Mercredis-recherche

Planification en IA et exploitation des ordinateurs

Jaël Champagne Gareau

Chercheur postdoctoral en informatique
Université TÉLUQ

19 février 2025

Plan

1 Planification d'itinéraires pour véhicules électriques
- Planification individuelle
- Planification coopérative

2 Planification de chemins couvrants

3 Processus décisionnels de Markov
- Introduction aux MDP
- CSR-MDP : Représentation mémoire des MDP
- pcTVI : Parallélisation de TVI
- eTVI et eiTVI : Exploitation de la hiérarchie de mémoire
- MDP synthétiques

Plan

1 Planification d'itinéraires pour véhicules électriques
- Planification individuelle
- Planification coopérative

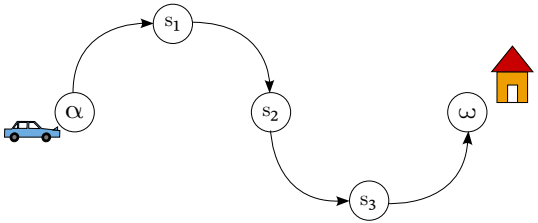
2 Planification de chemins couvrants

3 Processus décisionnels de Markov
- Introduction aux MDP
- CSR-MDP : Représentation mémoire des MDP
- pcTVI : Parallélisation de TVI
- eTVI et eiTVI : Exploitation de la hiérarchie de mémoire
- MDP synthétiques

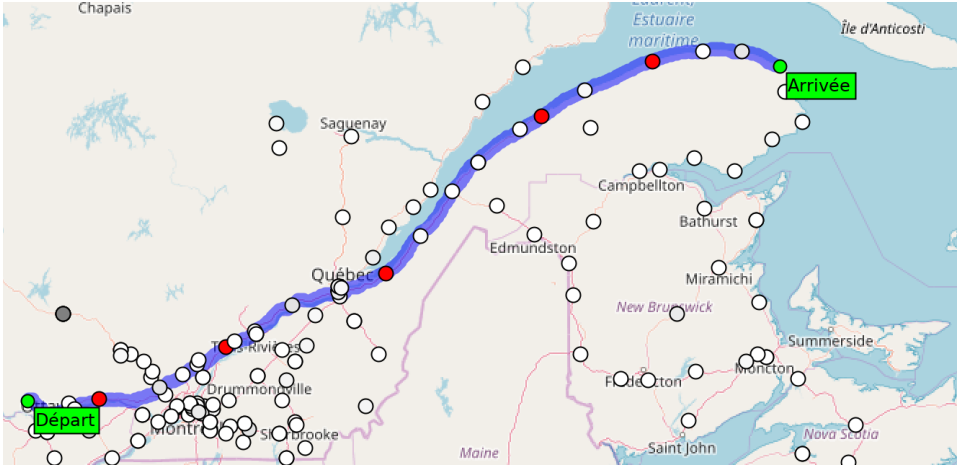
Problématique

Objectif

- Planification d'un VÉ de α vers ω dans un réseau routier ;
- Le VÉ a une autonomie ρ et doit se déplacer de stations en stations.
- Caractéristiques à respecter :
  - considère le temps d'attente à chaque borne ;
  - minimise le temps de calcul ;
  - freinage régénératif, etc



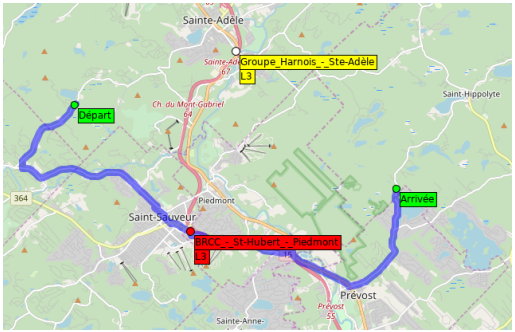
Exemple visuel de la problématique

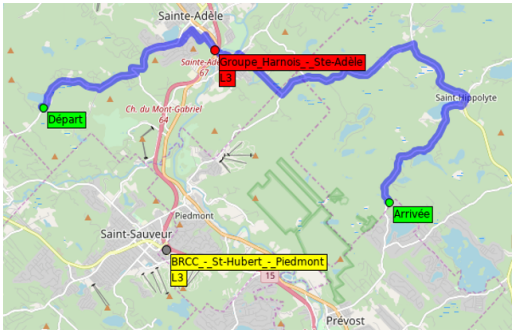


Temps d'attente — Considération à priori

Données d'occupation à priori

- Utilisation des données historiques de chaque borne *b*.
- Réétiquetage du graphe pour considérer ces données.



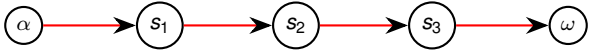


Chemin retourné par le planificateur Lundi midi et Mardi midi

Temps d'attente — Considération en temps réel

Génération de chemins alternatifs

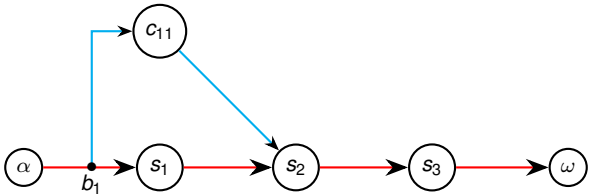
- Générer un chemin alternatif pour chaque borne sur le chemin initial.
- Utilisation d'une politique partielle pour choisir le chemin à suivre.



