrow$ GénérerProbabilités(n_a) ▷ P_a est un tableau où $\sum_{p \in P_a} p = 1.0$ et $|P_a| = n_a$
        pour tout $i \in [1..n_a]$ **faire**
            $s' \leftarrow$ VoisinAléatoire(Γ, s) ▷ Voisin aléatoire de s dans le graphe Γ
            $T(s, a, s') \leftarrow P_a[i]$

$G \leftarrow$ SousEnsembleAléatoire(S, k) ▷ k est un paramètre contrôlant le nombre de buts souhaités
retourner (S, A, T, C, G)

Exemple de graphe synthétique généré avec le modèle de Erdös-Rényi



MDP synthétique généré à partir du graphe précédent



Plan

Résumé des contributions (1)

Contribution 1 : Nouvelle représentation mémoire des MDP

- Nouvelle représentation mémoire des MDP, **CSR-MDP**, basée sur la représentation *Compressed Sparse Row* (CSR) des graphes.
- CSR-MDP a permis un gain de performance de 8.6, 2.8 et 6.6 en moyenne, respectivement pour VI, LRTDP et TVI.
- Publication présentée à la conférence **Canadian AI 2022** et ayant été sélectionnée pour un *Best-paper award*[10].

Canadian AI 2022
30 May–3 June 2022, Toronto

10. Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2022). Cache-Efficient Memory Representation of Markov Decision Processes. Dans Proceedings of the Canadian Conference on Artificial Intelligence. Canadian Artificial Intelligence Association (CAIAC). https://doi.org/10.21428/594757db.0e910d58

Résumé des contributions (2)

Contribution 2 : Exploitation du parallélisme de fils d'exécution

- Nouvel algorithme de résolution de MDP en parallèle, **pcTVI**, indépendant du domaine de planification.
- Nouveau domaine de planification paramétrique, le domaine **chaîné**, adapté à modéliser des situations où différentes stratégies peuvent atteindre un but.
- L'algorithme a permis un gain de performance de 20 en moyenne sur un ordinateur de 32 cœurs.
- Publication présentée à la conférence **IFCS 2022** [11].

11. Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2023). pcTVI : Parallel MDP solver using a decomposition into independent chains. Dans P. Brito, J. G. Dias, B. Lausen, A. Montanari et R. Nugent (dir.), Classification and data science in the digital age (p. 101-109). Springer International Publishing. https://doi.org/10.1007/978-3-031-09034-9_12

Résumé des contributions (3)

Contribution 3 : Exploitation de la hiérarchie de mémoire

- Deux améliorations de TVI exploitant la représentation CSR-MDP et améliorant la performance en cache :
  - **eTVI** : réordonne les états en mémoire pour que les données de chaque SCC soient contiguës ;
  - **eiTVI** : réordonne les états en mémoire pour que l'ordre en mémoire corresponde à l'ordre dans lequel les états sont considérés lors des balayages de Bellman.
- Les algorithmes ont permis un gain de performance de 2.1 en moyenne.
- Publication présentée à la conférence **ECAI 2023** [12].

12. Champagne Gareau, J., Gosset, G., Beaudry, É. et Makarenkov, V. (2023). Cache-Efficient Dynamic Programming MDP Solver. Dans Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023) (vol. 372, p. 373-380). IOS Press. https://doi.org/10.3233/FAIA230293

Résumé des contributions (4)

Contribution 4 : Instances synthétiques de MDP

- Proposition de caractéristiques topologiques d'intérêt pour les MDP.
- Nouvelle méthode de génération de MDP synthétiques couvrant les différentes combinaisons de caractéristiques d'intérêt.
- Publication présentée à la conférence **IFCS 2024** [13].

13. Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2024). Towards topologically diverse probabilistic planning benchmarks : Synthetic domain generation for markov decision processes. Dans J. Trejos, T. Chadjipadelis, A. Grané et V. Mario (dir.), Data science, classification and artificial intelligence for modeling decision making (p. 63-70). Springer International Publishing.

Implémentations

`MDPtk`

- Un projet en C++ a été implémenté, appelé `MDPtk`, comprenant :
  - la représentation CSR-MDP ;
  - plusieurs algorithmes de résolution de MDP de la littérature (VI, PI, TVI, LRTDP, ILAO*) ;
  - les algorithmes proposés (pcTVI, eTVI, eiTVI) ;
  - quelques générateurs d'instances de domaines de planification (par couches, SAP, plancher humide, etc.).
- Le projet est disponible sur GitLab [14].

`graph-toolkit`

- Un projet en C++ a été implémenté, appelé `graph-toolkit`, comprenant :
  - les quatre modèles de graphes synthétiques mentionnés précédemment ;
  - l'algorithme de génération de MDP synthétiques ;
  - différentes fonctions pour trouver les propriétés topologiques des graphes/MDP générés.
- Le projet est disponible sur GitLab [15].

14. `https://gitlab.info.uqam.ca/champagne_gareau.jael/mdptk`
15. `https://gitlab.info.uqam.ca/champagne_gareau.jael/graph-toolkit`

Autres contributions

Planification de chemins couvrants

- Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2021). Fast and Optimal Planner for the Discrete Grid-Based Coverage Path-Planning Problem. Dans Intelligent Data Engineering and Automated Learning – IDEAL 2021 (p. 87-96). Springer International Publishing. https://doi.org/10.1007/978-3-030-91608-4_9
- Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2023). Fast and optimal branch-and-bound planner for the grid-based coverage path planning problem based on an admissible heuristic function. Frontiers in Robotics and AI, 9, 1076897. https://doi.org/10.3389/frobt.2022.1076897

Planification coopérative pour véhicules électriques

- Champagne Gareau, J., Lavoie, M.-A., Gosset, G. et Beaudry, É. (2024). Cooperative Electric Vehicles Planning. Dans Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (p. 290-298). International Foundation for Autonomous Agents and Multiagent Systems. https://www.ifaamas.org/Proceedings/aamas2024/pdfs/p290.pdf
- Champagne Gareau, J., Gosset, G., Lavoie, M.-A., Beaudry, É. et Makarenkov, V. (2024). Increased Plan Stability in Cooperative Electric Vehicles Path-Planning. ICAPS 2024 Workshop on Human-Aware Explainable Planning. https://openreview.net/forum?id=vtWg28K6Lu

Idées de suites et prolongement des travaux

Décomposition plus fine de l'espace d'états

- Idée 1 : Utiliser des algorithmes de regroupement, p. ex., l'algorithme de Louvain, pour subdiviser les CFC.
- Idée 2 : Trouver les ponts orientés et vérifier lesquels peuvent être élagués, créant ainsi plus de CFC.

Généralisation des contributions à des situations plus variés

- MDP où l'espace d'état est trop grand pour être stocké en mémoire ;
- MDP où l'espace d'états n'est pas connu à l'avance (ex. : apprentissage par renforcement)
- MDP où les transitions sont dynamiques ;

Entraînement d'un classificateur pour prédire l'algorithme le plus rapide

- Générer une grande quantité de MDP synthétiques avec différentes caractéristiques.
- Utiliser ces MDP pour entraîner un classificateur qui prédit l'algorithme le plus rapide pour un MDP donné.

Conclusion

Merci pour votre attention !

Des questions ?

Reconnaissances

Nous remercions le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) ainsi que le Fonds de recherche du Québec – Nature et Technologie (FRQNT) pour leur soutien.