Intégration de planification de mouvement et d'apprentissage par renforcement pour résoudre des problèmes d'exploration difficile

Soutenance de thèse

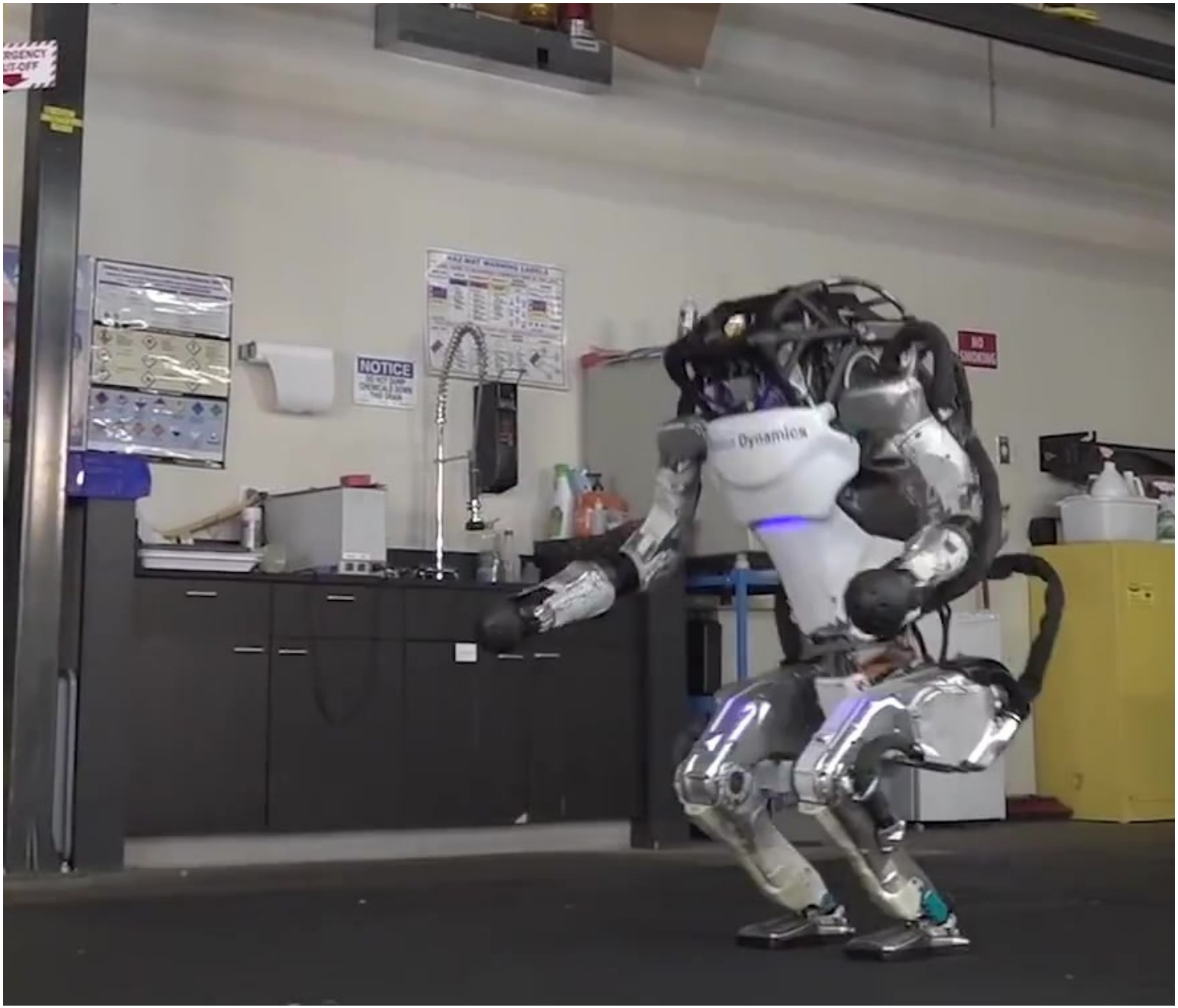
Guillaume Matheron, ISIR

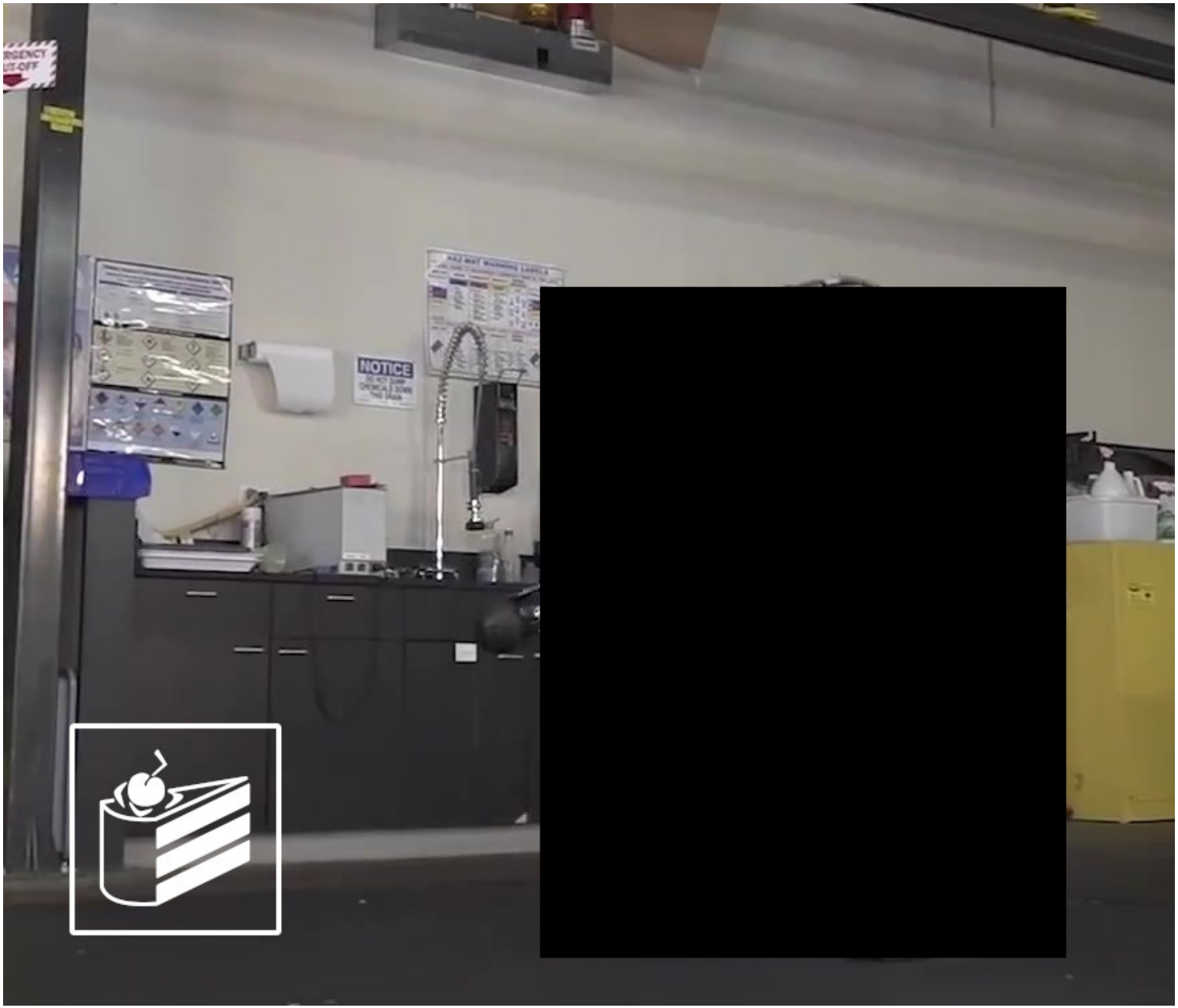
Thèse encadrée par Nicolas Perrin et Olivier Sigaud

Institut des Systèmes Intelligents et de Robotique

18 novembre 2020

Introduction


NOTICE
Dynamics


NOTICE

Formalisation

Robotique

Robot ramené à un point q dans l'espace des configurations [1].

Contraintes sur q ou sur $\dot{q}$.

Modélisation des forces, contacts, dynamique.

Étude séparée du robot et de l'environnement

Apprentissage par renforcement

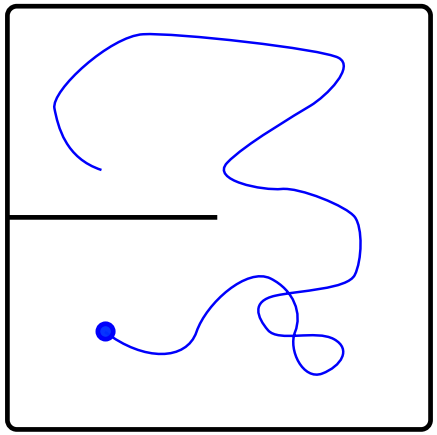
Problème de Décision Markovien (*MDP*) déterministe [2].

- S ensemble d'états du système.
- A ensemble d'actions.
- pas : $S \times A \to S$ fonction de transition.
- $R : S \times A \to \mathbb{R}$ récompense.

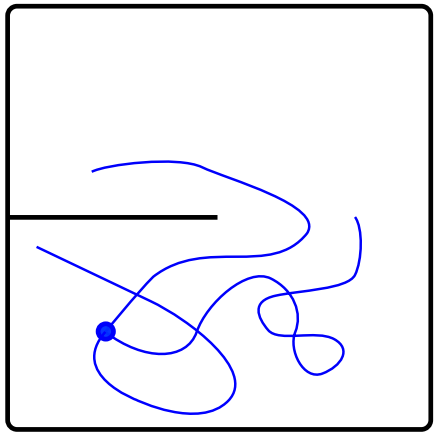
L'environnement comprend le robot. Il est traité comme une boîte noire.

1. Latombe, *Robot motion planning*, 1991.
2. Sutton et Barto, *Reinforcement Learning*, 2018.

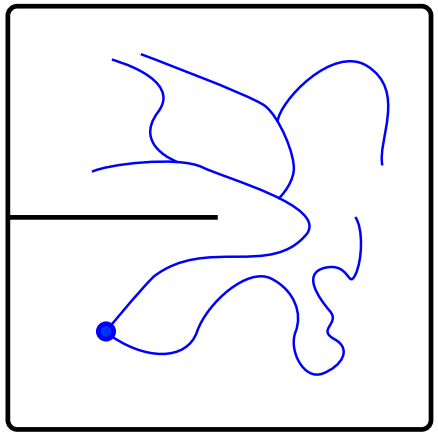
Niveaux d'accès à la fonction pas



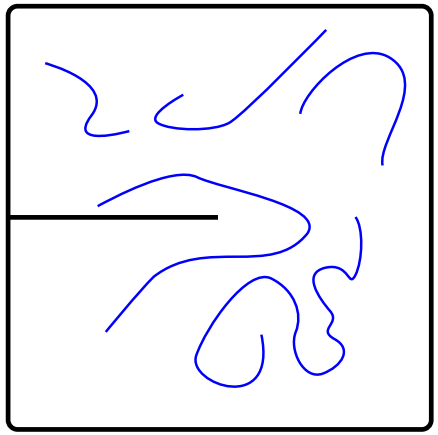
Réel



Par épisode



Réinitialisation aux états connus



Réinitialisation n'importe où (*reset-anywhere*)

Hypothèse principale

Entraînements avec possibilité de réinitialiser partout, tests épisodiques.

Cadre de l'étude

Portée limitée à la simulation (pas de robots réels dans cette thèse)

Analyse des mécanismes d'AR et de PM.

Applications :

- Directes : visualisation, cinéma [3], jeux vidéo [4].
- Transfert au réel [5].

3. Jaedong Lee, Won et Jehee Lee, ≪ Crowd simulation by deep reinforcement learning ≫, 2018.
4. Sharifi, Zhao et Szafron, ≪ Learning Companion Behaviors Using Reinforcement Learning in Games ≫, 2010.
5. Christiano et al., ≪ Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model ≫, 2016.

Apprentissage par renforcement

Récompense d'une trajectoire $\tau = s_0, \dots s_N$ **:** $R(\tau) = \sum_{i=0}^{N} \gamma^i R(s_i)$.

Politique : $\pi : S \to A$.

Récompense d'une politique π **:**
$R^\pi(s_0) = R(s, \pi(s)) + \gamma R^\pi(\mathtt{pas}(s, \pi(s)))$.

On veut trouver : $\mathrm{argmax}_\pi \, R^\pi(s_0)$.

Si S est continu alors π est paramétrée par ψ :

$$\underset{\psi}{\mathrm{argmax}} \, R^{\pi_\psi}(s_0)$$

Méthodes d'optimisation

$$\dim \psi \approx 600$$

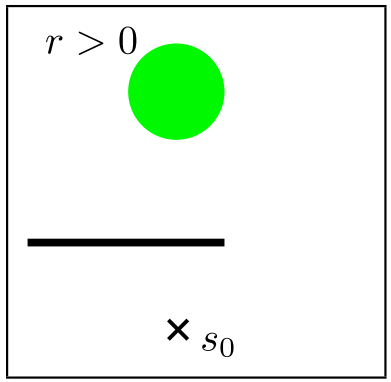
- Algorithmes évolutionnistes $\rightarrow$ en général moins rapides.
- Méthodes combinatoires $\rightarrow$ ne passent pas à l'échelle
- Méthodes de gradient, se combinent bien avec les réseaux de neurones.

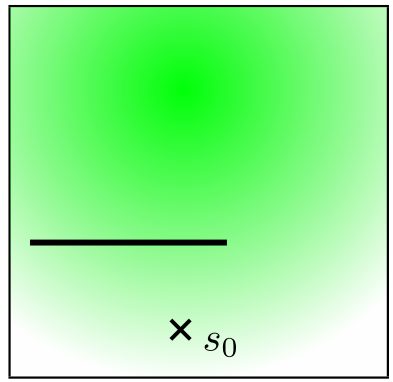
Les méthodes de gradient de politique estiment $\nabla_\psi R^{\pi_\psi}(s_0)$.
Mais R dépend de la fonction `pas` qui n'est pas différentiable !

Densité de récompense et impact sur le gradient

Définition de la récompense :

1. Éparse $\rightarrow$ $R > 0$ au moment où la tâche est réussie, sinon $R = 0$.
2. Dense $\rightarrow$ on guide avec la récompense.

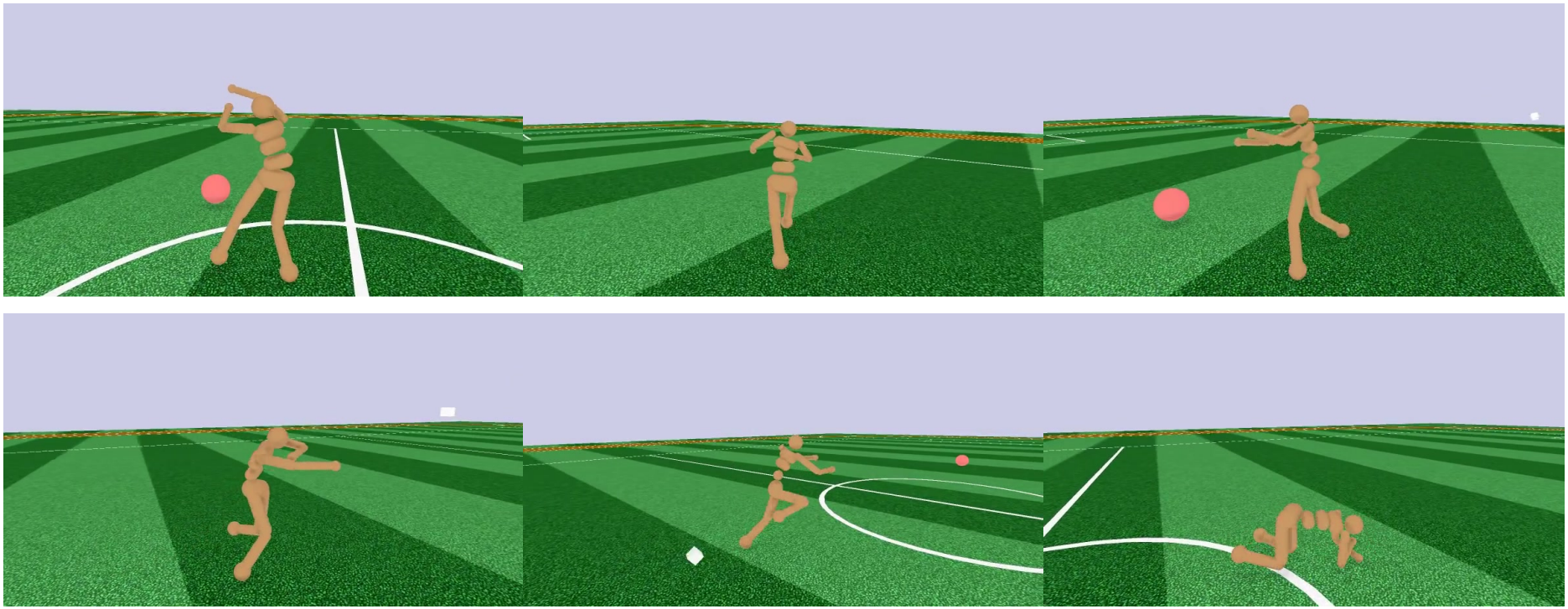




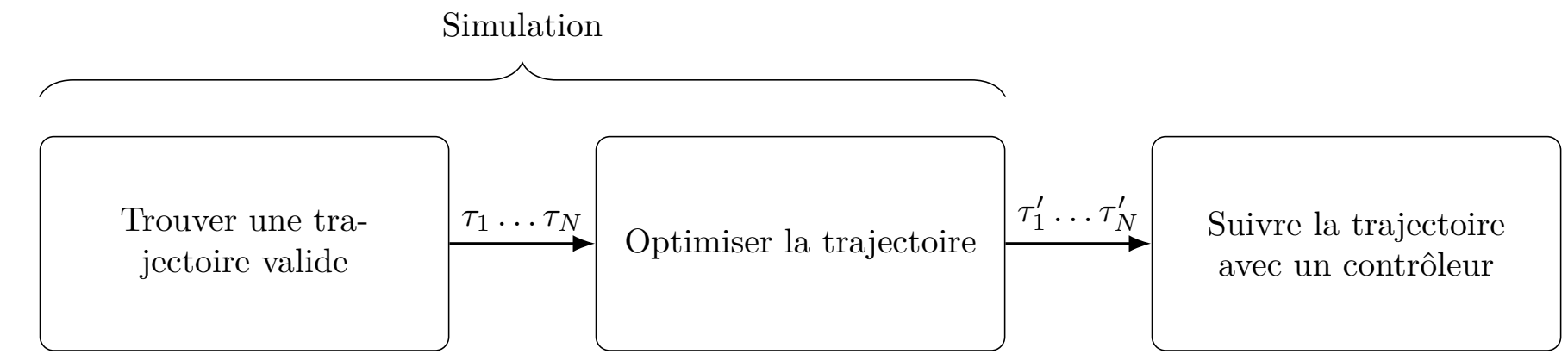
Problème : on veut guider le moins possible pour garder une solution générale.

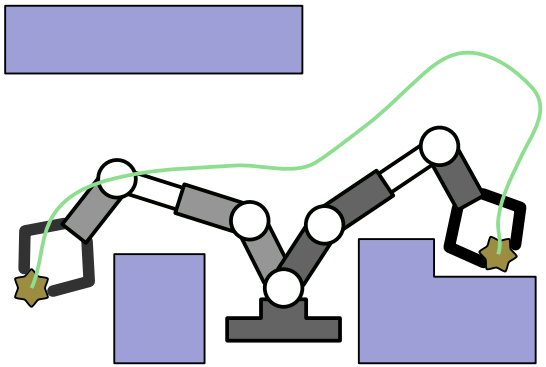
Mais si la récompense est éparse, en général $\nabla_\psi R^{\pi_\psi}(s_0) = 0$.