Temps d'attente — Considération en temps réel

Génération de chemins alternatifs

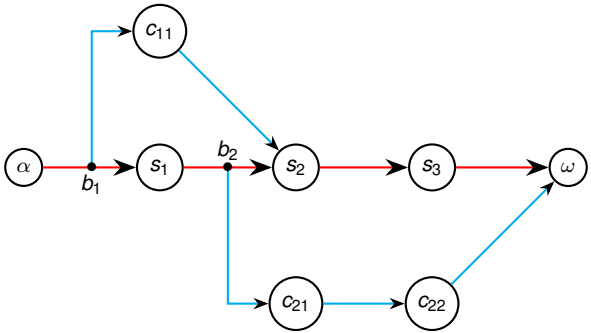
- Générer un chemin alternatif pour chaque borne sur le chemin initial.
- Utilisation d'une politique partielle pour choisir le chemin à suivre.



Temps d'attente — Considération en temps réel

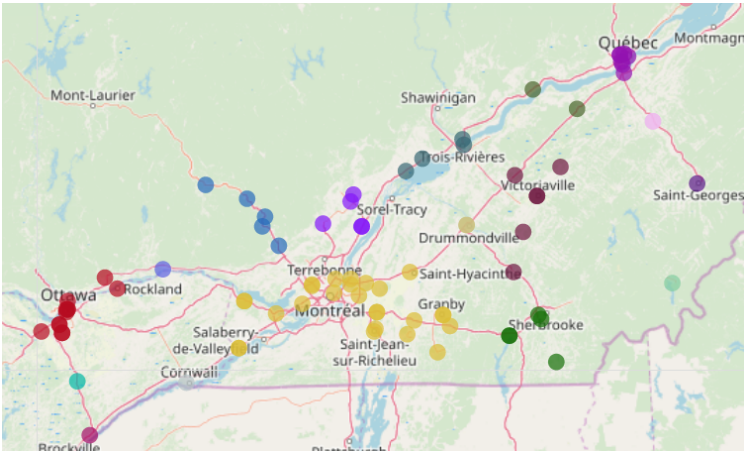
Génération de chemins alternatifs

- Générer un chemin alternatif pour chaque borne sur le chemin initial.
- Utilisation d'une politique partielle pour choisir le chemin à suivre.

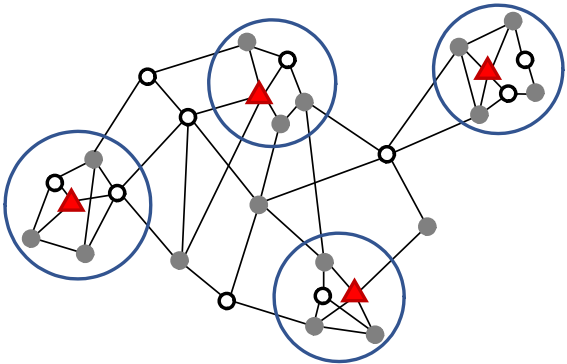


Regroupement de bornes 1

- La plupart des bornes sont situées dans les mêmes secteurs (grandes villes, sites touristiques, etc.) ;
- Regrouper les bornes rapprochées diminuerait grandement la taille du graphe.