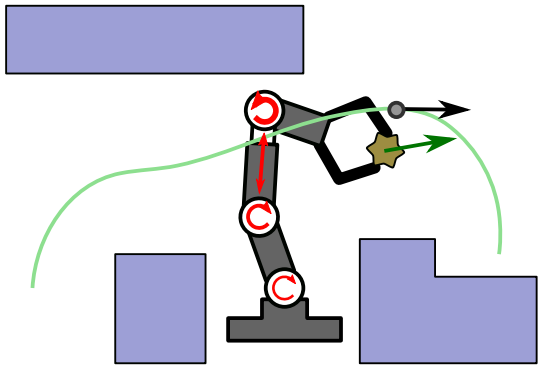
Apprentissage par renforcement

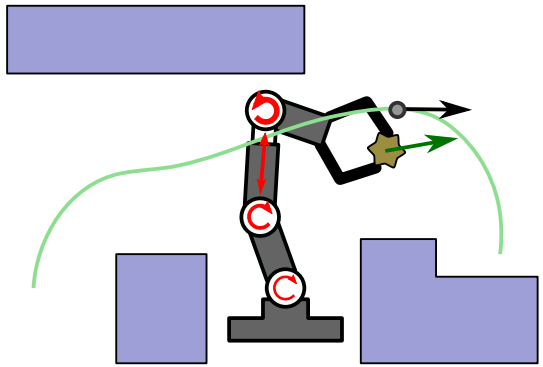


Planification de mouvement

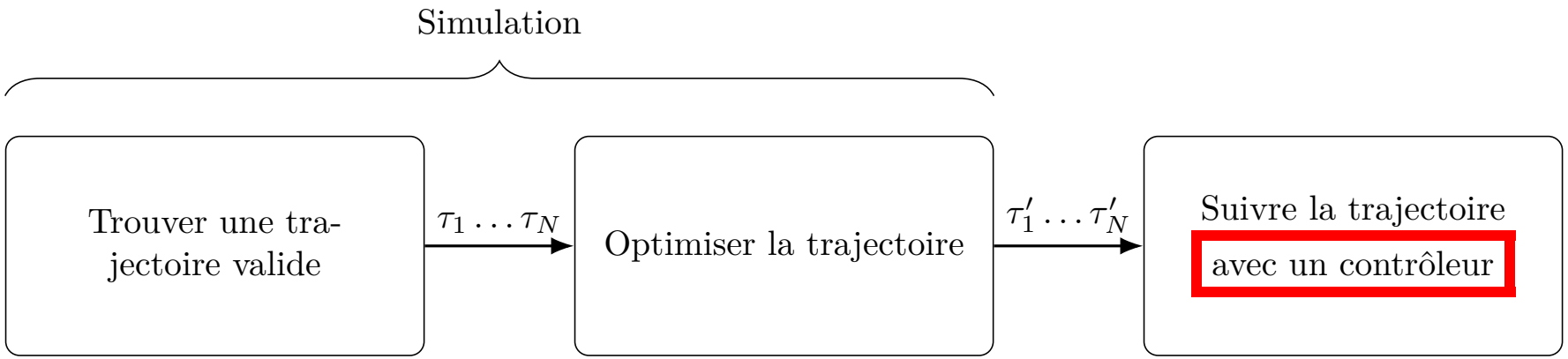


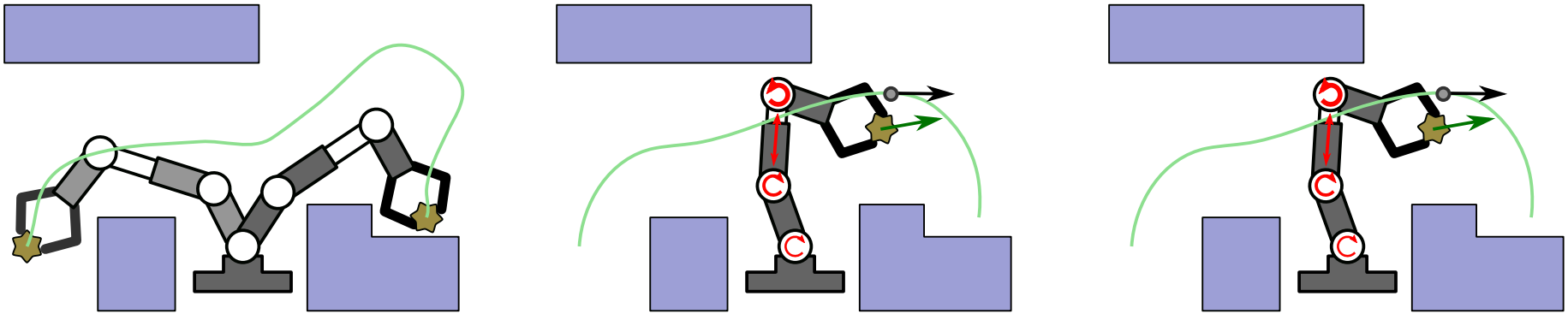






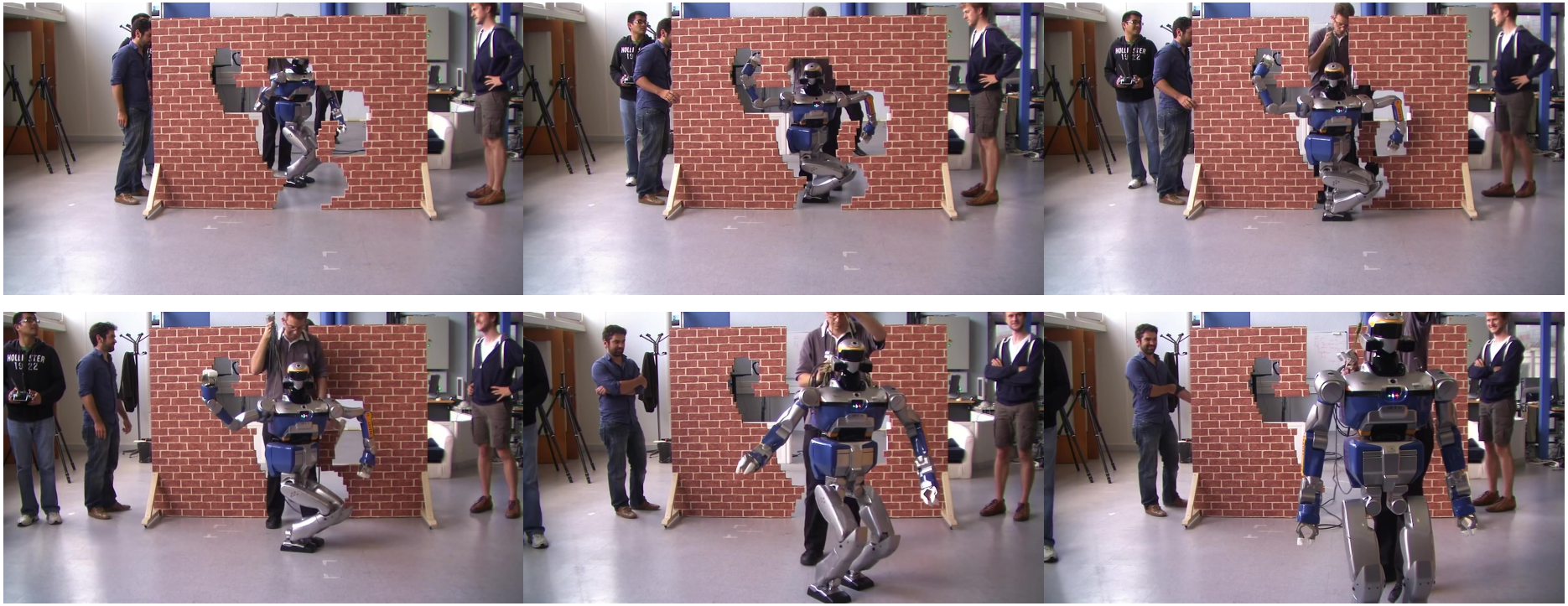
Planification de mouvement





Mais pour ça il faut un modèle inverse.

Planification de mouvement



Deux approches complémentaires

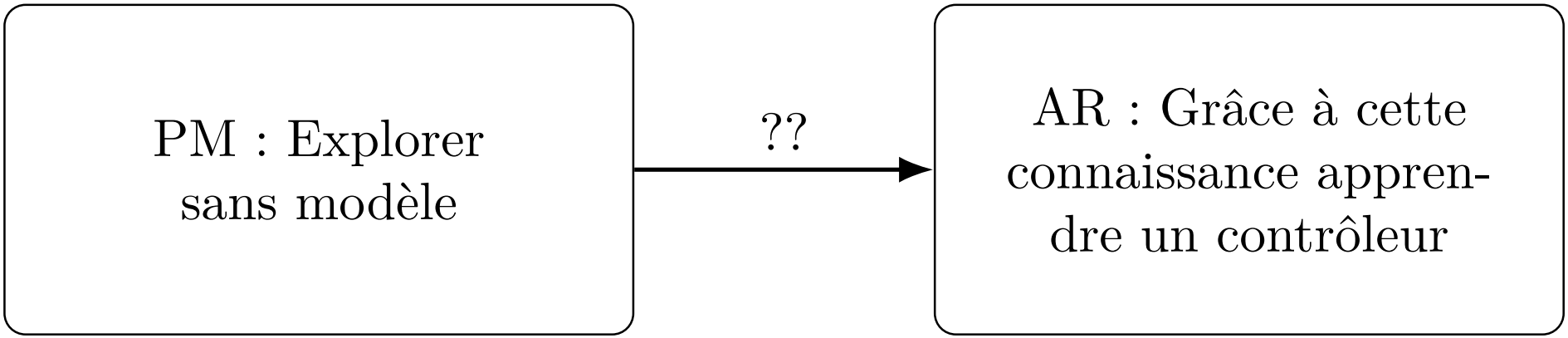
Planification de mouvement

- Explore efficacement
- Renvoie une trajectoire et nécessite un modèle pour la suivre

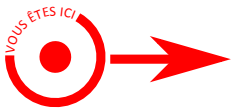
Apprentissage par renforcement

- Explore mal en l'absence de récompense
- Apprend des contrôleurs

Approche retenue :

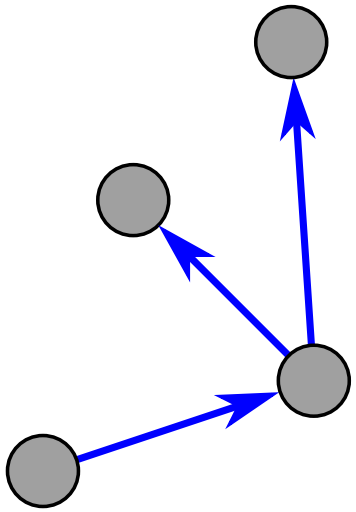


- Introduction
  - Cadre
  - Apprentissage par renforcement
  - Planification de mouvement



- Phase 1 : planification de mouvement
  - RRT, EST, Ex
- Phase 2 : apprentissage par renforcement
  - DDPG
  - Transfert d'expérience
  - Plan, Backplay
  - Cas de blocage de DDPG
  - Plan, Backplay, Chain Skills
- Conclusion

Rapidly-exploring Random Trees (RRT) [6]

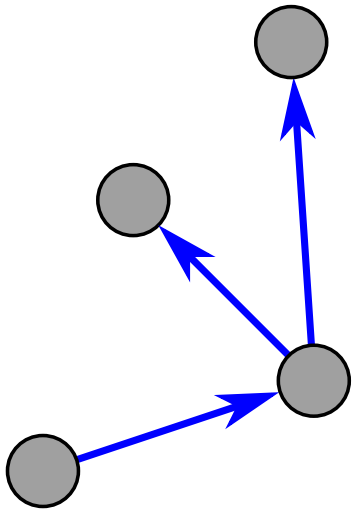


6. Lavalle, *Rapidly-exploring random trees*, 1998.

Rapidly-exploring Random Trees (RRT) [6]

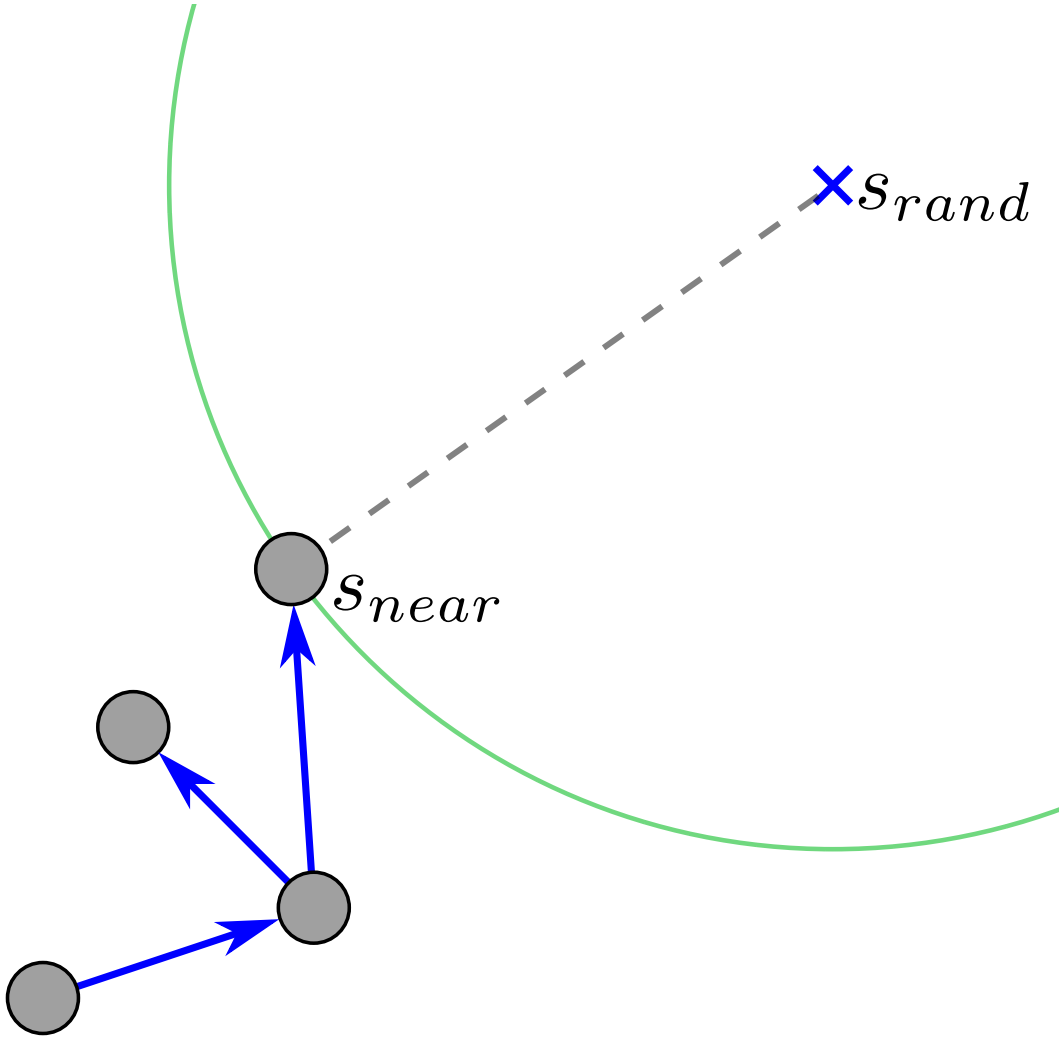
$\times s_{rand}$

Il faut une estimation de l'espace accessible



6. Lavalle, *Rapidly-exploring random trees*, 1998.

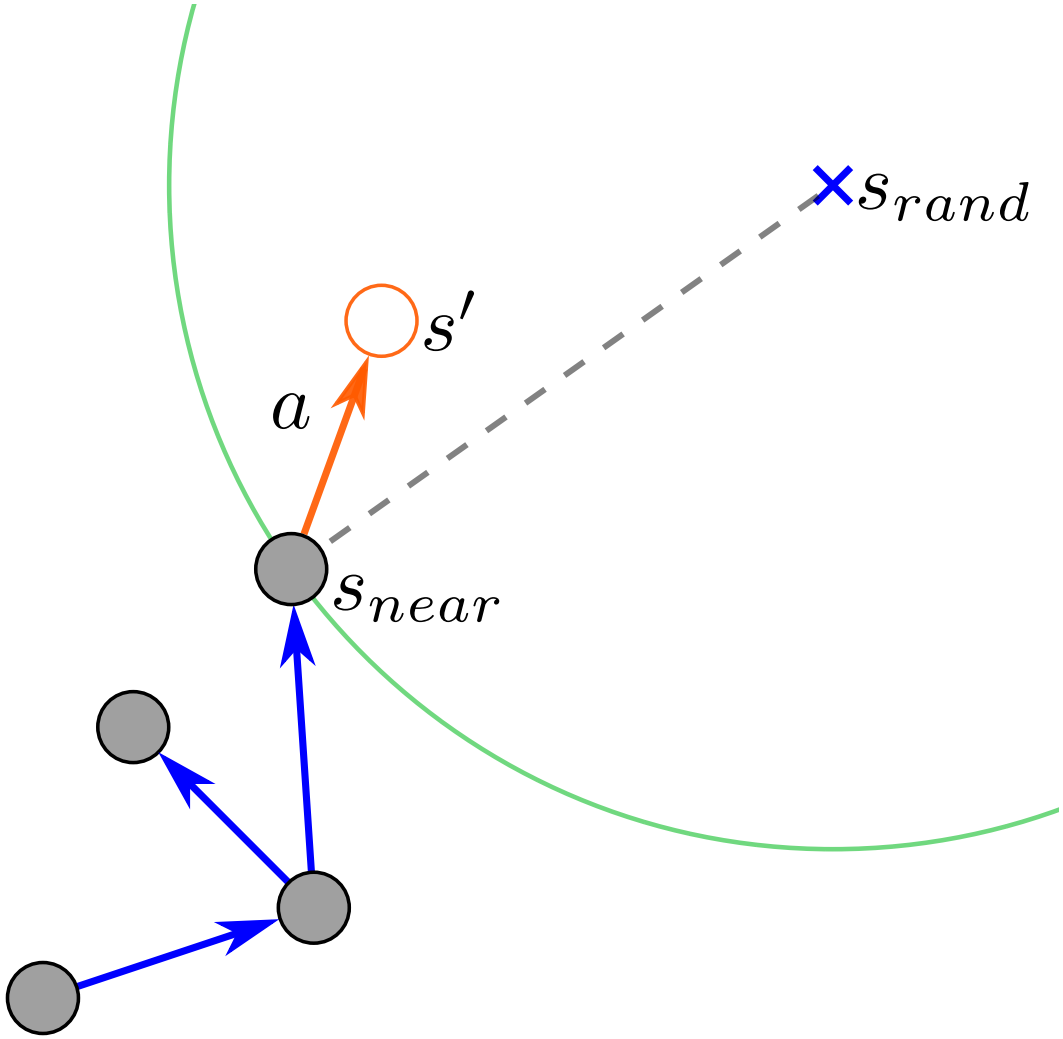
Rapidly-exploring Random Trees (RRT) [6]



Il faut une estimation de l'espace accessible

6. Lavalle, *Rapidly-exploring random trees*, 1998.

Rapidly-exploring Random Trees (RRT) [6]

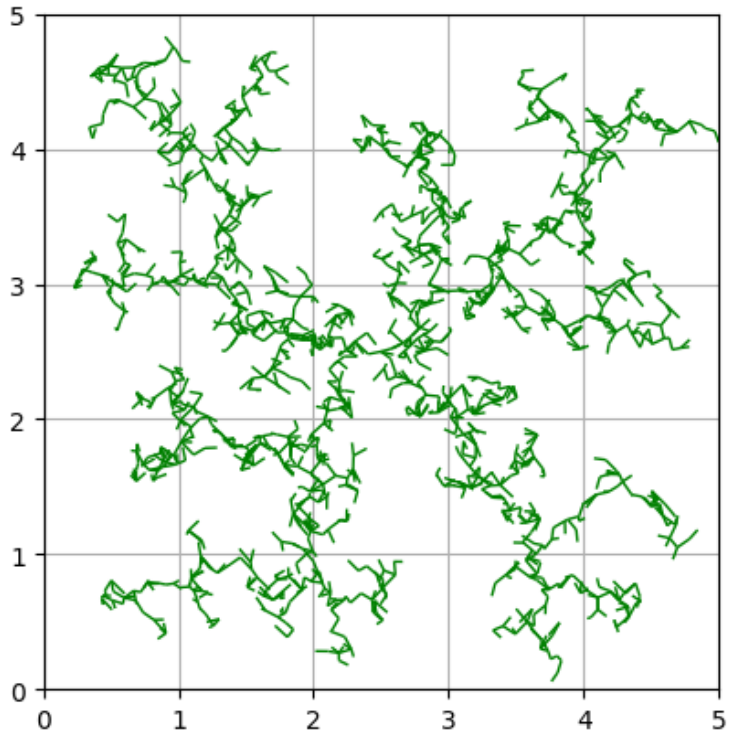


Il faut une estimation de l'espace accessible

Choix de a avec un modèle inverse ou une heuristique sinon au hasard

6. Lavalle, *Rapidly-exploring random trees*, 1998.

Les limites de RRT

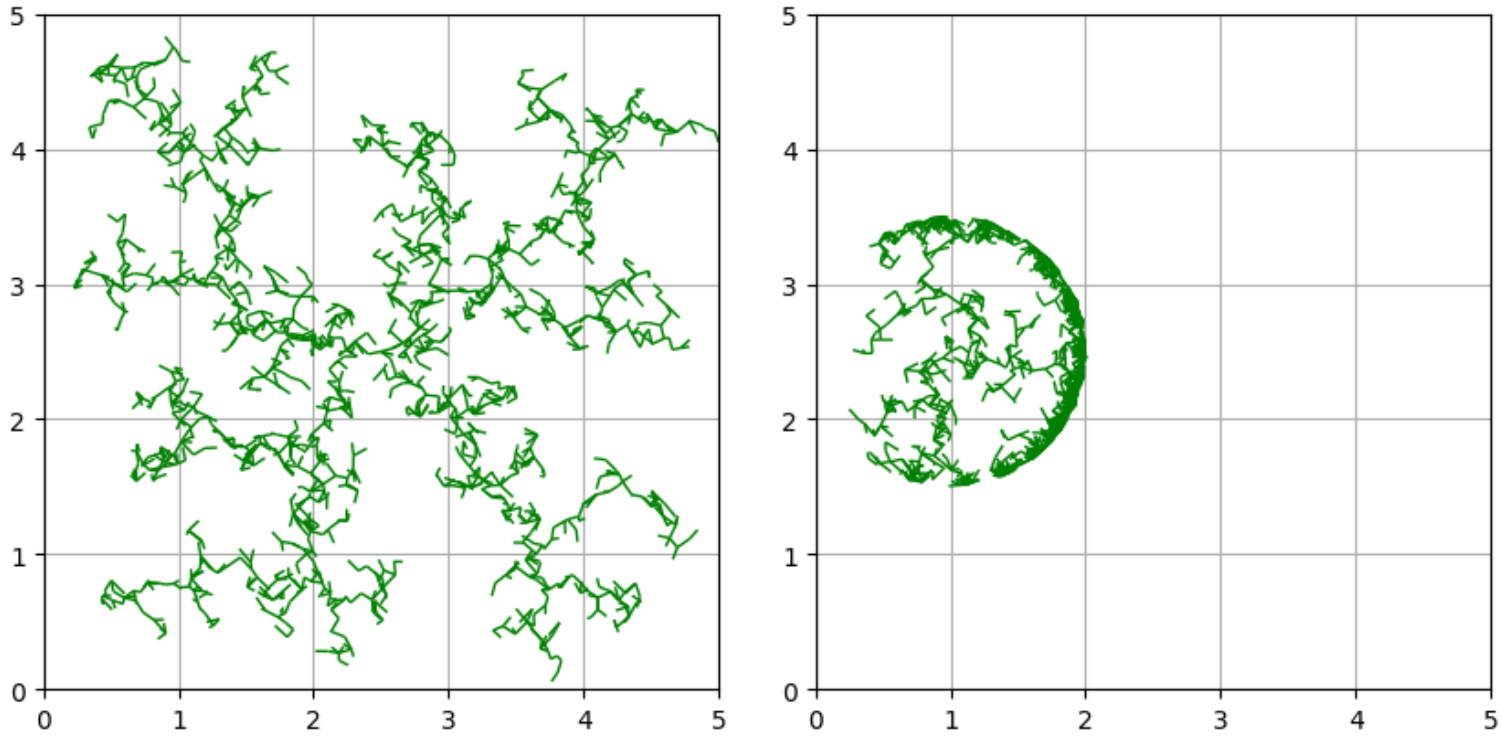


(a) (b)

FIGURE – 2000 itérations de RRT avec actions aléatoires, lorsque l'espace d'échantillonnage est (a) égal ou (b) plus grand que l'espace accessible.