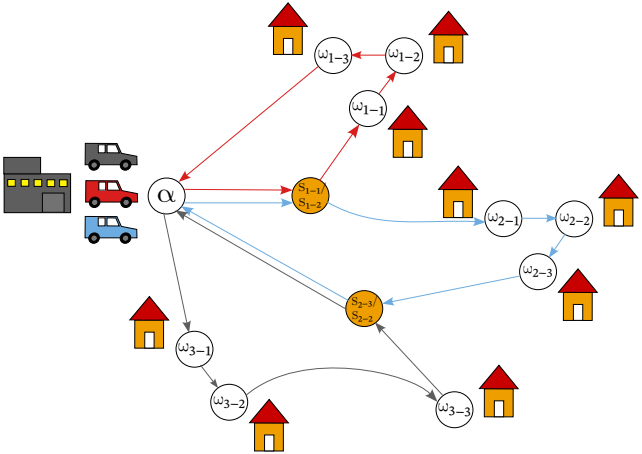


Regroupement de bornes 2

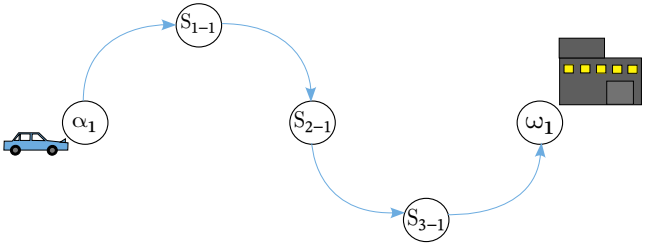


Planification pour une flotte de véhicules électriques

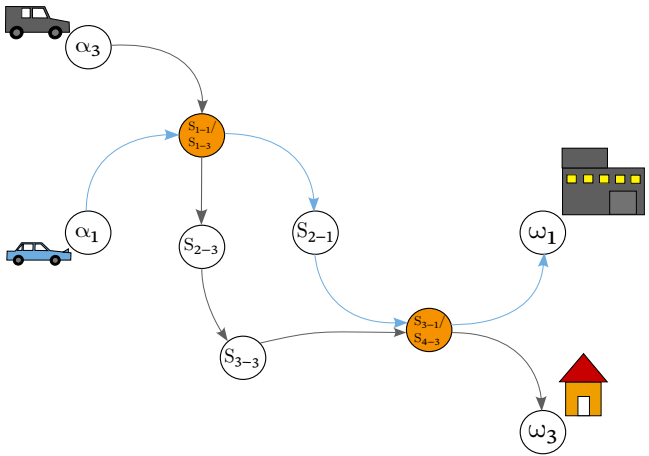
- Une flotte de VÉ contrôlée par une même entitée et ayant un même objectif ;
  - Ex. : livrer des colis d'un entrepôt à des clients ;



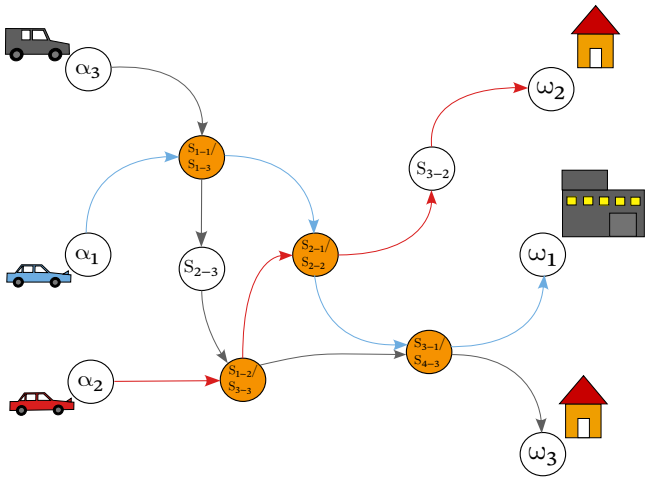
Planification coopérative — Exemple



Planification coopérative — Exemple



Planification coopérative — Exemple



Problème de planification coopérative pour véhicules électriques

- Il y a plusieurs VÉ, contrôlés par différents utilisateurs, chacun avec son propre objectif ;
- Il est souhaitable de planifier leurs itinéraires collectivement pour réduire les temps d'attente globaux ;
- Les conducteurs de VÉ peuvent envoyer une demande de planification à un planificateur centralisé ;
- De nouveaux VÉ peuvent entrer dans le problème de planification à tout moment ;
- En pratique, le planificateur peut recalculer un plan global
  - à chaque N nouvelles demandes au planificateur depuis le dernier recalcul ;
  - toutes les T minutes.

Plan

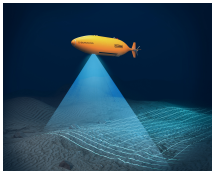
1 Planification d'itinéraires pour véhicules électriques
- Planification individuelle
- Planification coopérative

2 Planification de chemins couvrants

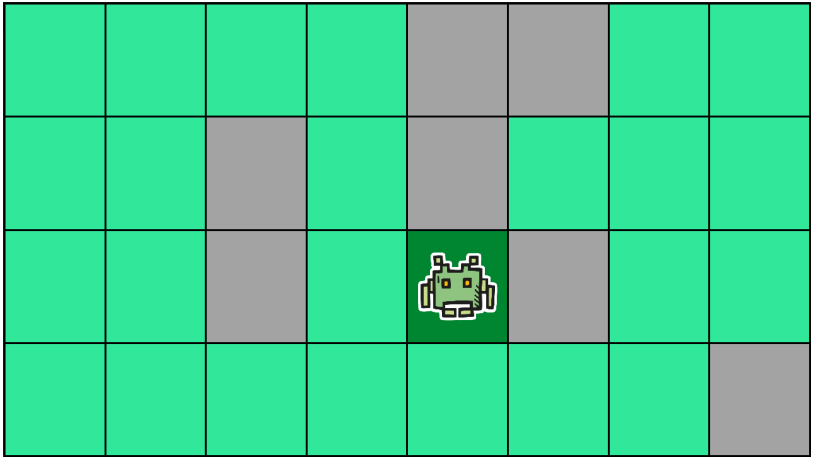
3 Processus décisionnels de Markov
- Introduction aux MDP
- CSR-MDP : Représentation mémoire des MDP
- pcTVI : Parallélisation de TVI
- eTVI et eiTVI : Exploitation de la hiérarchie de mémoire
- MDP synthétiques

Problématique

- Le problème de planification de chemins couvrants est un problème de planification de mouvement.
- Objectif : trouver une séquence minimale d'actions permettant à un agent de passer sur tous les points d'une zone ou d'un volume d'intérêt.
- Applications :
  - aspirateur robotique ;
  - impression 3D ;
  - déminage ;
  - véhicules autonomes sous-marins (AUV) ;
  - recherche et sauvetage ;
  - etc.



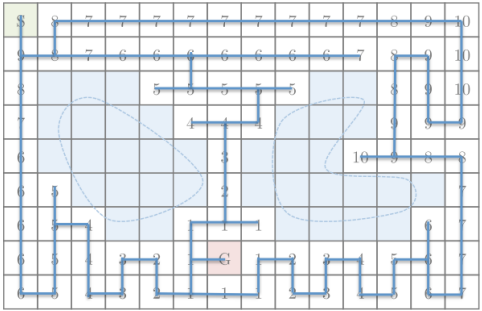
Visualisation



CPP dans des environnements discrets – Algorithme du front d'onde (Wavefront)

- Une vague est propagée du point d'arrivée souhaité ;
- L'agent se déplace vers le voisin non exploré ayant la plus grande valeur de la vague ;
- Pas de garantie d'optimalité.

S	8	7	7	7	7	7	7	7	7	7	8	9	10
9	8	7	6	6	6	6	6	6	6	7	8	9	10
8				5	5	5	5	5			8	9	10
7					4	4	4				9	9	9
6						3				10	9	8	8
6	5					2							7
6	5	4			1	1	1					6	7
6	5	4	3	2	1	G	1	2	3	4	5	6	7
6	5	4	3	2	1	1	1	2	3	4	5	6	7



Solution optimale

- L'espace d'états peut être représenté par un graphe orienté ;
  - Note : le nombre d'états est exponentiel en la taille de la grille.
  - **Recherche largeur** et variantes (Dijkstra, A^*, etc.) :
    - nécessite de pouvoir stocker l'espace d'états complet en mémoire.
  - **Recherche profondeur** :
    - peut aller arbitrairement profondément dans l'arbre de recherche, même si la solution est proche de la racine ;
    - risque de s'enliser dans des boucles.
- Le problème de planification de chemin couvrant optimal est NP-difficile.

Élagage de l'espace d'états

- Comme le problème est NP-difficile, explorer naïvement l'espace d'états est intractable.
- On peut toutefois élaguer des parties de l'arbre de recherche pour améliorer la performance du planificateur.
- Un type de sous-arbre non prometteur survient lors de la visite d'une cellule déjà visitée sans avoir exploré d'autres cellules depuis la dernière visite.
- On peut élaguer le sous-arbre lorsqu'un tel cycle est détecté.

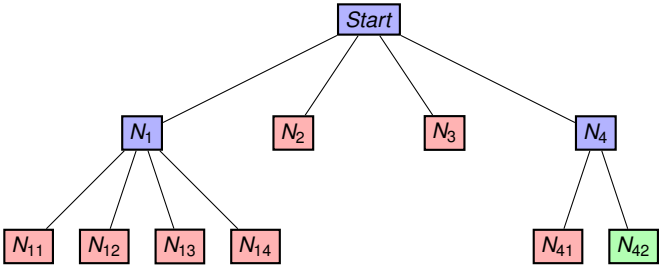
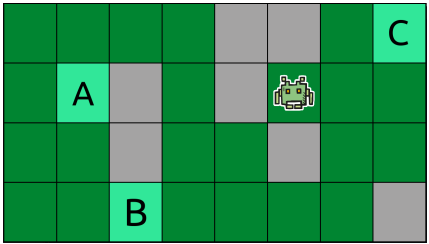


Figure – Exemple d'élagage de l'espace d'états. Les états explorés, élagués et objectifs sont respectivement bleu, rouges et verts.

Élagage par fonction heuristique

Exemple de fonction heuristique

- L'action "gauche" doit être utilisée au moins 4 fois pour atteindre A.
- On doit faire au moins $\max(4, 3, 0) = 4$ actions "gauche" pour terminer.
- Au total, au moins 4 + 2 + 2 + 1 = 9 actions sont nécessaires pour atteindre la solution optimale.
- Coût restant réel : 14 actions.
- Chaque action augmente le nombre d'actions opposées requises pour atteindre la solution optimale.
- $h(s) = 4 + 2 + min(4, 2) + 2 + 1 + min(2, 1) = 12$.



Plan

1 Planification d'itinéraires pour véhicules électriques
- Planification individuelle
- Planification coopérative

2 Planification de chemins couvrants

3 Processus décisionnels de Markov
- Introduction aux MDP
- CSR-MDP : Représentation mémoire des MDP
- pcTVI : Parallélisation de TVI
- eTVI et eiTVI : Exploitation de la hiérarchie de mémoire
- MDP synthétiques

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.

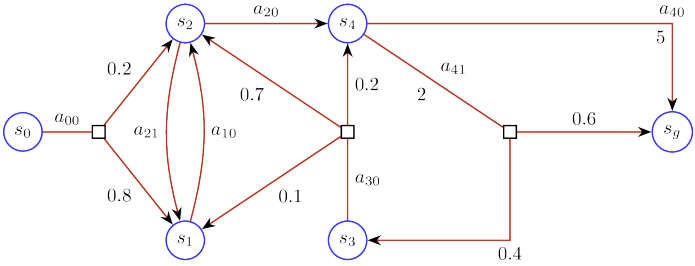
Exemples de domaines où les MDP sont utilisés

- 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.