L'échantillonnage donne à RRT des directions d'exploration.

Les limites de RRT



(a) (b)

FIGURE – 2000 itérations de RRT avec actions aléatoires, lorsque l'espace d'échantillonnage est (a) égal ou (b) plus grand que l'espace accessible.

L'échantillonnage donne à RRT des directions d'exploration.

Retirons l'échantillonnage : marche aléatoire

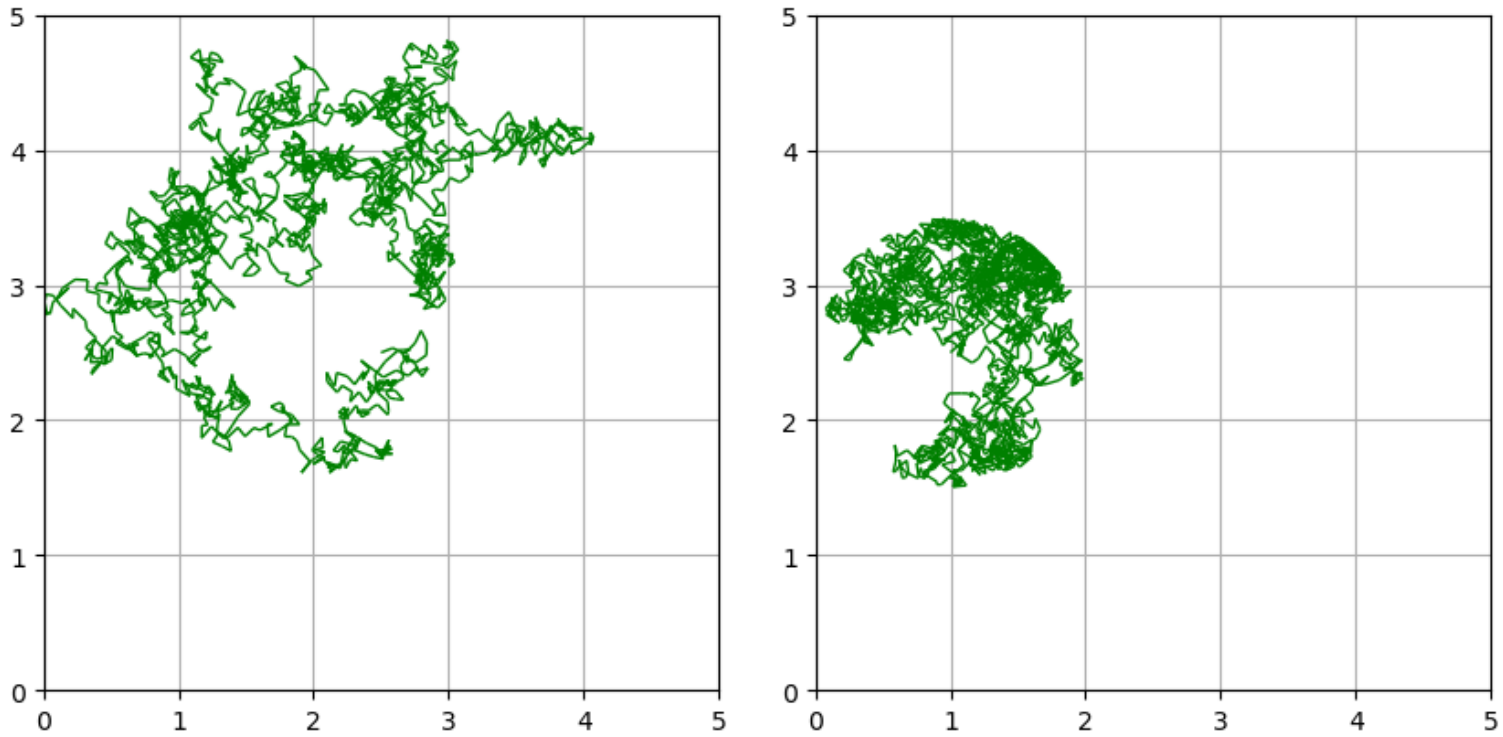
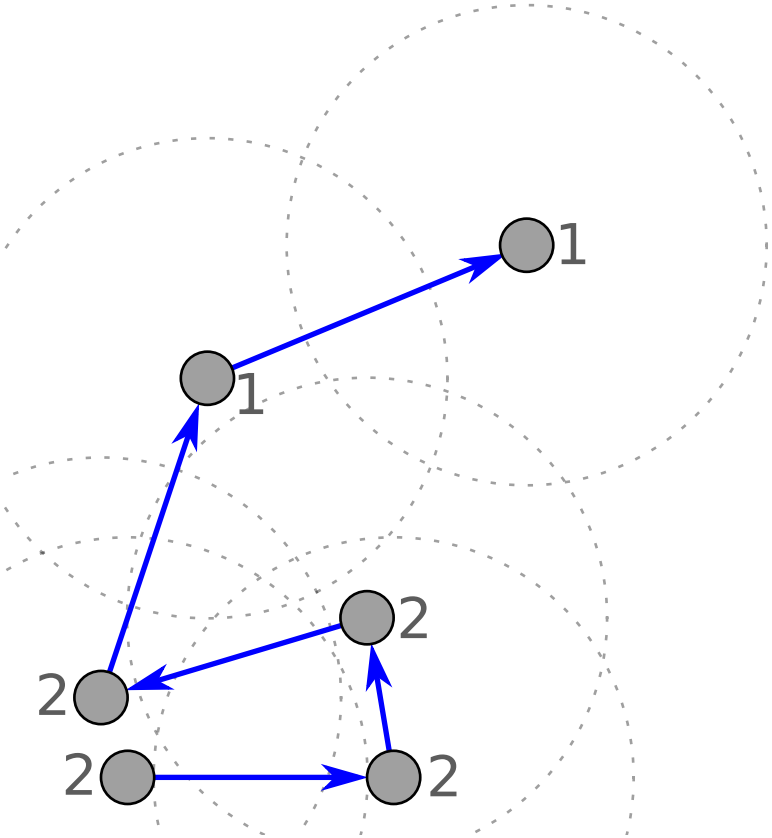


FIGURE – 2000 itérations de marche aléatoire, lorsque l'espace d'échantillonnage est (a) égal ou (b) plus grand que l'espace accessible.

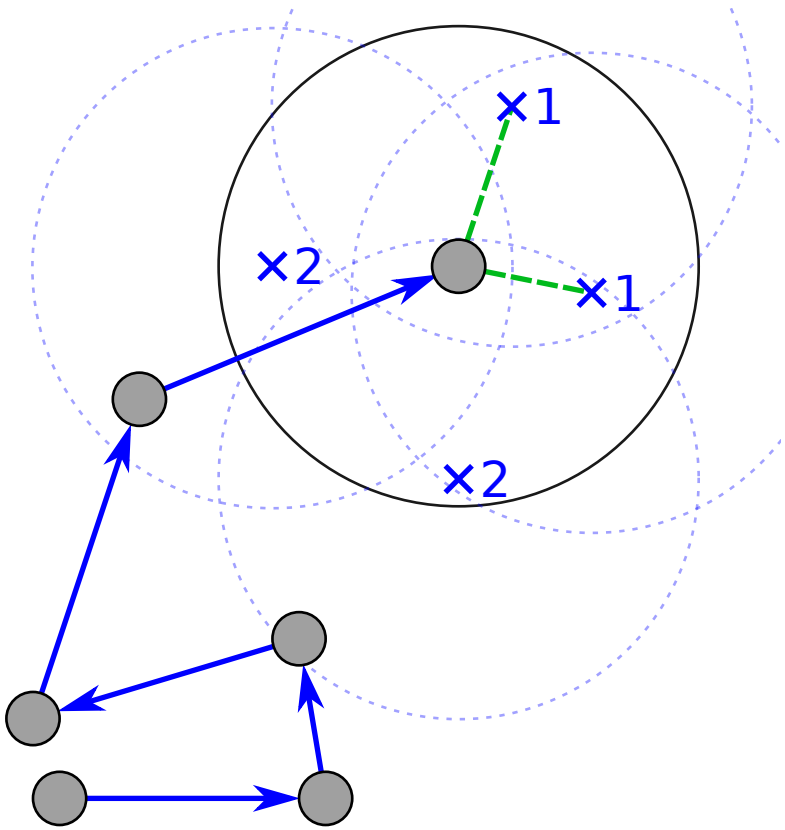
Éviter l'agglutinement avec EST [7]



- Choisir un noeud dans une zone peu dense

7. Hsu, Latombe et Motwani, « Path planning in expansive configuration spaces », 1997.

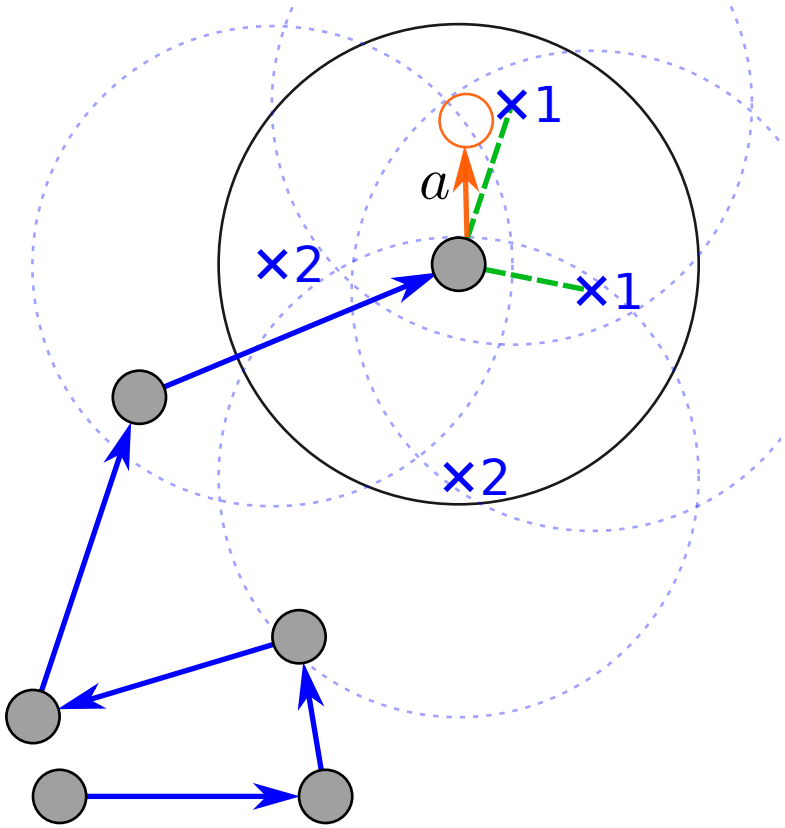
Éviter l'agglutinement avec EST [7]



- Choisir un noeud dans une zone peu dense
- Sélectionner des cibles proches

7. Hsu, Latombe et Motwani, « Path planning in expansive configuration spaces », 1997.

Éviter l'agglutinement avec EST [7]

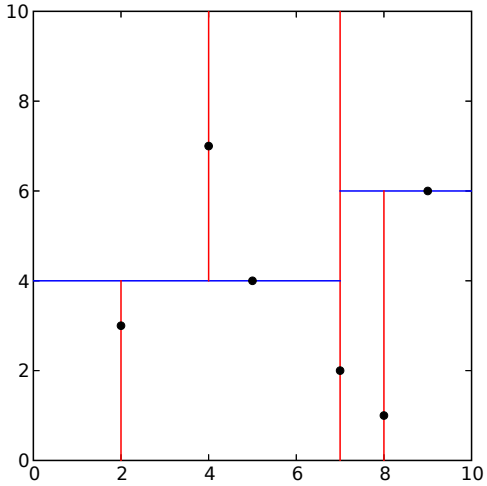


- Choisir un noeud dans une zone peu dense
- Sélectionner des cibles proches
- Retenir les cibles ayant une faible densité locale
- Utiliser une heuristique pour aller vers une de ces cibles

7. Hsu, Latombe et Motwani, « Path planning in expansive configuration spaces », 1997.

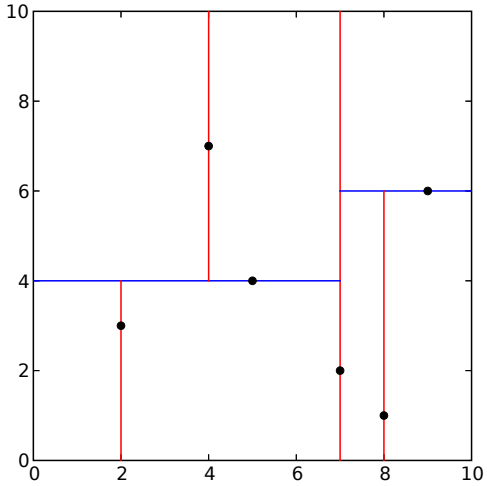
Problèmes :

- La recherche des plus proches voisins est en $O(\log n)$ avec des arbres Kd.
- Blocages intrinsèques
- Sans heuristique, seule la première étape est nécessaire.



Problèmes :

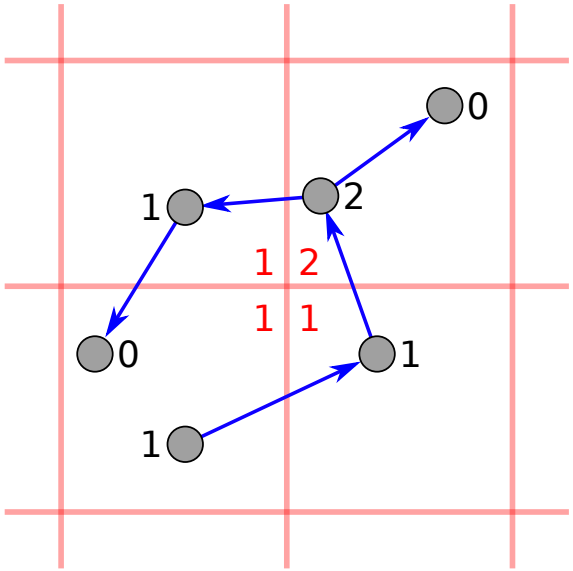
- La recherche des plus proches voisins est en $O(\log n)$ avec des arbres Kd.
- Blocages intrinsèques
- Sans heuristique, seule la première étape est nécessaire.



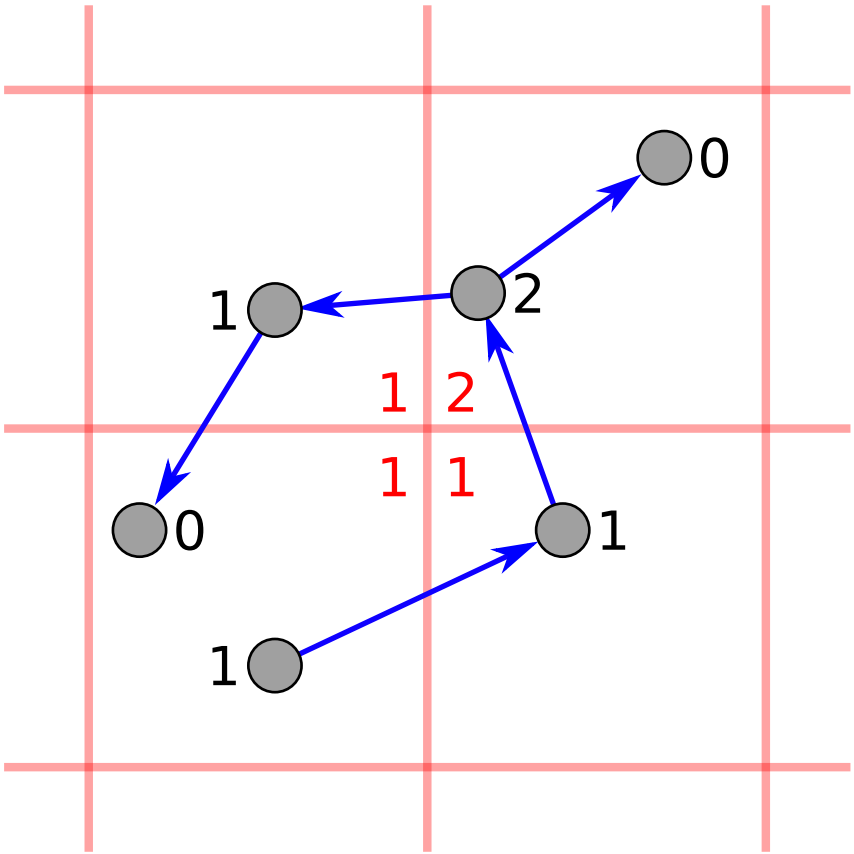
Notre proposition : Ex.

- Composante spatiale : arbre à 1 niveau (*grid-based*) avec un espace des caractéristiques (*feature space*).
- Compteur d'essais : on compte les tentatives.

Résultat : algorithme en $O(1)$.