Recherche : accélérer les algorithmes de MDP par l'exploitation des ordinateurs

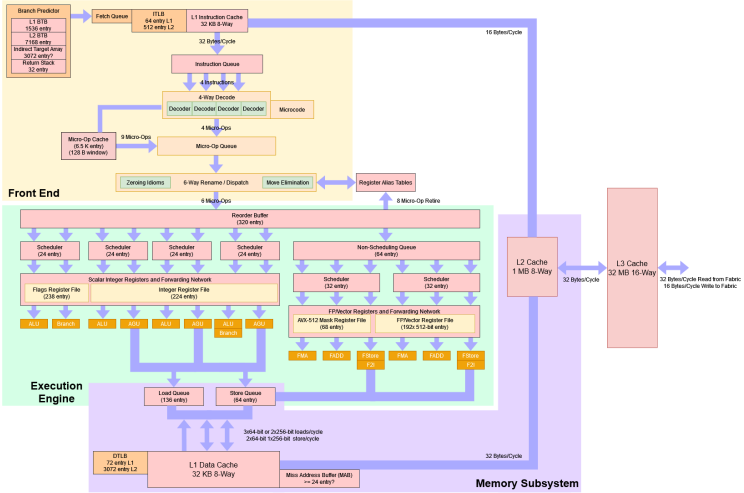
Idée

- 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.

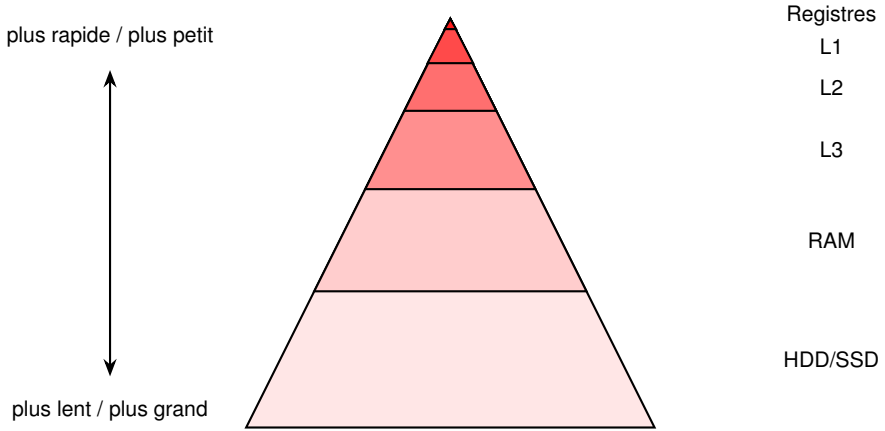
Objectif

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

Architecture moderne des ordinateurs – Schéma d'un processeur Skylake



Architecture moderne des ordinateurs – Hiérarchie de mémoire



Architecture moderne des ordinateurs

Type	Modèle	Taille	Lecture Go/s	Écriture Go/s	Latence ns
Cache L1d	AMD Ryzen 9 9900X	$12 \cdot 48$ ko	7506	3906	0.7
Cache L2		$12 \cdot 1$ Mo	2148	1934	2.5
Cache L3		64 Mo	1131	1145	7.7
Mémoire vive DDR5	Trident Z5 neo 6000Mhz	8-128 Go	75.7	76.9	62.7
Stockage de masse SSD	Samsung Evo 990 Pro	0.25-2 To	5.6	5.9	356 400
Stockage de masse HDD	Seagate IronWolf Pro	2-24 To	0.285	0.285	4 160 000

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.

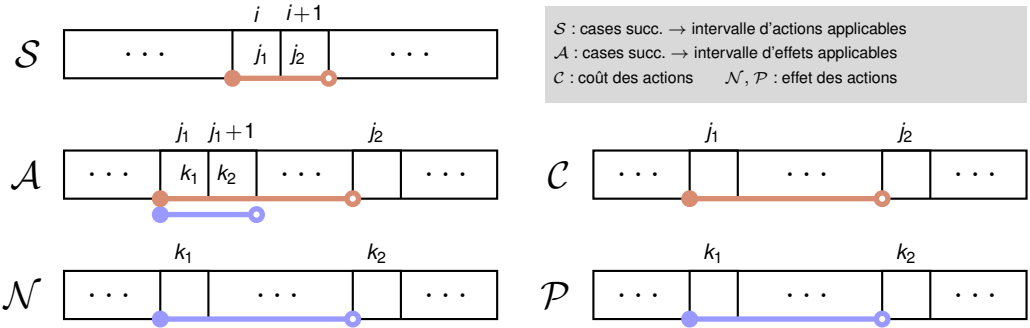
4. E. Bargiacchi, D. M. Roijers, and A. Nowé. AI-Toolbox : A C++ library for Reinforcement Learning and Planning (with Python Bindings). Journal of Machine Learning Research (2020), pp. 1-12.

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.

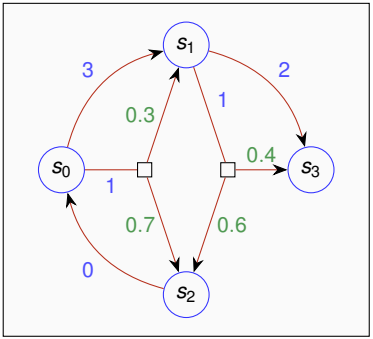
Représentation CSR-MDP

- 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 $\mathcal{N}, \mathcal{P}$: 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

$\mathcal{P}$

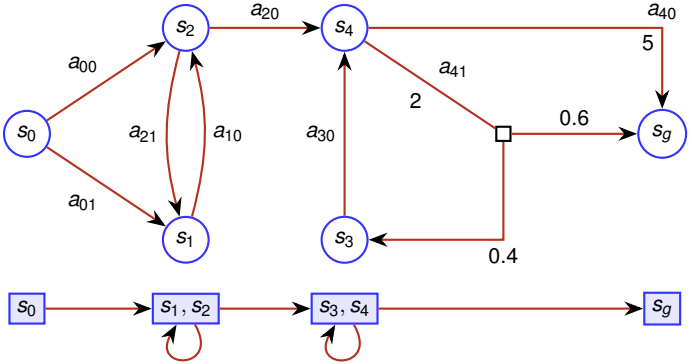
0	1	2	3	4	5	6
$\frac{10}{10}$	$\frac{3}{10}$	$\frac{7}{10}$	$\frac{10}{10}$	$\frac{4}{10}$	$\frac{6}{10}$	$\frac{10}{10}$

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*

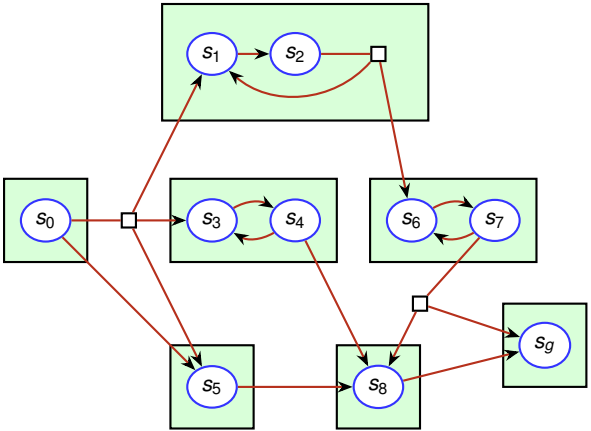
- Considère le graphe correspondant à la structure topologique du MDP ;
- Utilise l'algorithme de Tarjan pour décomposer le graphe en composantes fortement connexes (CFC) ;
- Si on considère les CFC dans l'ordre topologique inverse, on peut résoudre le MDP en un seul balayage.



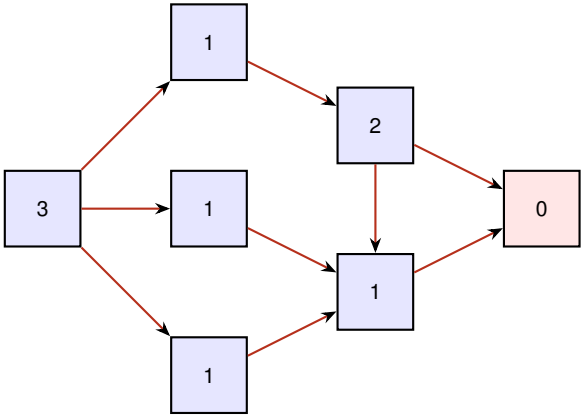
Algorithme pcTVI : *Parallel-Chained Topological Value Iteration*

- **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.