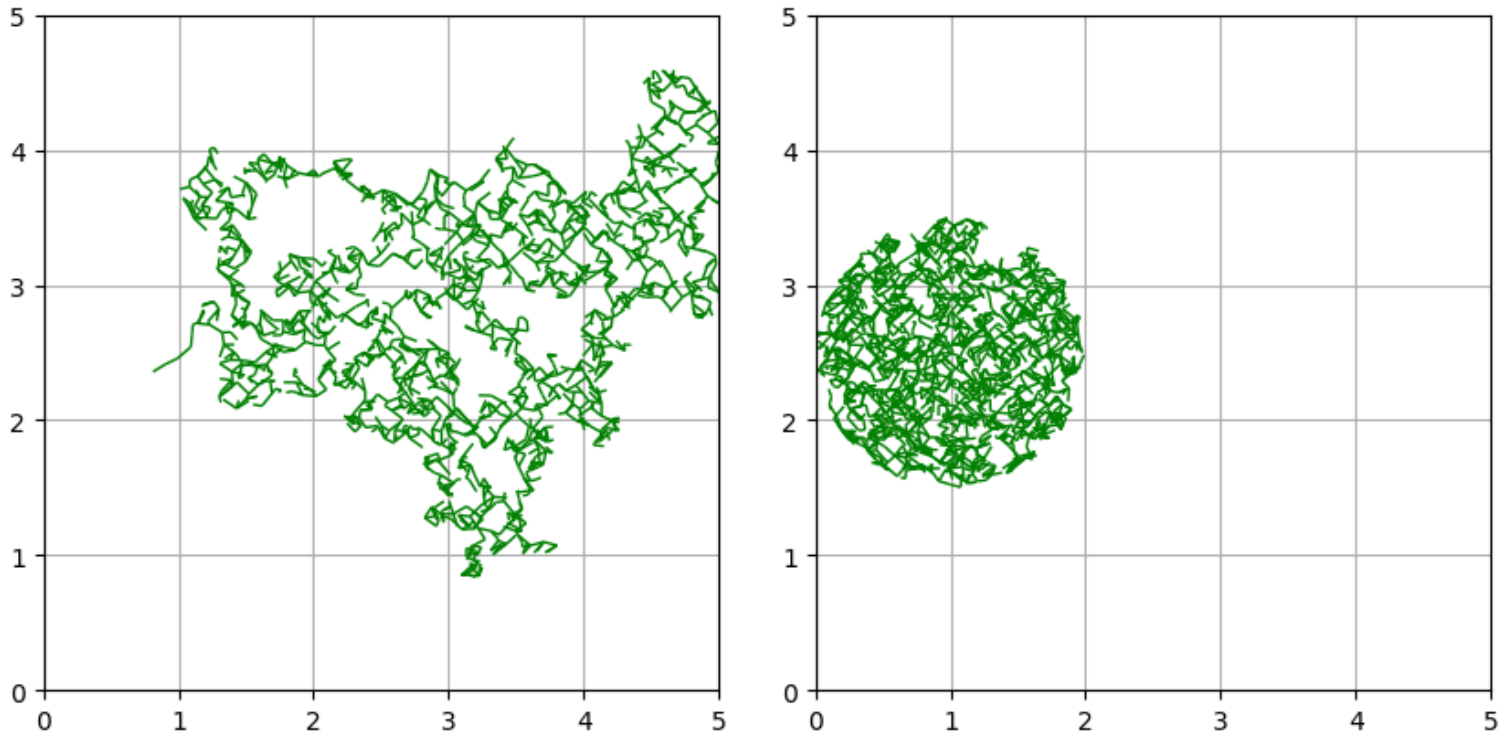
Éviter l'agglutinement avec Ex



Entrées: $s_0 \in S$, `pas`, `Cellule`
Sortie : Arbre d'exploration T

$T \leftarrow \{s_0\}$, $c_{s_0} \leftarrow 0$
while $|\text{arbre}| < \text{itérations}$ **do**
  $b = \underset{b}{\text{argmin}} \sum_{s \in T, \text{Bin}(s)=b} c_s$
  $s = \underset{s \in T, \text{Bin}(s)=b}{\text{argmin}} c_s$
  Incrémenter c_s
  $a =$action_aleatoire()
  $s' = \text{pas}(s, a)$
  $T \leftarrow T \cup \{s'\}$
  Si $c_{s'}$ n'est pas défini alors
    $c_{s'} \leftarrow 0$
end

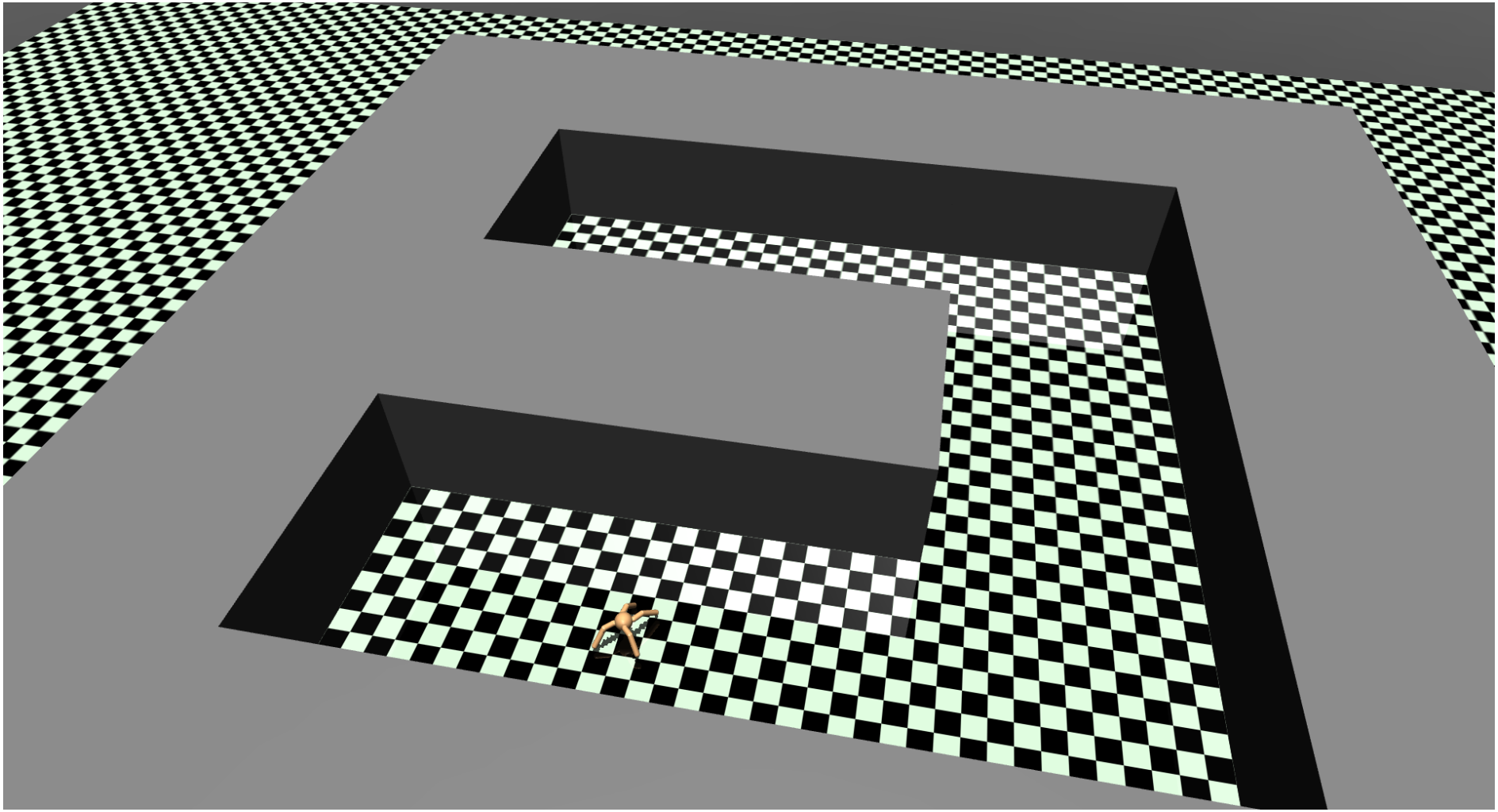
Éviter l'agglutinement avec Ex



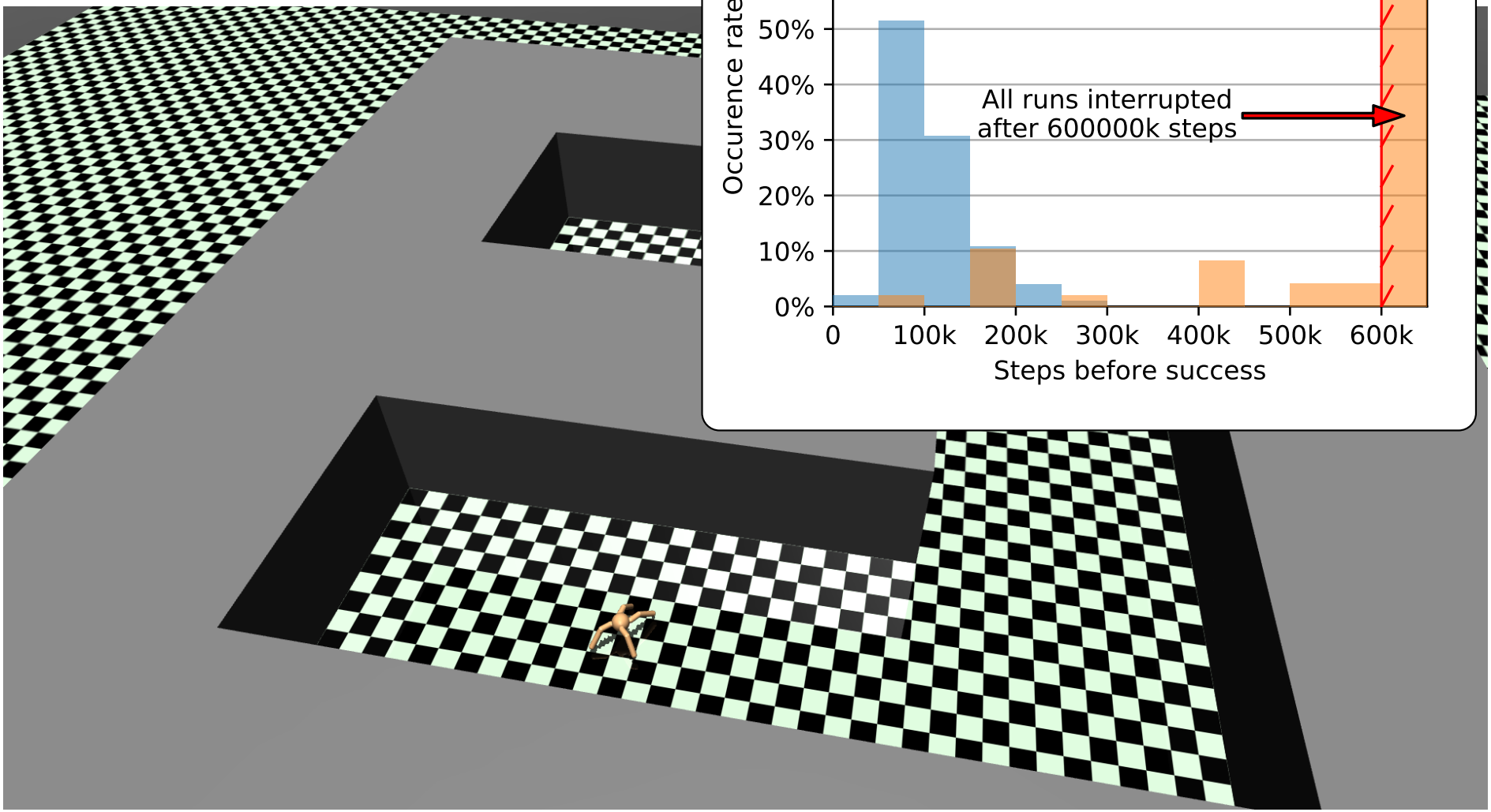
(a) (b)

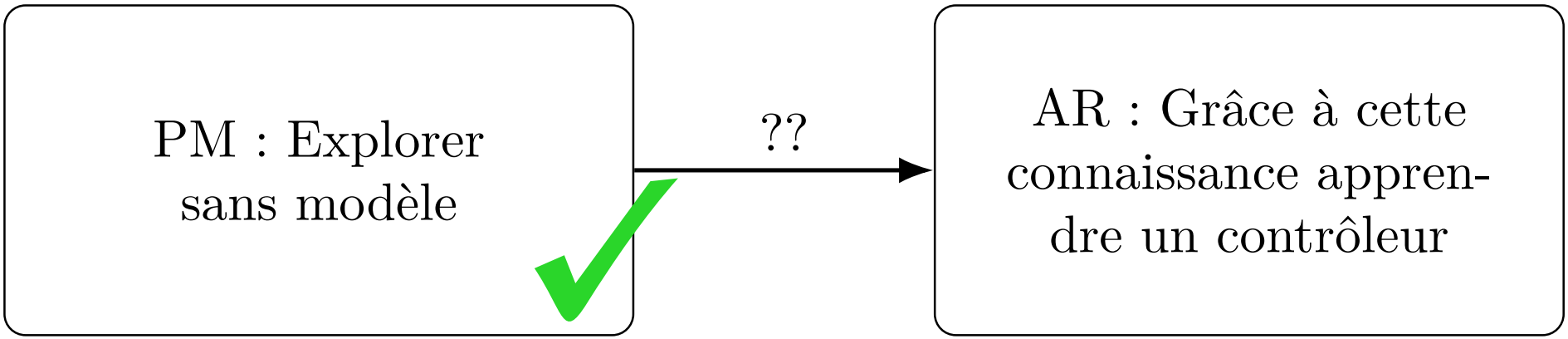
FIGURE – 2000 itérations de Ex, lorsque l'espace d'échantillonnage est (a) égal ou (b) plus grand que l'espace accessible.

Résultats sur AntMaze

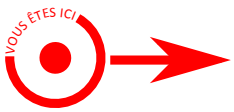


Résultats sur AntMaze



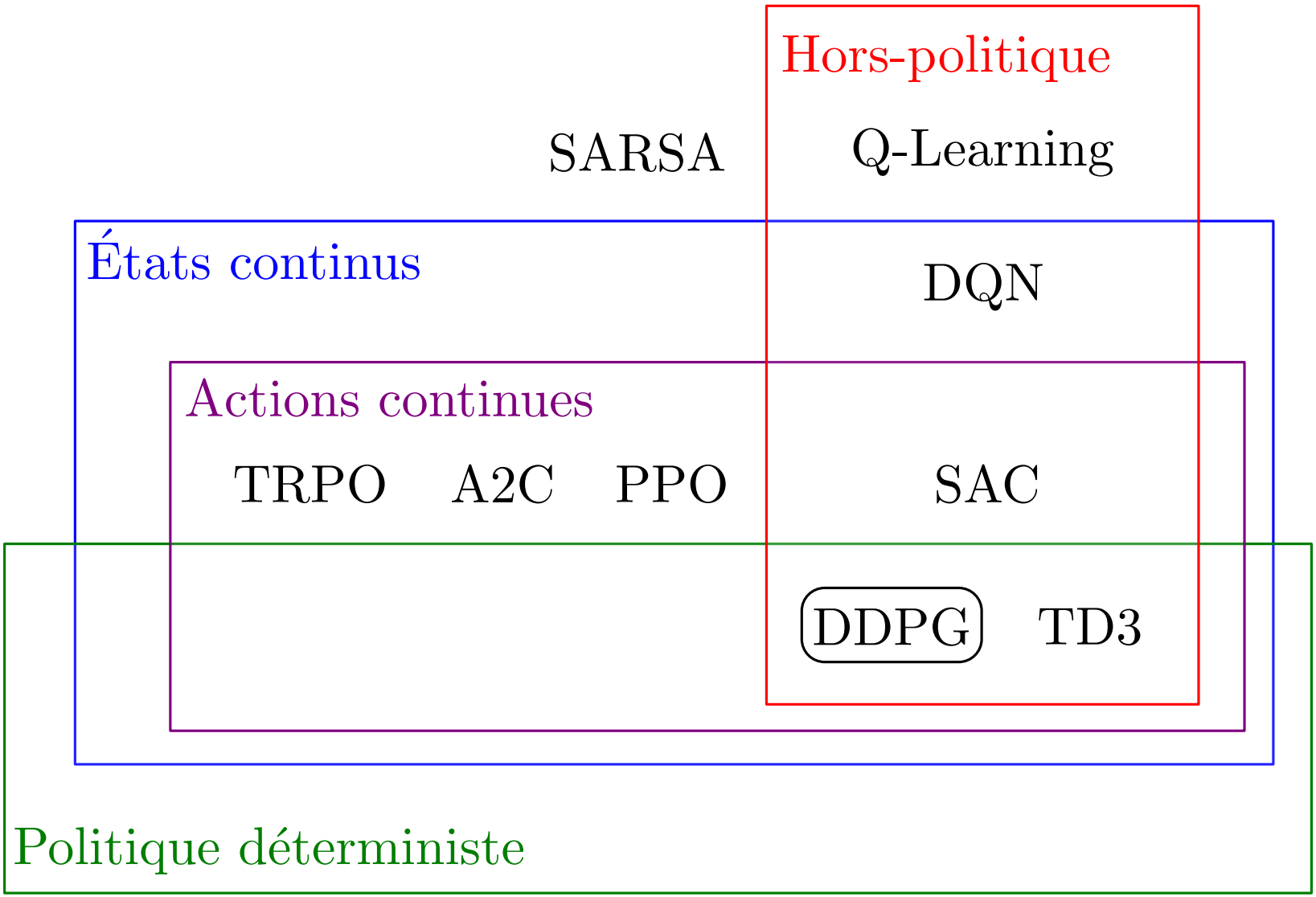

PM : Explorer sans modèle
??
AR : Grâce à cette connaissance apprendre un contrôleur

- Introduction
  - Cadre
  - Apprentissage par renforcement
  - Planification de mouvement
- Phase 1 : planification de mouvement
  - RRT, EST, Ex



- Phase 2 : apprentissage par renforcement
  - DDPG
  - Transfert d'expérience
  - Plan, Backplay
  - Cas de blocage de DDPG
  - Plan, Backplay, Chain Skills
- Conclusion

Choix d'un algorithme



Deep Q-Network (DQN) [8]

On se place dans le cas d'environnements déterministes.
On définit la fonction de valeur état-action :

$$Q^\pi(s,a) = R(s,a) + \gamma R^\pi(\mathrm{pas}(s,a)).$$

DQN maintient un réseau de neurones Q_θ qui estime Q^{π^*}.

Règle de mise à jour

Suite à une transition (s_i, a_i, r_i, s'_i),
mise à jour de $Q_\theta(s_i, a_i)$ vers $R(s_i, a_i) + \gamma \max_{a'} Q_\theta(s'_i, a')$.

Politique $\pi(s) = \mathrm{argmax}_a\, Q_\theta(s,a)$

8. Mnih et al., « Human-level control through deep reinforcement learning », 2015.

Deep Deterministic Policy Gradient (DDPG)[9]

Passage au cas continu en maintenant une estimation $\pi_\psi(s)$ de $\text{argmax}_a\, Q_\theta(s,a)$.

Règle de mise à jour

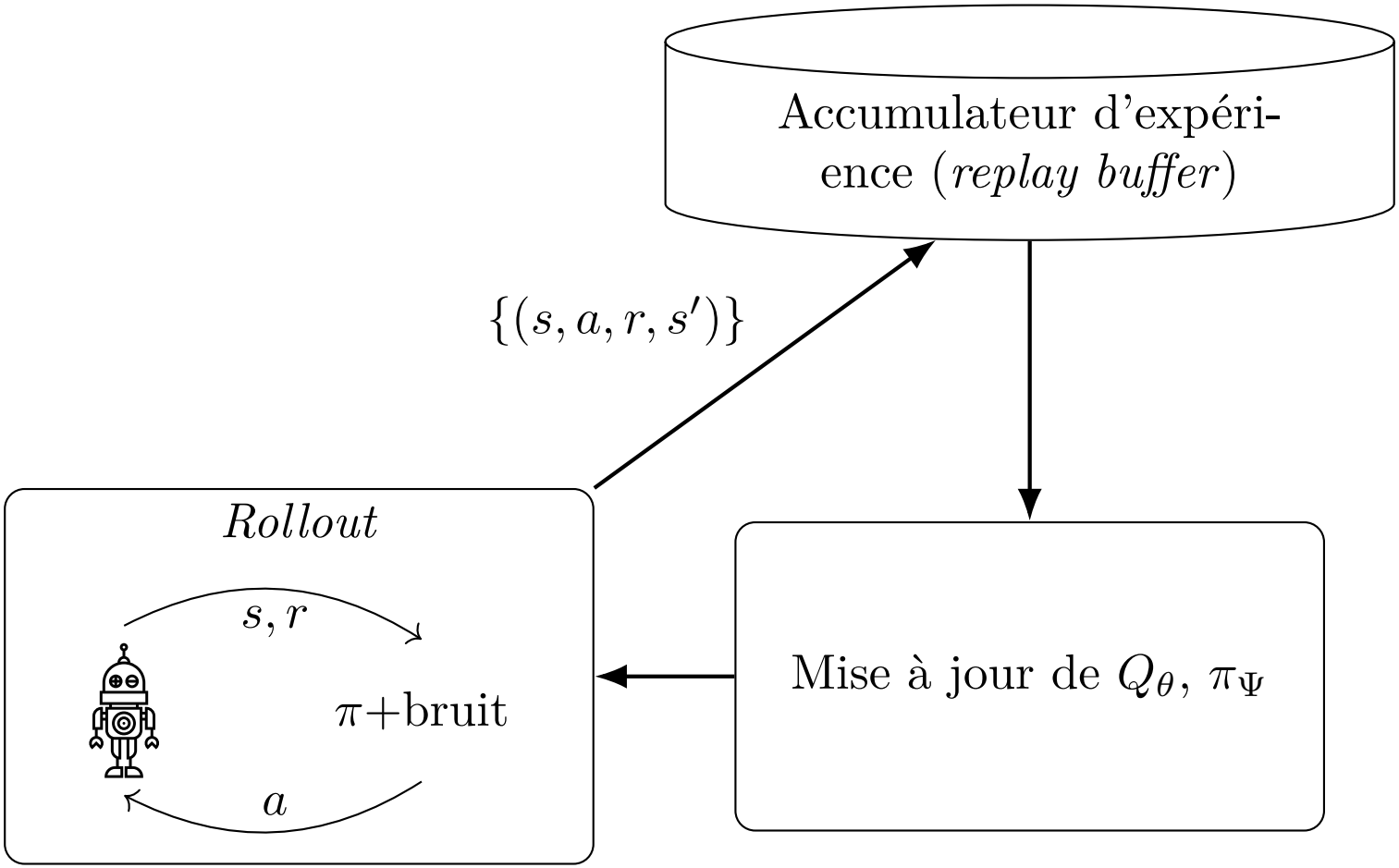
$$\begin{cases} \text{Mettre à jour } Q_\theta(s_i,a_i) \text{ vers } R(s_i,a_i)+\gamma Q_\theta(s_i',\pi_\psi(s_i')) \\ \text{Maximiser } Q_\theta(s_i,\pi_\psi(s_i)) \text{ par rapport à } \psi \end{cases}$$

Policy Gradient Theorem La mise à jour de ψ est contrôlée par

$$\nabla_a Q_\theta(s_i,a)|_{a=\pi_\psi(s_i)}$$

9. Lillicrap et al., « Continuous control with deep reinforcement learning », 2015.

DDPG



Mécanisme d'exploration : bruit ajouté aux actions durant la génération d'expérience.