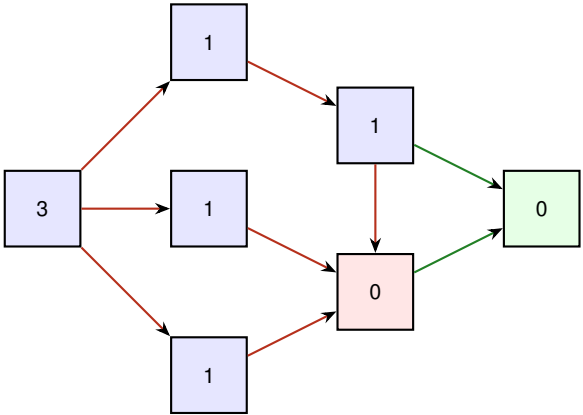
Exemple d'exécution de pcTVI



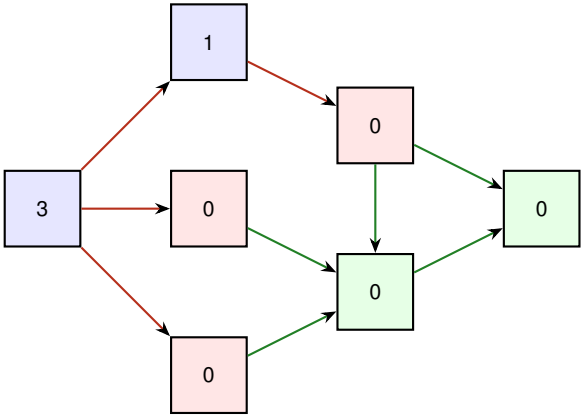
Exemple d'exécution de pcTVI



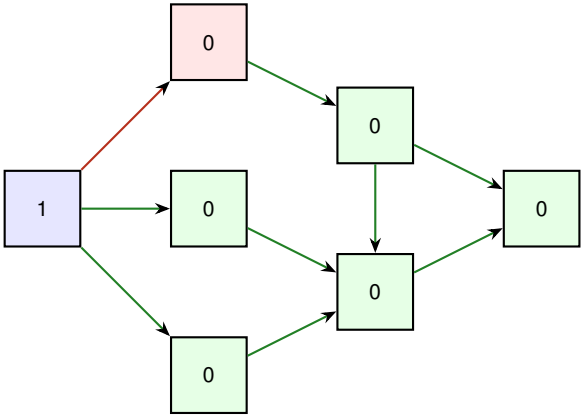
Exemple d'exécution de pcTVI



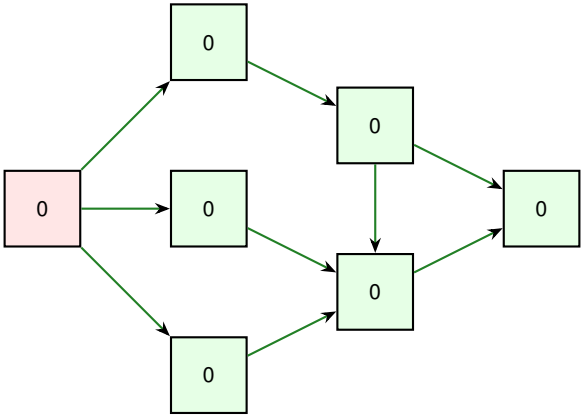
Exemple d'exécution de pcTVI



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*

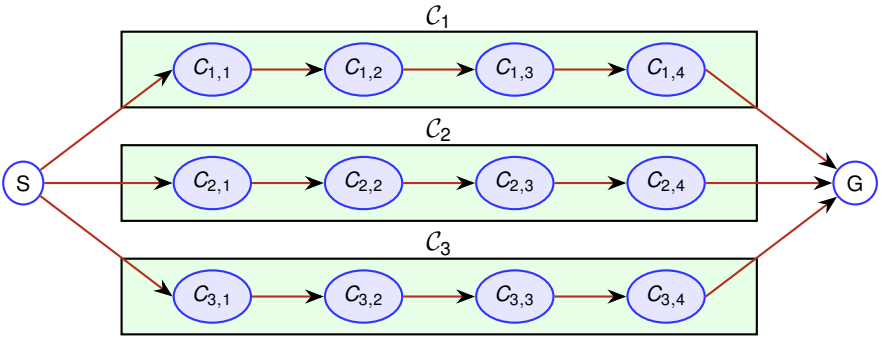


Figure – Instance du domaine *chaîné* où $n_c = 3$ et $n_C = 4$. Chaque ellipse représente une CFC.

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*

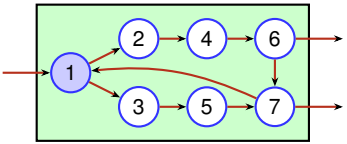
- L'algorithme **CEC** [8] est basé sur l'algorithme **FTVI** (*Focused Topological Value Iteration*).
- Il divise l'espace d'états en partis à l'aide d'un algorithme de partitionnement :
  - prendre un état aléatoire s dans la CFC courante ;
  - faire une recherche largeur depuis s ;
  - arrête lorsque 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

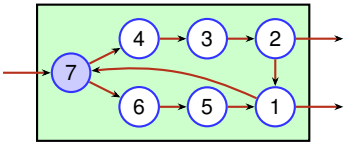
8. Jain, A. et Sahni, S. (2020). Cache efficient Value Iteration using clustering and annealing. Computer Communications, 159, 186-197. https://doi.org/10.1016/j.comcom.2020.04.058

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



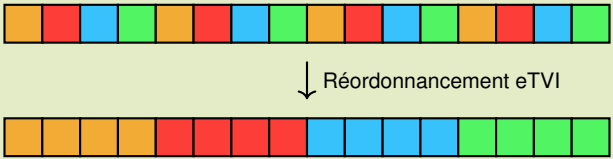
TVI : DFS postordre

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.

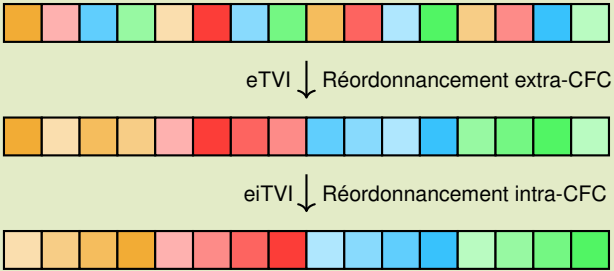


eiTVI : Réordonnancement des états en fonction de leur ordre interne aux CFC

- 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).

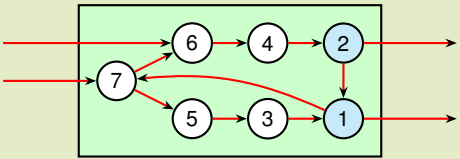


Quel est le meilleur ordre de balayage des états dans une CFC ?

- 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.



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).