DDPG dans un labyrinthe

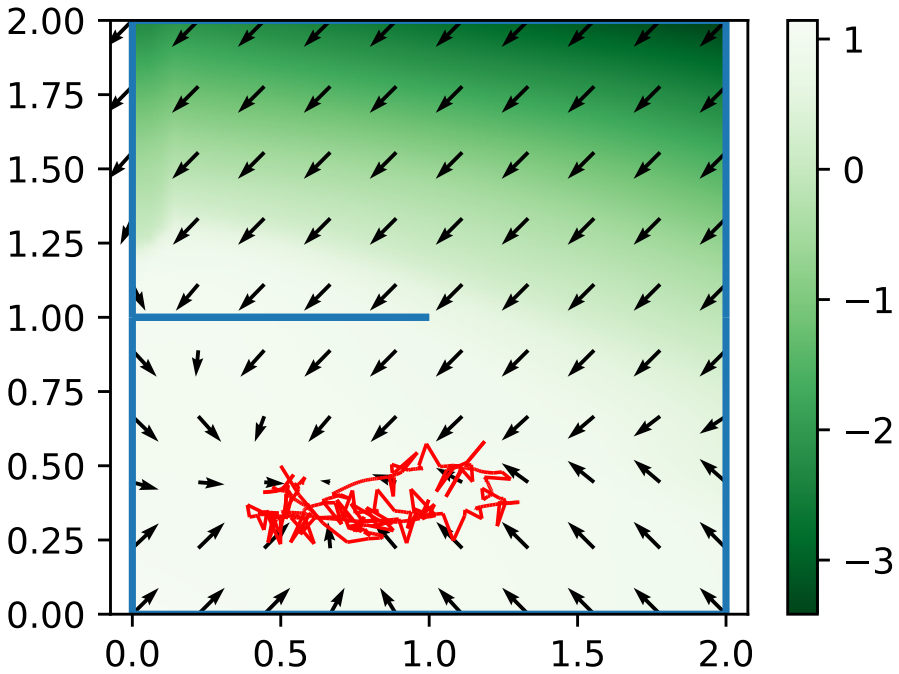
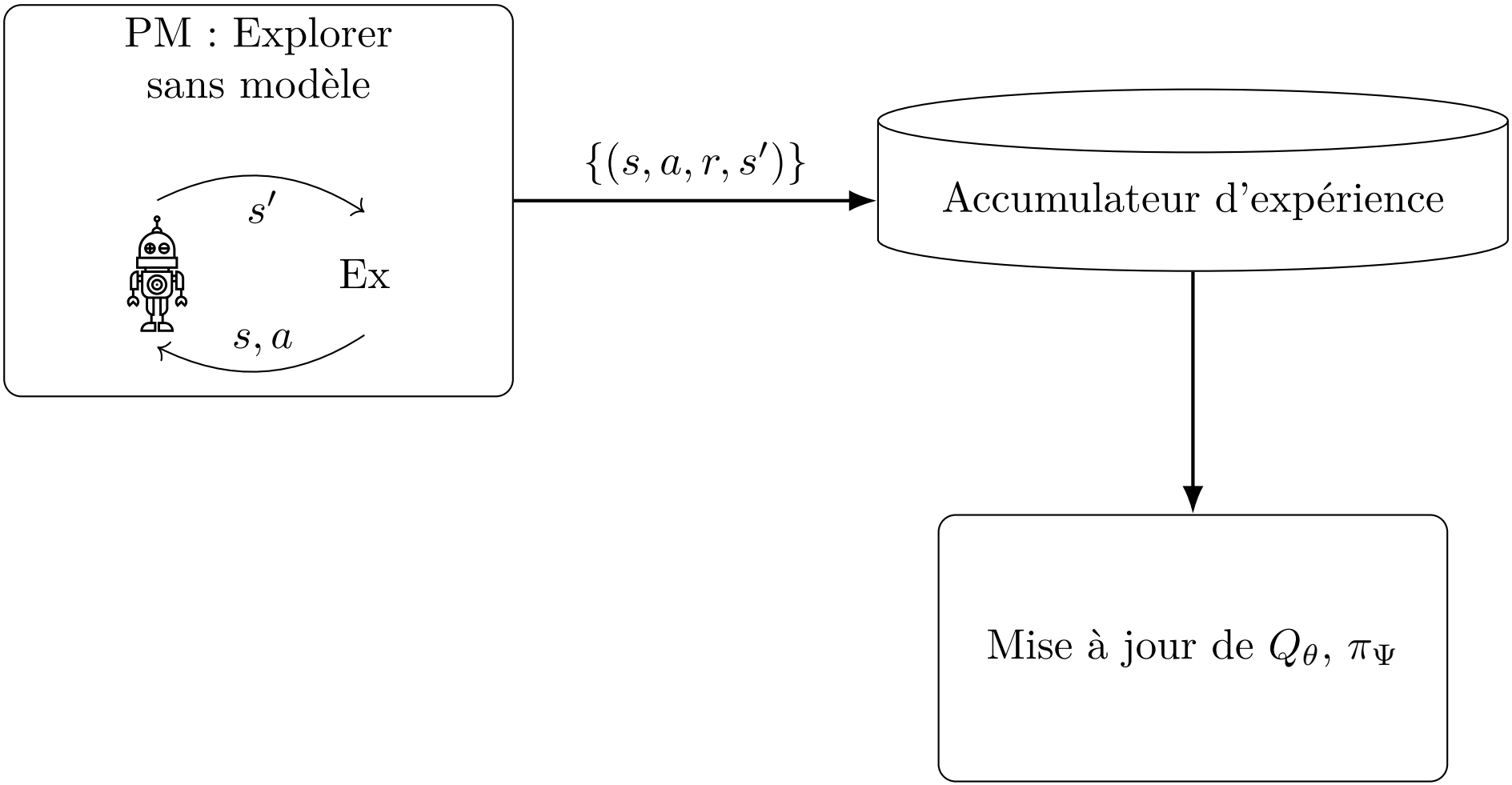


FIGURE – DDPG après 10k pas. En vert $Q_\theta(s, \pi_\psi(s))$, en noir $\pi_\psi(s)$. Les murs ont une récompense négative, une récompense positive est présente en haut à gauche, et partout ailleurs la récompense est nulle.

$\rightarrow$DDPG explore mal

DDPG hors-ligne



DDPG hors-ligne dans un labyrinthe

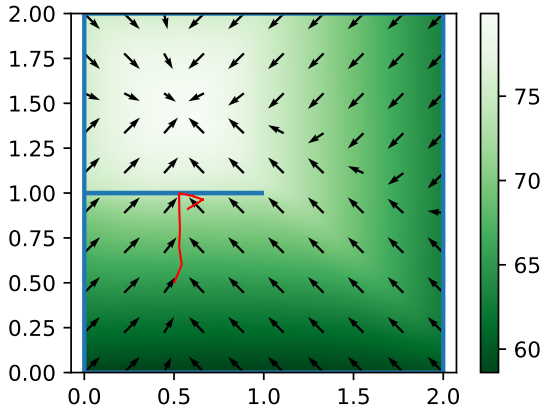


FIGURE – Après convergence de π_ψ et Q_θ, avec 10k transitions d'exploration.

DDPG hors-ligne dans un labyrinthe

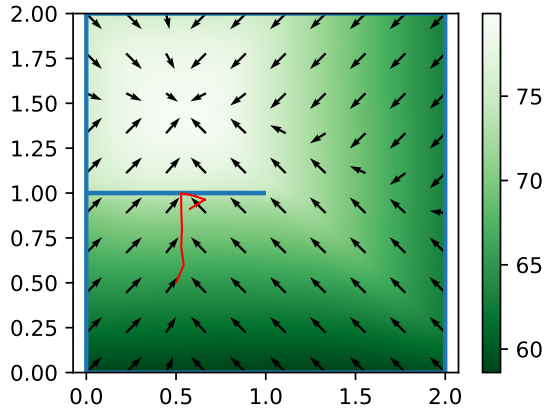


FIGURE – Après convergence de π_ψ et Q_θ, avec 10k transitions d'exploration.

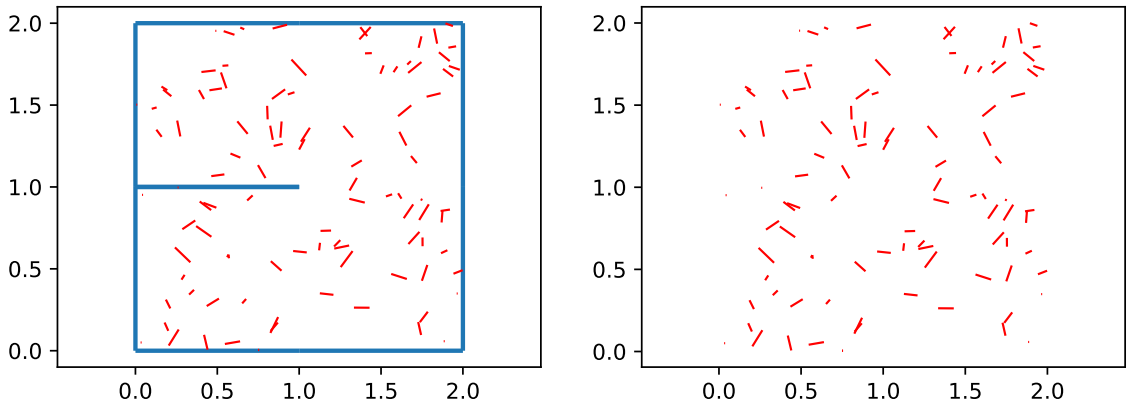
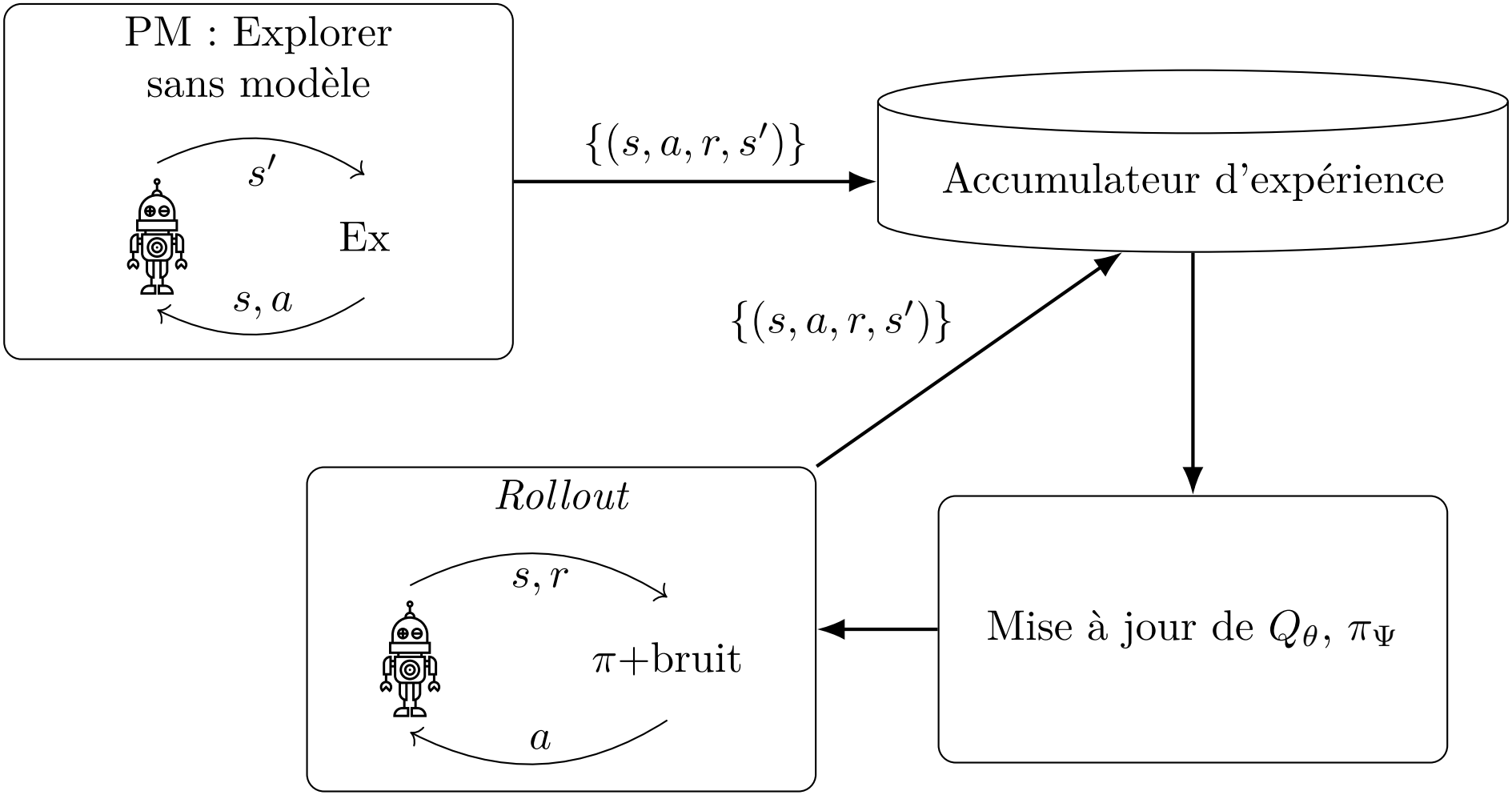
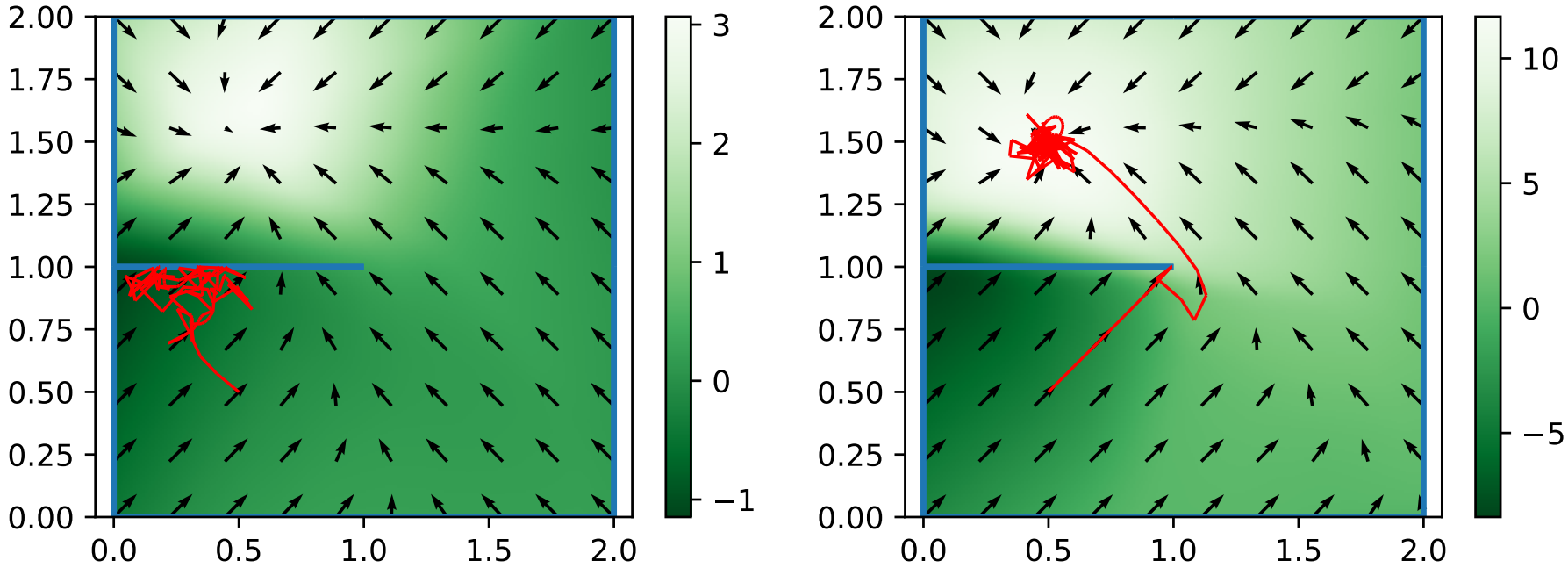


FIGURE – Échantillon de 100 transitions de l'accumulateur d'expérience.

DDPG amorcé


PM : Explorer
sans modèle
s'
Ex
s, a
$\{(s, a, r, s')\}$
Accumulateur d'expérience
$\{(s, a, r, s')\}$
Rollout
s, r
π+bruit
a
Mise à jour de Q_θ, π_Ψ

DDPG amorcé dans un labyrinthe



(a) (b)

FIGURE – DDPG amorcé après 10k pas d'exploration et (a) 1000 puis (b) 2000 pas d'apprentissage.

Mais ça ne fonctionne plus dès qu'on augmente la difficulté

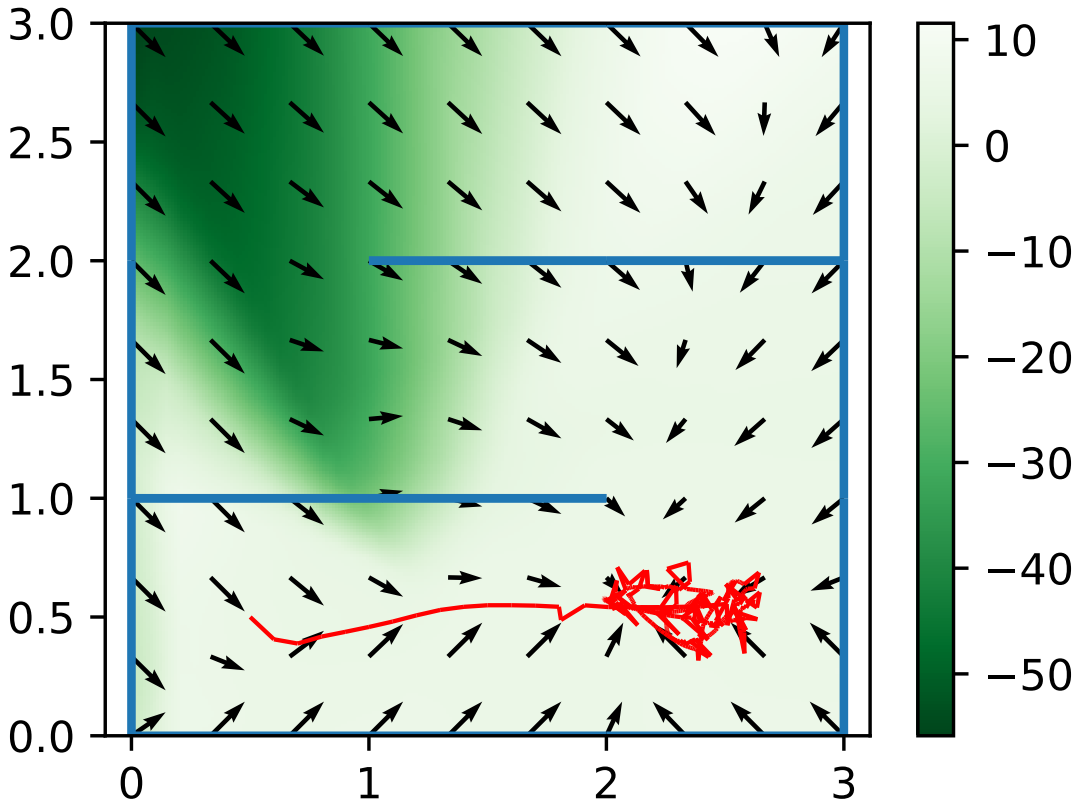
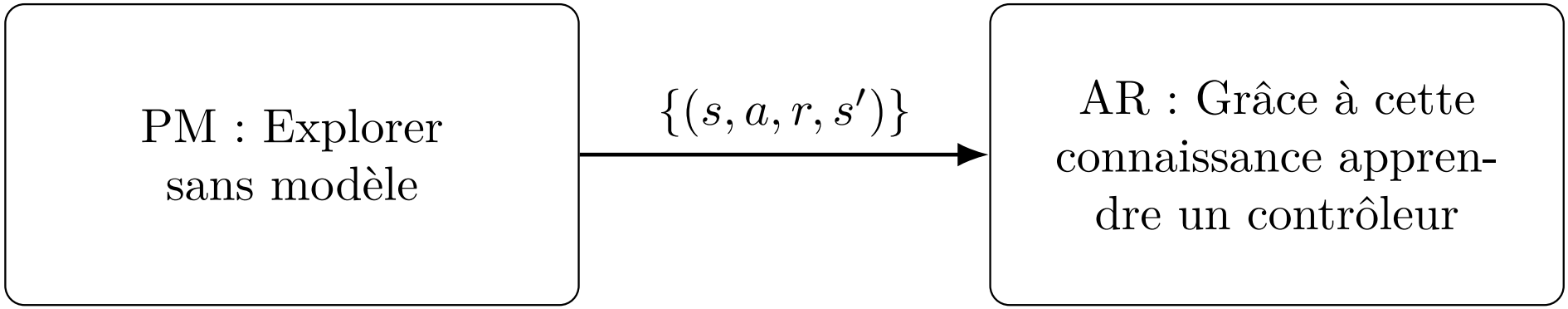
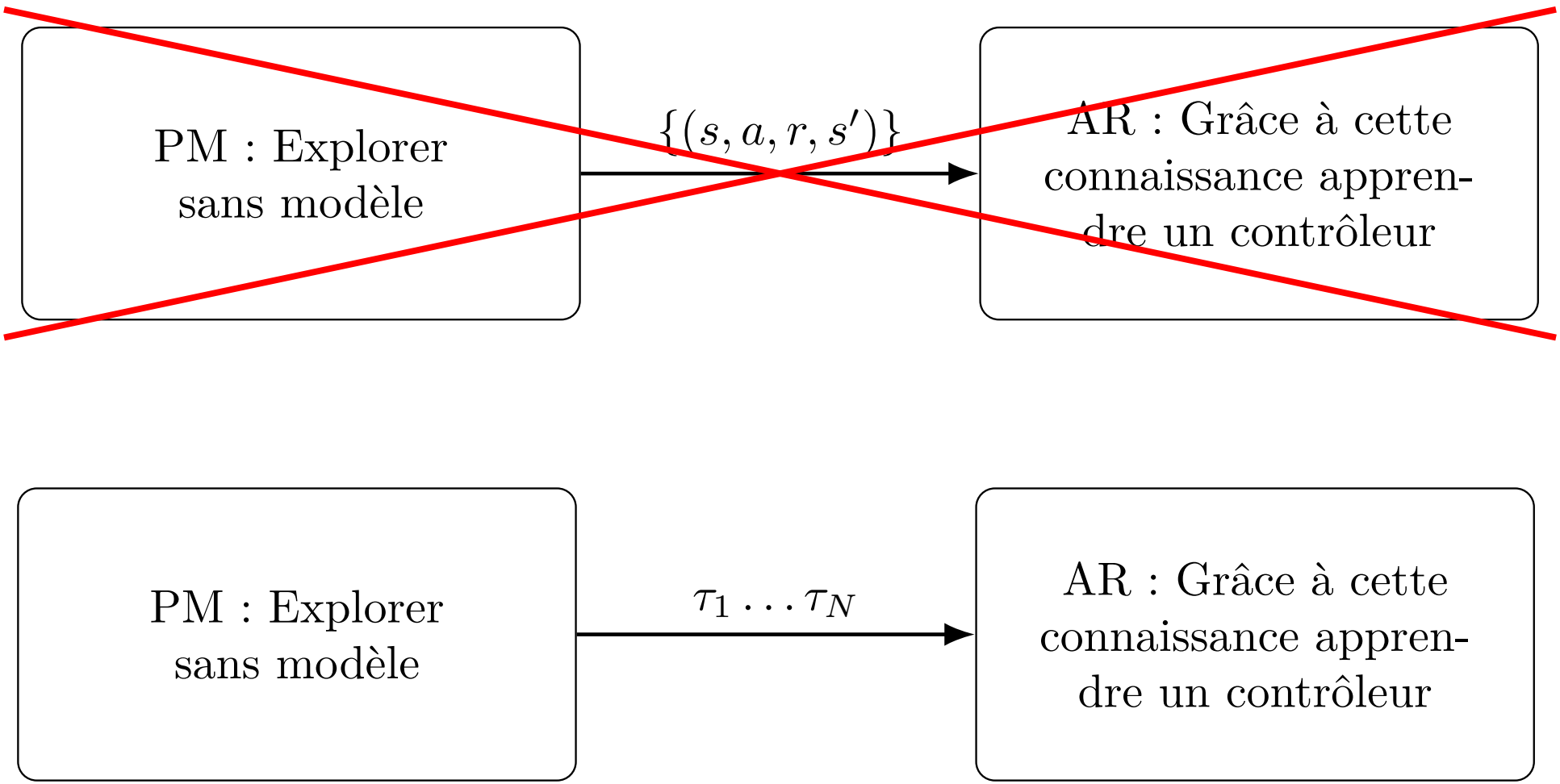
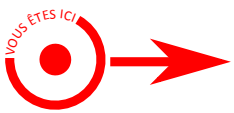


FIGURE – DDPG amorcé après 20k pas d'exploration et 20k pas d'apprentissage.

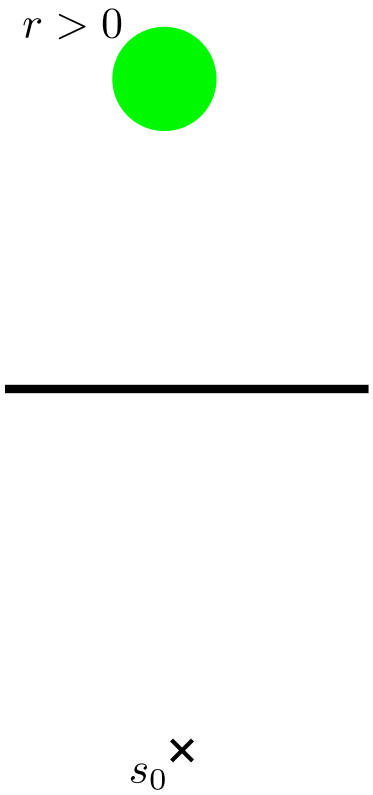

PM : Explorer sans modèle
$\{(s, a, r, s')\}$
AR : Grâce à cette connaissance apprendre un contrôleur

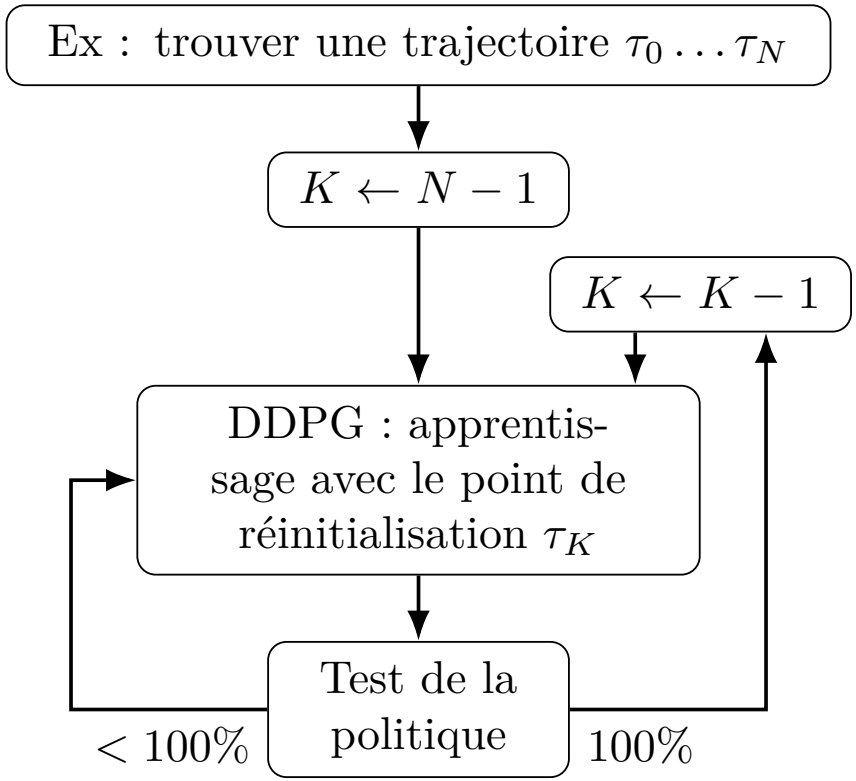
Transfert de connaissances via une trajectoire





PB : *Plan, Backplay*

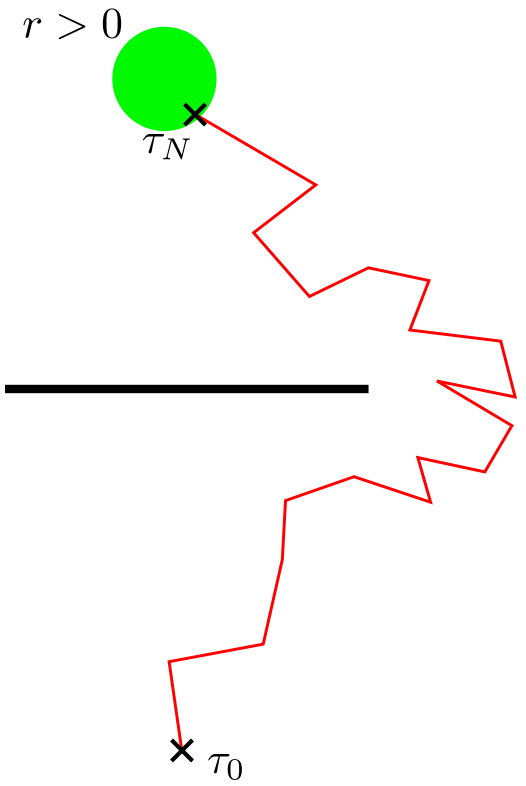


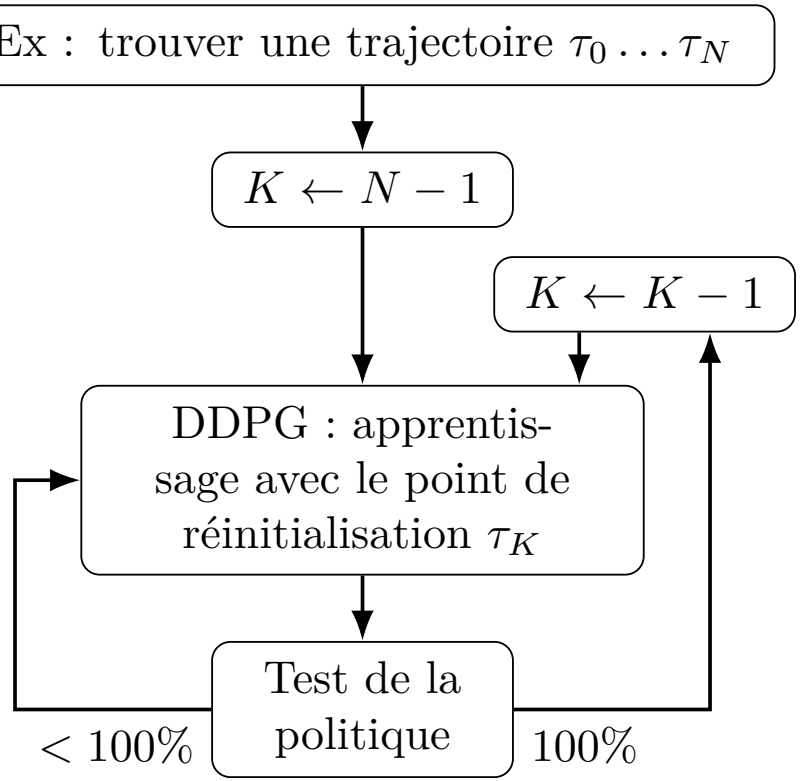


Version en boucle fermée de Backplay [10].
Technique exploitée en actions finies par Go-Explore [11].

10. Resnick et al., « Backplay », 2018.
11. Ecoffet et al., « Go-explore », 2019.

PB : *Plan, Backplay*



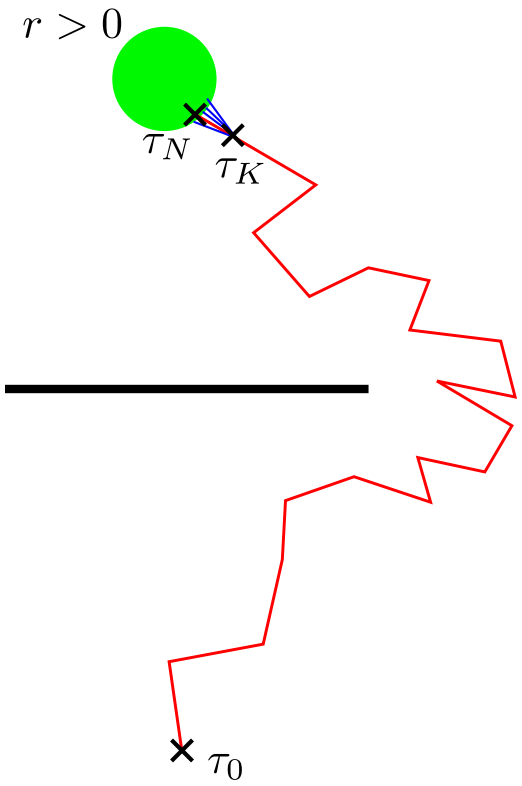


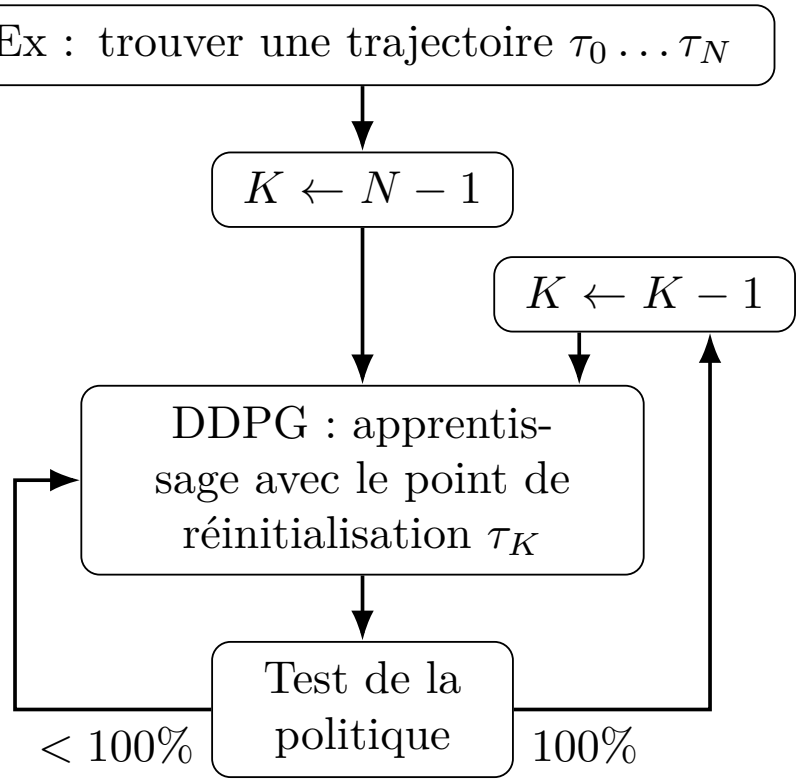
Version en boucle fermée de Backplay [10].

Technique exploitée en actions finies par Go-Explore [11].

10. Resnick et al., « Backplay », 2018.
11. Ecoffet et al., « Go-explore », 2019.

PB : *Plan, Backplay*



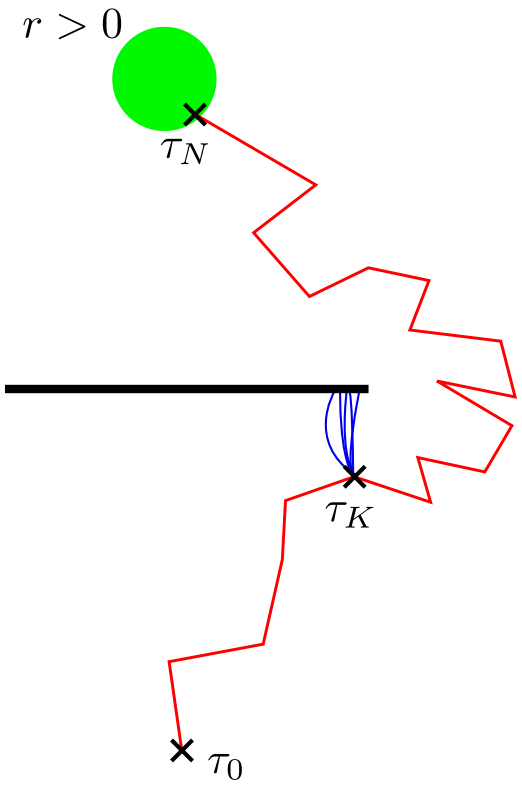


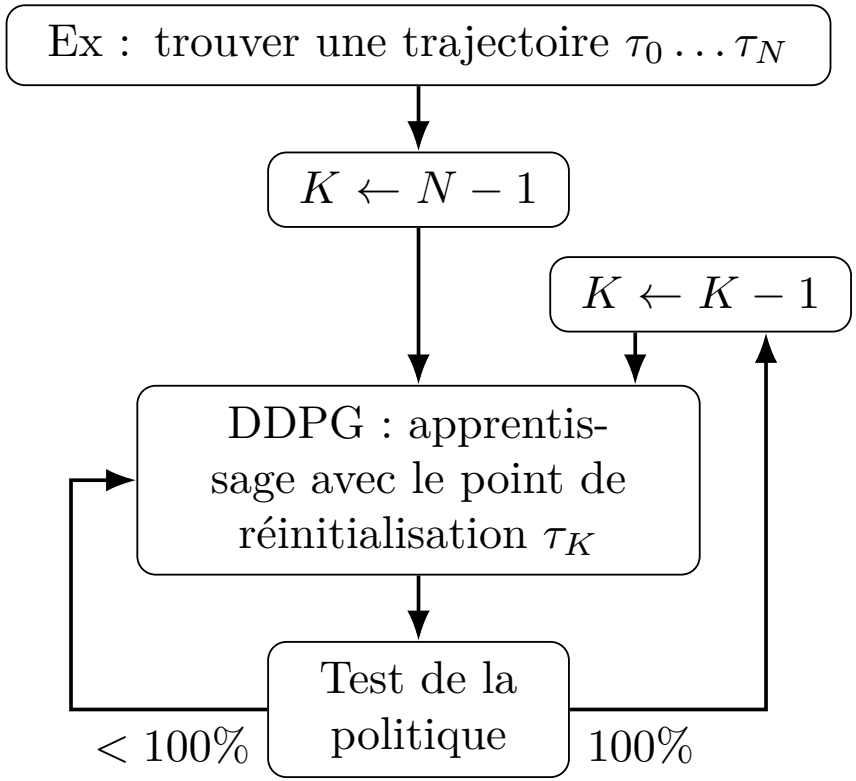
Version en boucle fermée de Backplay [10].

Technique exploitée en actions finies par Go-Explore [11].

10. Resnick et al., « Backplay », 2018.
11. Ecoffet et al., « Go-explore », 2019.

PB : *Plan, Backplay*



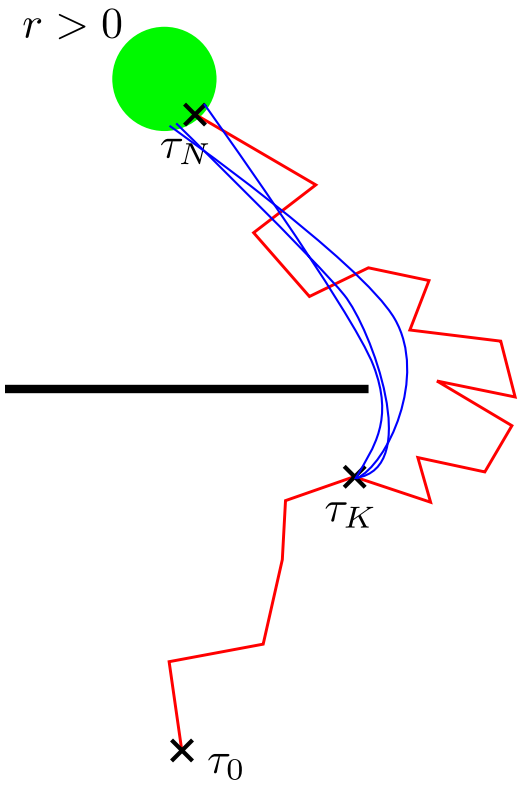


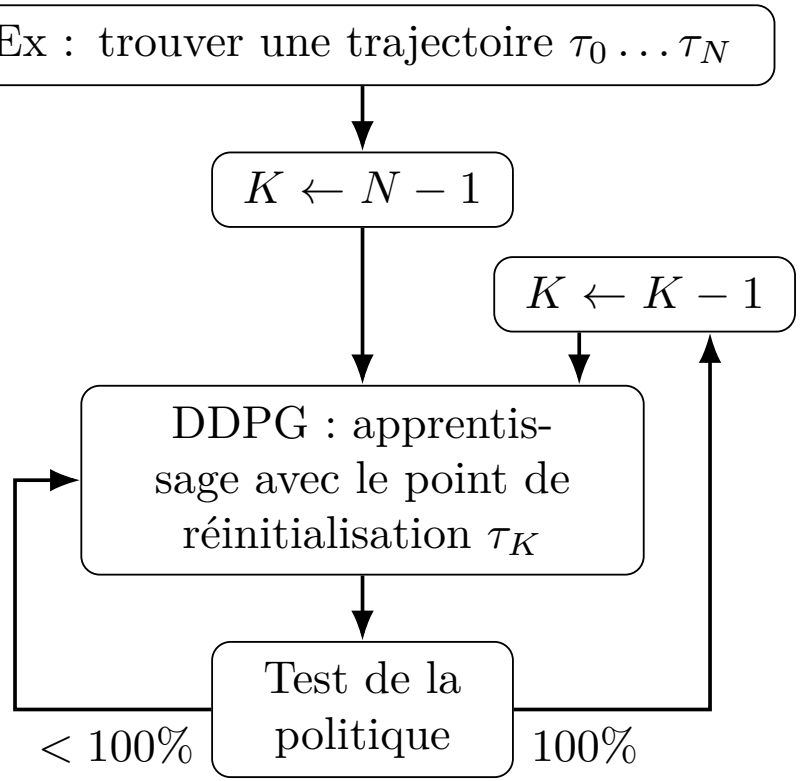
Version en boucle fermée de Backplay [10].

Technique exploitée en actions finies par Go-Explore [11].

10. Resnick et al., « Backplay », 2018.
11. Ecoffet et al., « Go-explore », 2019.

PB : *Plan, Backplay*





Version en boucle fermée de Backplay [10].

Technique exploitée en actions finies par Go-Explore [11].