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

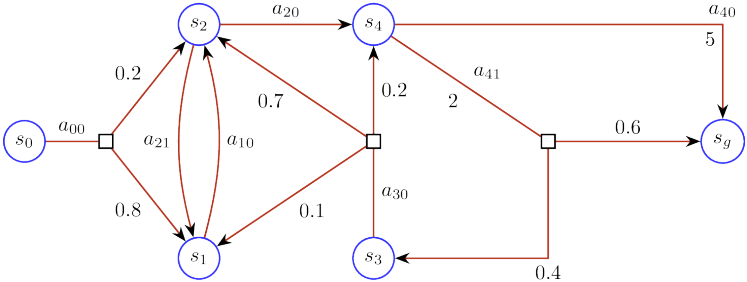
- Le **nombre d'états** $|S|$.
- Le **nombre d'actions** $|A|$.
- Le **nombre d'états buts** $|G|$.

- 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.

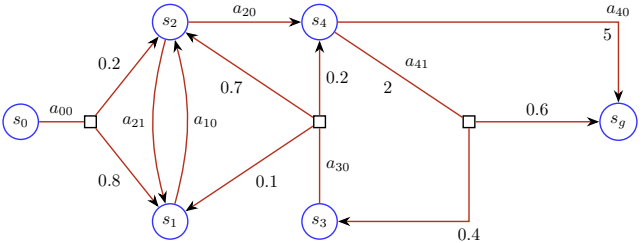
- La **distribution des transitions** : $\forall k, P_k^t :=$ proportion des actions qui ont k transitions probabilistes.
- L'**excentricité des états buts** : $\mathcal{G} := \min_{g\in G} \max_{s\in S} \bar{d}(s,g)$, où $\bar{d}(s,g)$ est le nombre minimum d'actions (le coût de chaque action n'est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

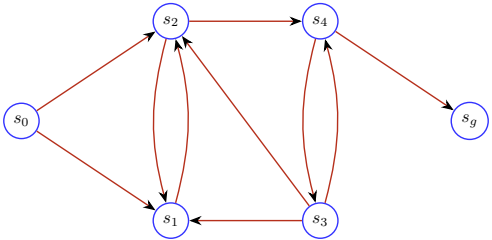
Exemple



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

Déterminisation d'un MDP





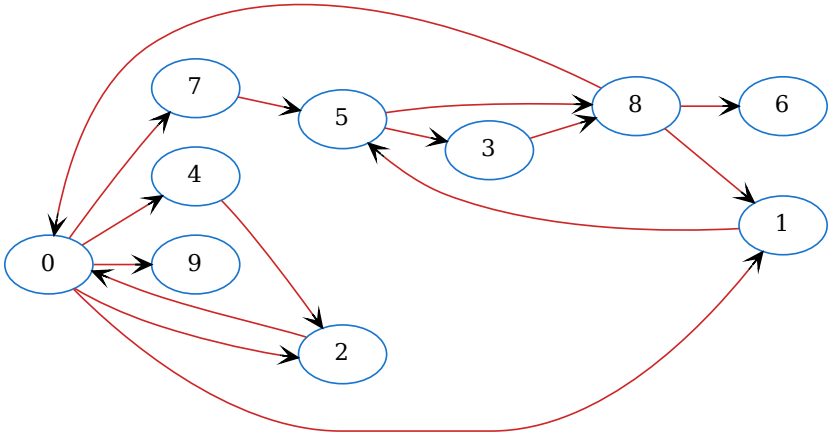
Génération synthétique de graphes

- 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.

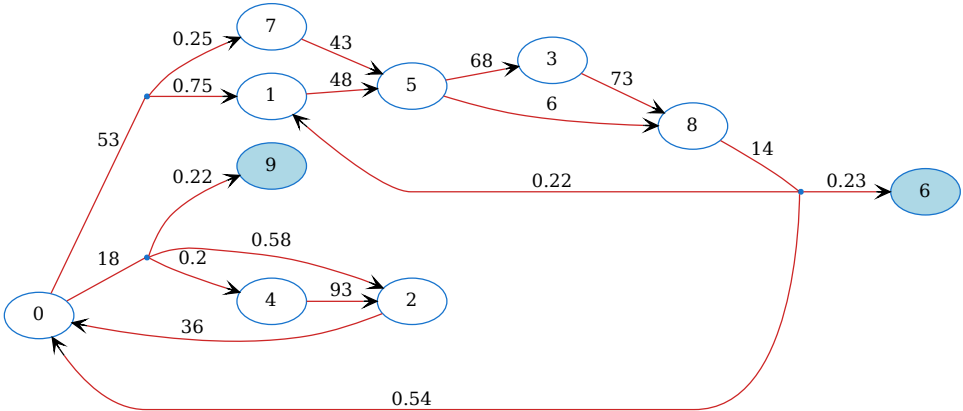
Modèle	**Distribution degrés**	**Coefficient de clustering**	**Diamètre**
Erdös-Rényi	binomiale	petit ($\bar{k}/n$)	petit : $\mathcal{O}(\log n)$
Watts-Strogatz	quasi constante	grand	petit
Barabási–Albert	invariant d'échelle ($\bar{k}^{-3}$)	grand ($\bar{k}^{-1}$)	petit : $\mathcal{O}(\frac{\log n}{\log(\log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

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



MDP synthétique généré à partir du graphe précédent



Conclusion

Recherche actuelle

- Conversion efficace de nombres à virgule flottante binaire en chaînes de caractères décimales.

Conclusion

Recherche actuelle

- Conversion efficace de nombres à virgule flottante binaire en chaînes de caractères décimales.

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.