10. Resnick et al., « Backplay », 2018.
11. Ecoffet et al., « Go-explore », 2019.

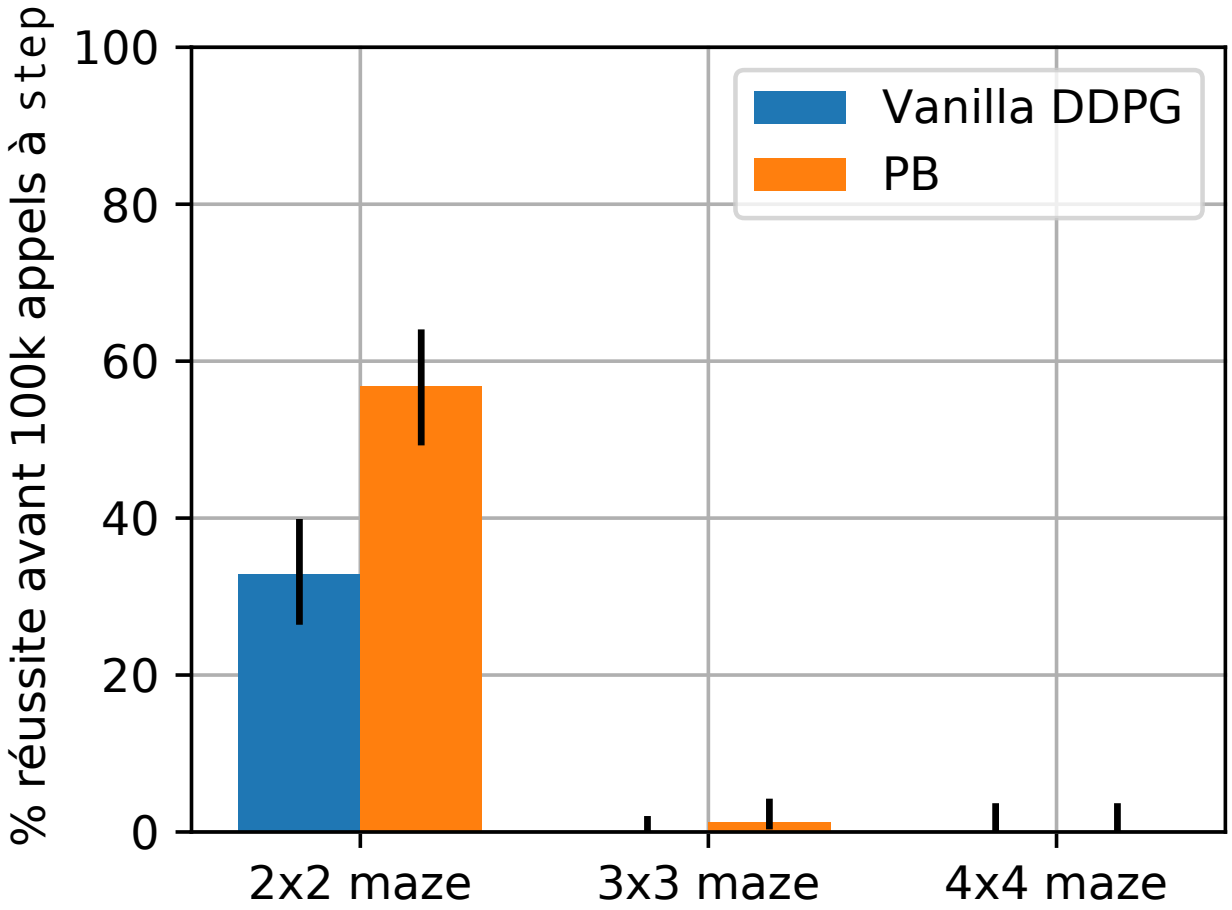
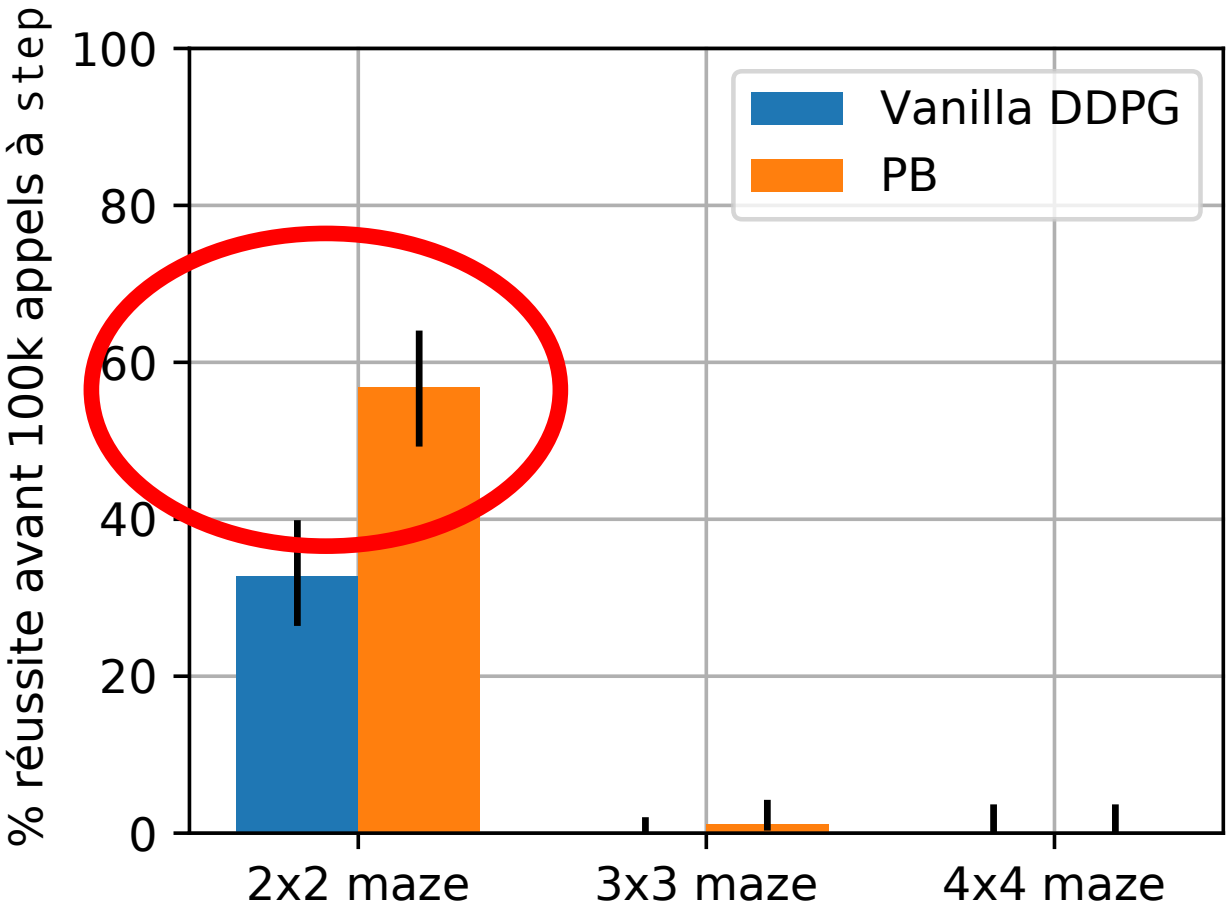
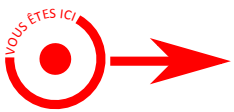


FIGURE – Taux de succès de PB sur des labyrinthes 2D, comparé à DDPG.

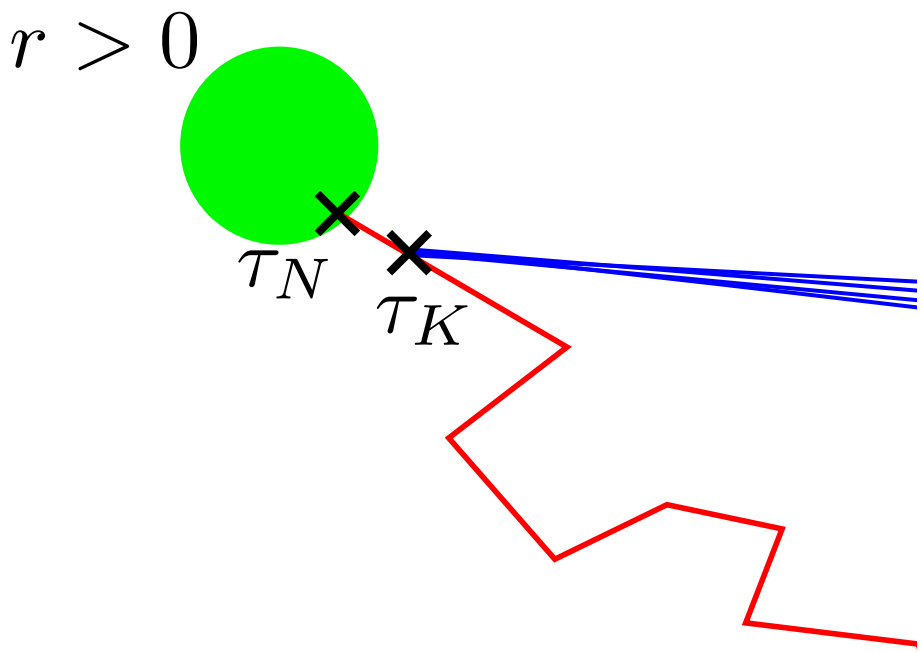
Barres d'erreur calculées avec la méthode du score de Wilson. Les labyrinthes 3×3 sont de forme fixe en forme de $\ll$ S $\gg$. Les labyrinthes 4×4 sont générés à la volée pour chaque échantillon. De gauche à droite, $N = 183, 169, 185, 168, 101, 101$.

Pourquoi si mauvais ?



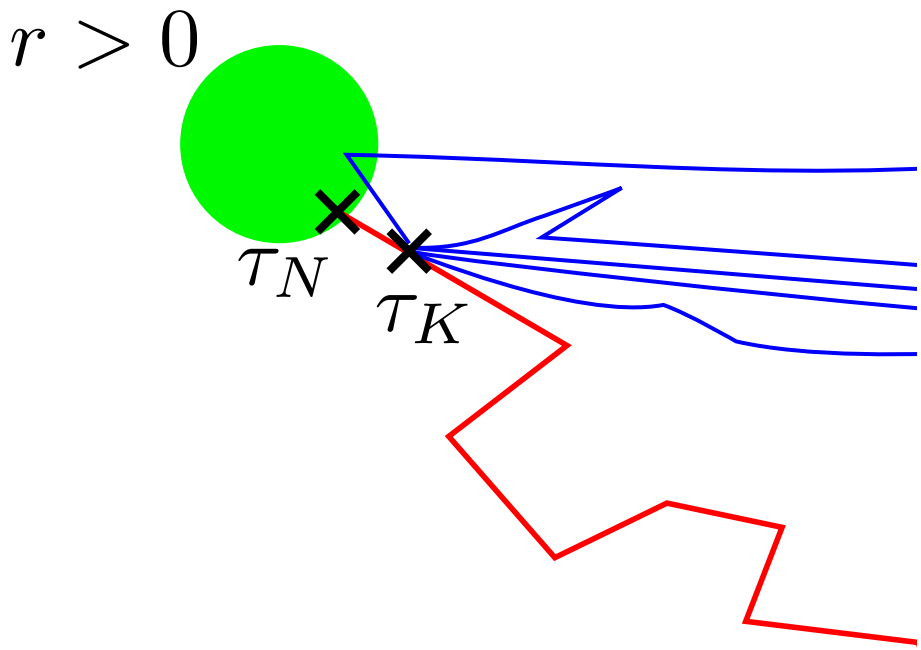


Le problème de gradient de DDPG



Convergence prématurée vers un optimum local (bruit OU)

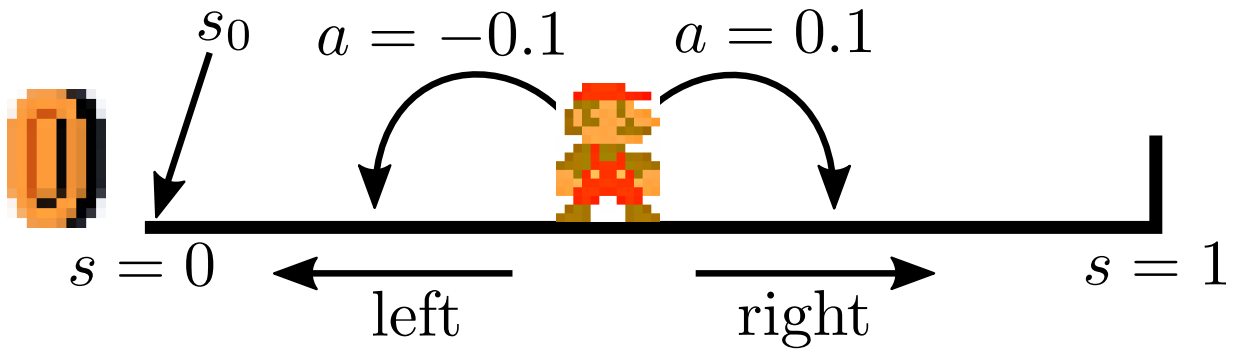
Le problème de gradient de DDPG

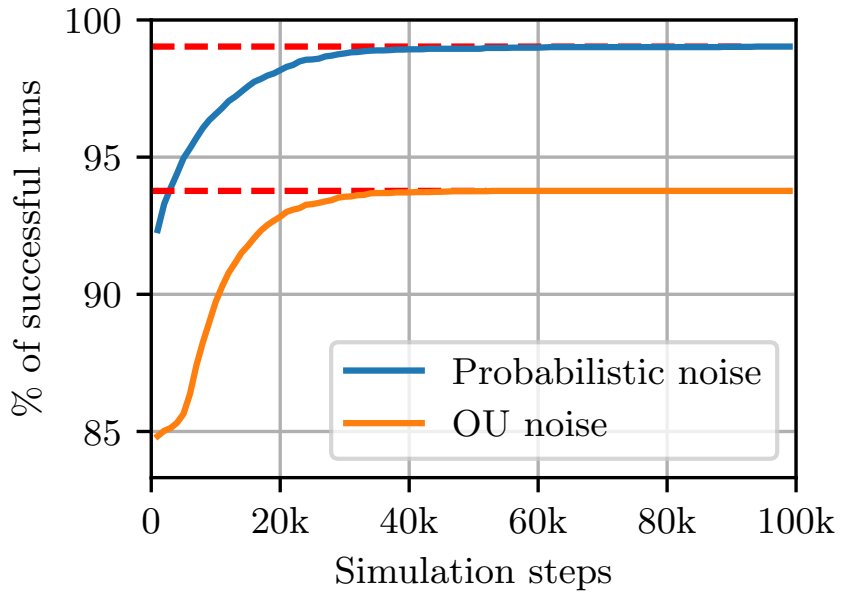


Convergence prématurée vers un optimum local malgré des expériences contraires (bruit ϵ-glouton)

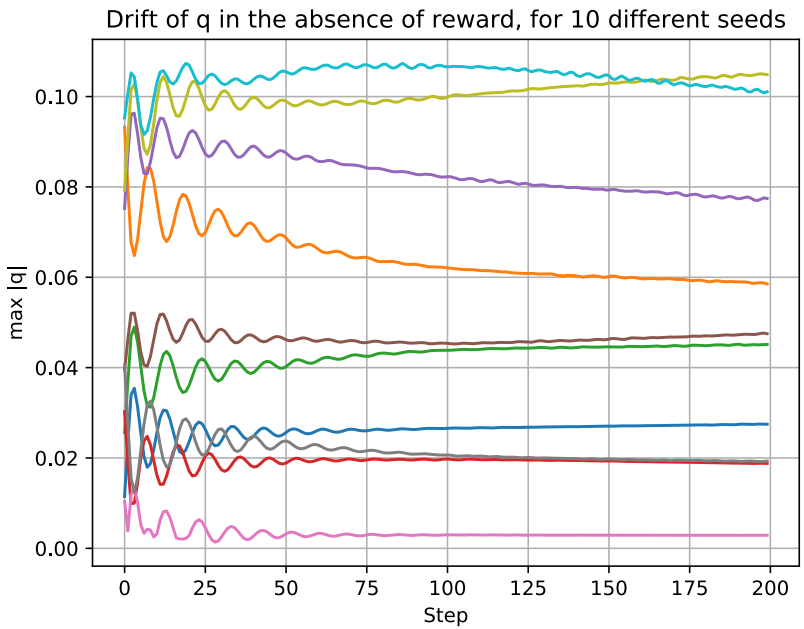
On simplifie encore...

Environnement 1D-Toy

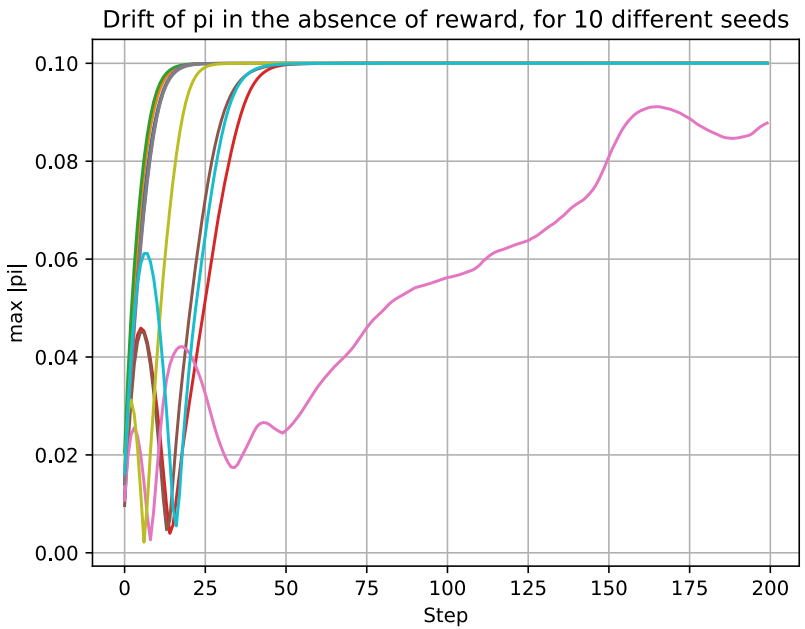




Une convergence prématurée



(a)



(b)

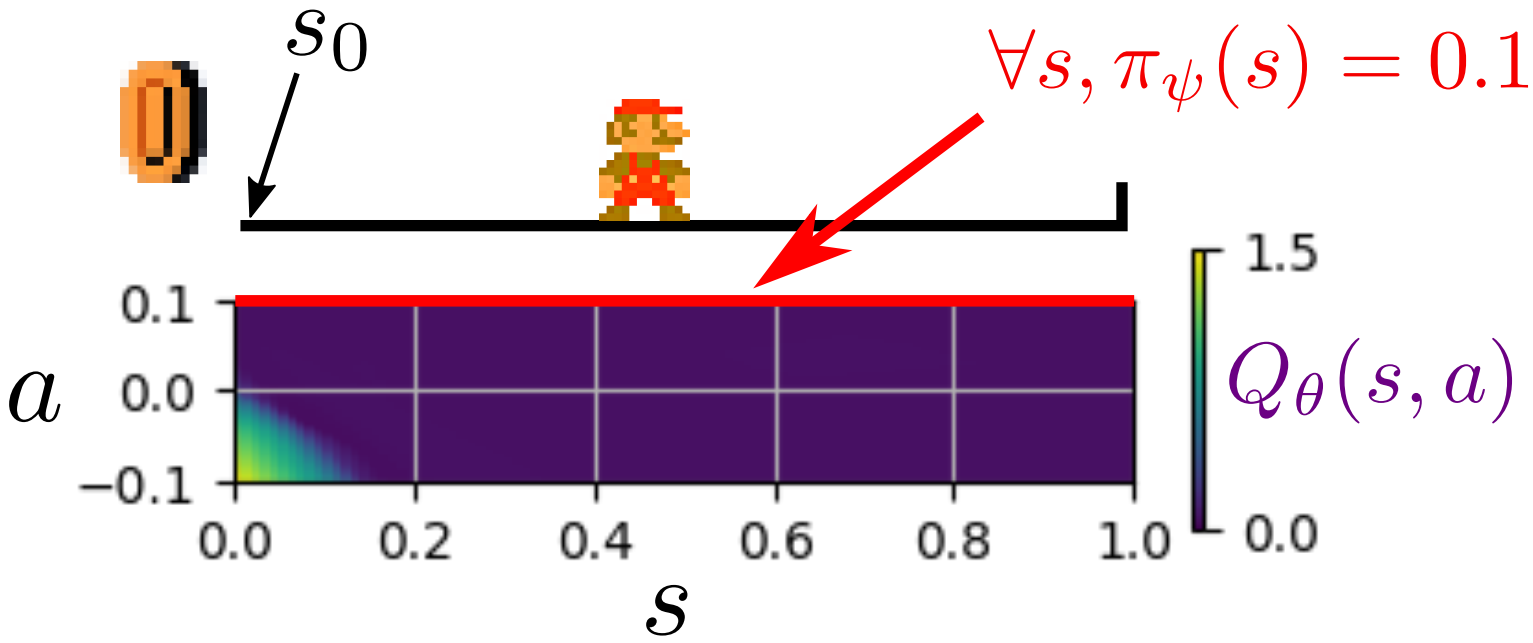
FIGURE – Convergence prématurée (a) du critique et (b) de l'acteur sur 1D-Toy avant la découverte de récompense.

Un épisode dure 50 pas de temps $\rightarrow$ souvent l'acteur est saturé avant la fin du premier épisode.

Une convergence prématurée... qui entraîne un gradient nul

Maximiser $Q_\theta(s_i, \pi_\psi(s_i))$ par rapport à ψ

$Q_\theta(s_i, a)$ est dérivé en $a = \pi_\psi(s_i)$ donc en $a = 0.1$.



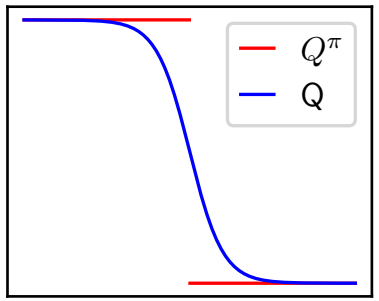
Un cas plus général

Hypothèses :

1. π_ψ est constant (mis à jour plus lentement que Q_θ).
2. $R(s, a)$ prend un nombre fini de valeurs, toutes positives.
3. Si $R(s, a) > 0$ alors l'état s est terminal (récompenses éparses).

Q_θ converge vers Q^{π_ψ}.[12]

Or $Q^{\pi_\psi} = \begin{cases} \gamma^N R(s_N, a_N) & \text{si récompense après } N \text{ pas} \\ 0 & \text{sinon} \end{cases}$.

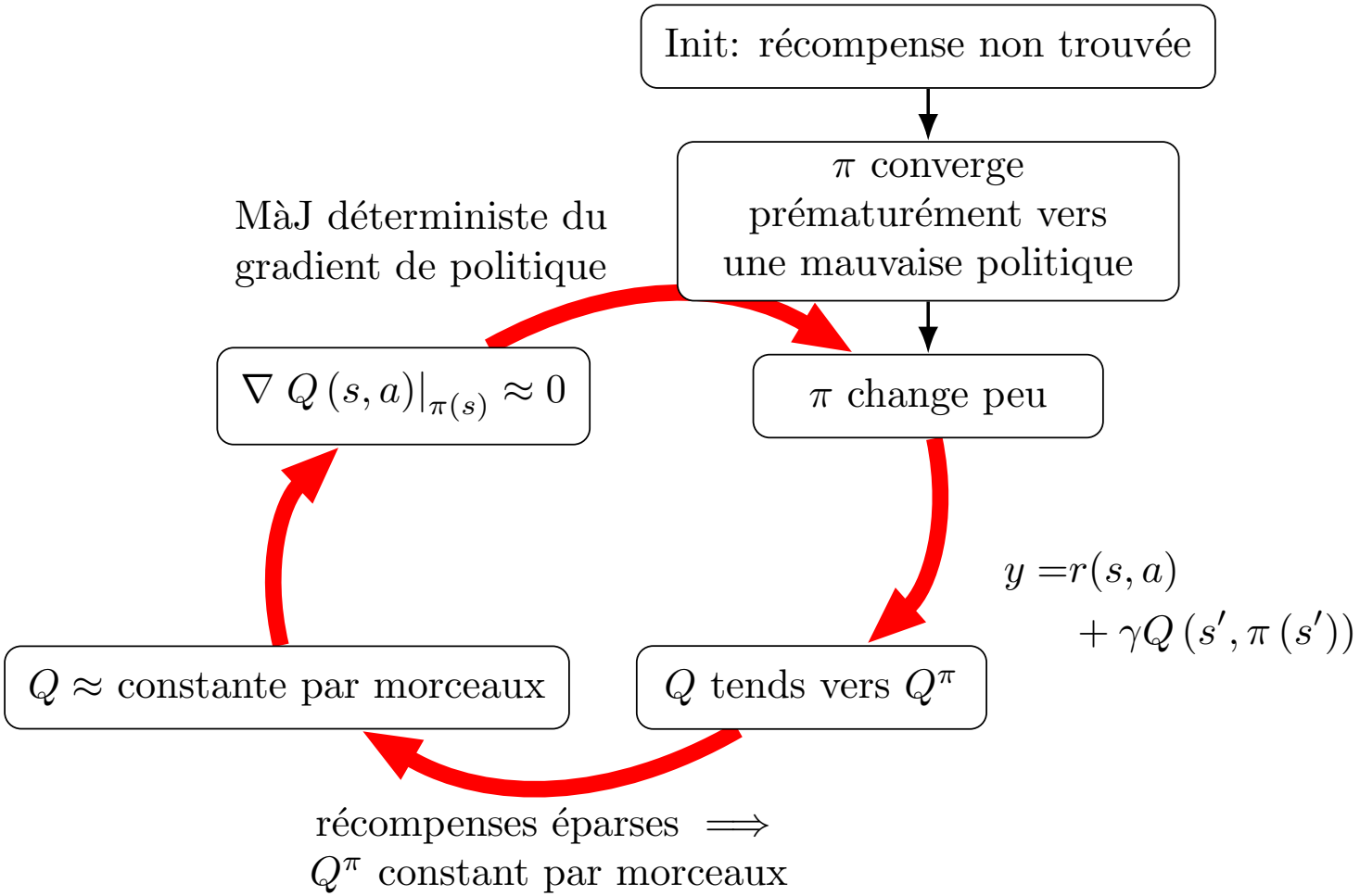


Donc Q_θ tend à avoir une dérivée nulle partout (constante par morceaux).

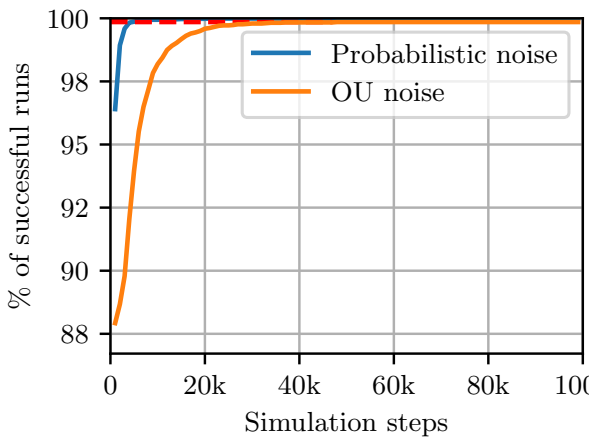
L'approximation de fonction limite ce phénomène.

12. Matheron, Perrin et Sigaud, « Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards », 2020.

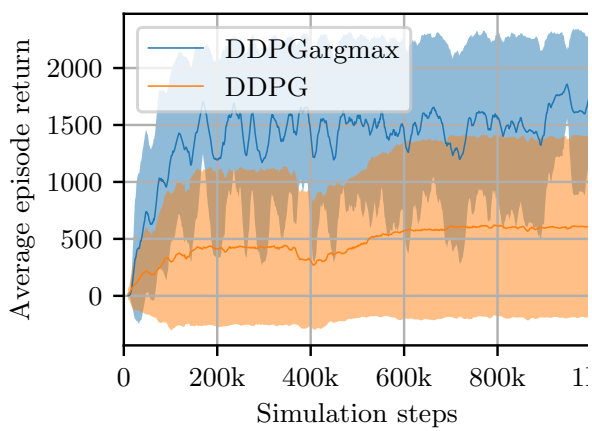
Récapitulatif



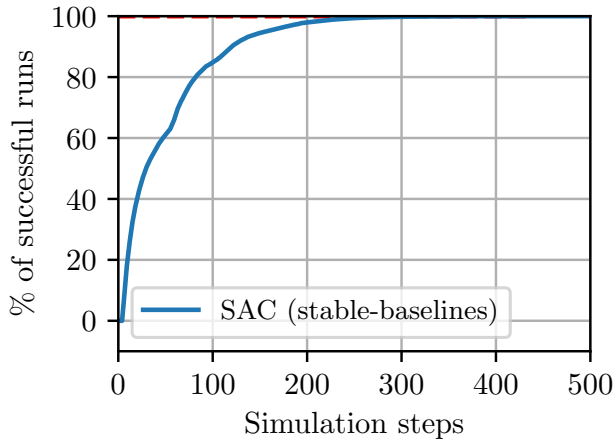
Solutions ?



(a)



(b)

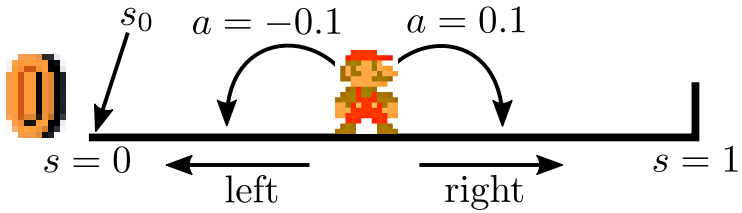


(c)

FIGURE – (a) DDPG-argmax sur 1D-Toy (argmax explicite avec sampling).
(c) DDPG et DDPG-argmax sur une version à récompenses éparses de HalfCheetah (robot simulé avec MuJoCo).
(b) SAC (acteur stochastique).

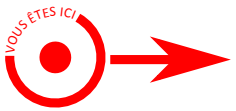
D'autres problèmes de DDPG

- Sensibilité à la distribution dans l'accumulateur [13].
- Oubli catastrophique [14].
- Sur-estimation des Q-valeurs [15].
- Fuites de Q-valeur à travers des murs fins [16].
- ≪ Deadly Triad ≫ de l'apprentissage par renforcement [17].

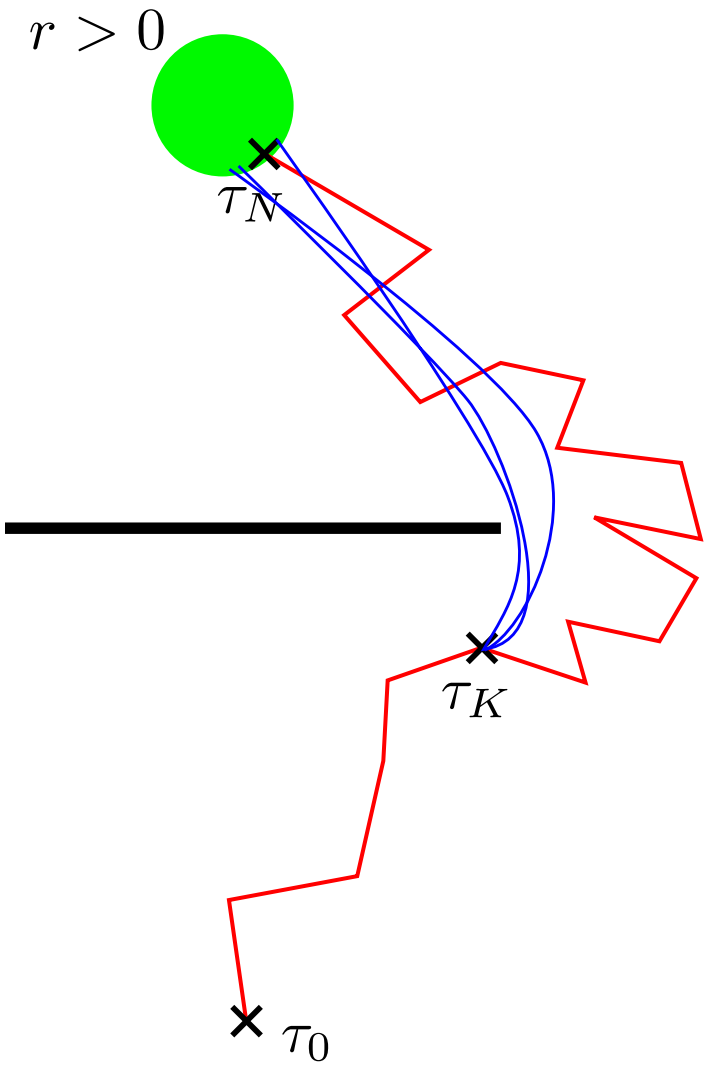


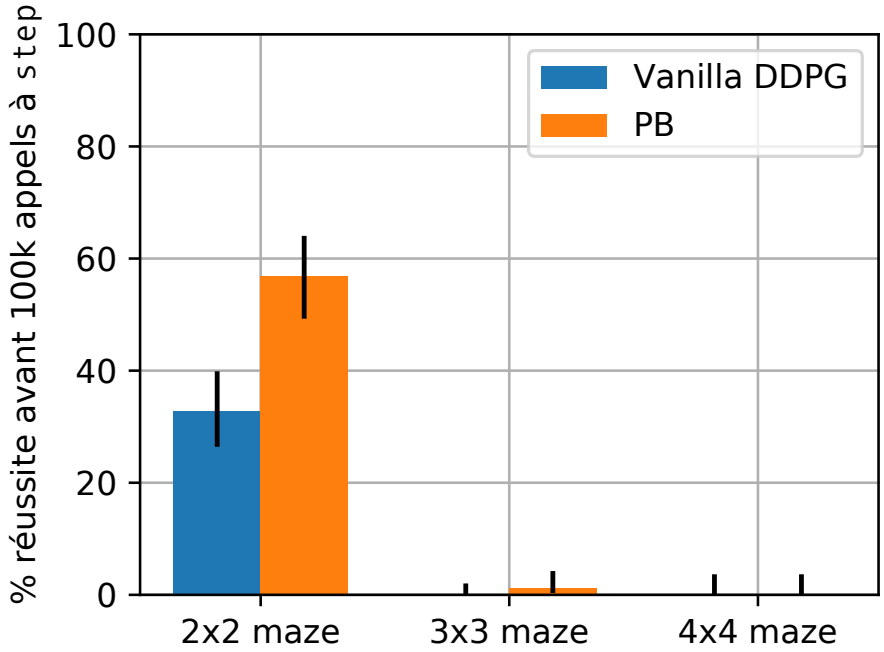
13. Schaul et al., ≪ Prioritized Experience Replay ≫, 2015.
14. Titsias et al., ≪ Functional Regularisation for Continual Learning with Gaussian Processes ≫, 2020.
15. Ciosek et al., ≪ Better Exploration with Optimistic Actor-Critic ≫, 2019.
16. Penedones et al., ≪ Adaptive Temporal-Difference Learning for Policy Evaluation with Per-State Uncertainty Estimates ≫, 2019.
17. van Hasselt et al., ≪ Deep reinforcement learning and the deadly triad ≫, 2018.

- Introduction
  - Cadre
  - Apprentissage par renforcement
  - Planification de mouvement
- Phase 1 : planification de mouvement
  - RRT, EST, Ex
- Phase 2 : apprentissage par renforcement
  - DDPG
  - Transfert d'expérience
  - Plan, Backplay
  - Cas de blocage de DDPG
  - Plan, Backplay, Chain Skills
- Conclusion

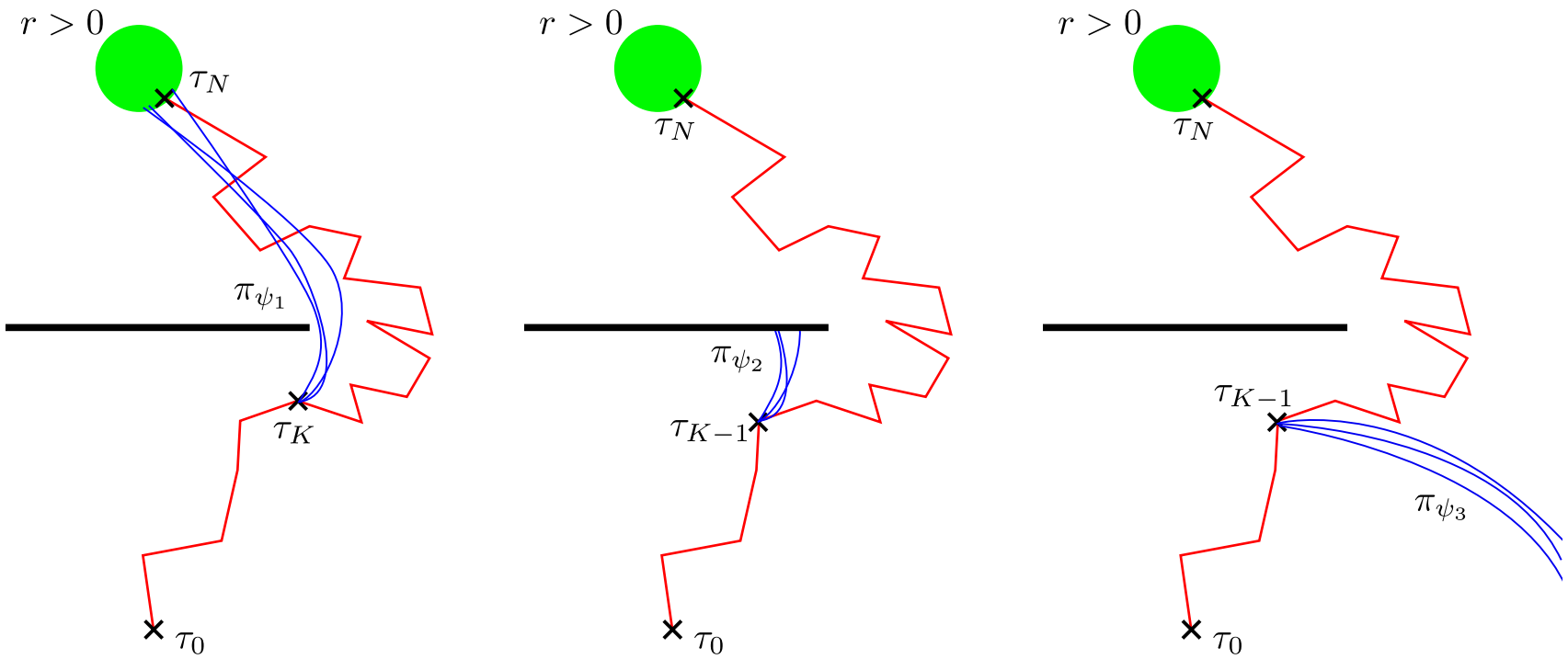


Rappel : *Plan, Backplay*



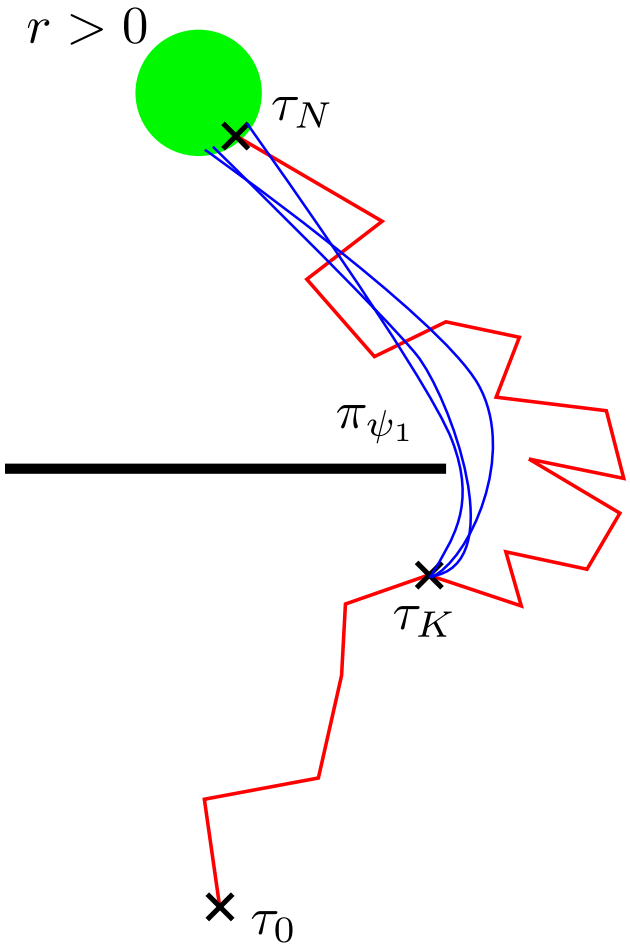


Quel est le mode d'échec le plus fréquent ?



La distribution de l'accumulateur d'expérience devient trop biaisée pour être informative $\rightarrow$ *oubli catastrophique* de la politique.

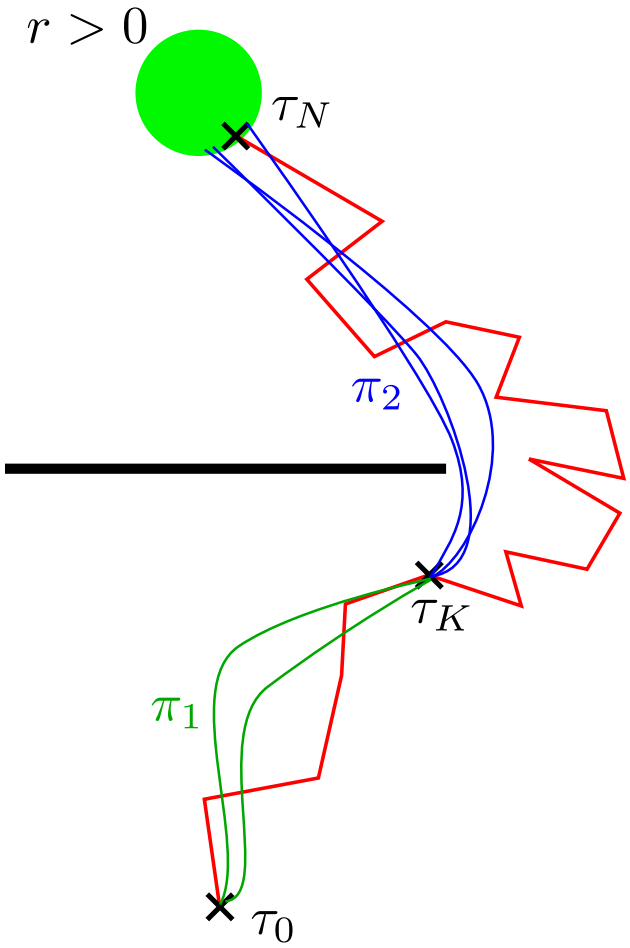
PBCS : *Plan, Backplay, Chain Skills*



Idée du *chaînage de compétences (skill chaining)* :

- Détecter quand l'apprentissage échoue
- utiliser la politique partielle π_{ψ_1} qui était capable de résoudre le problème entre τ_K et τ_N.

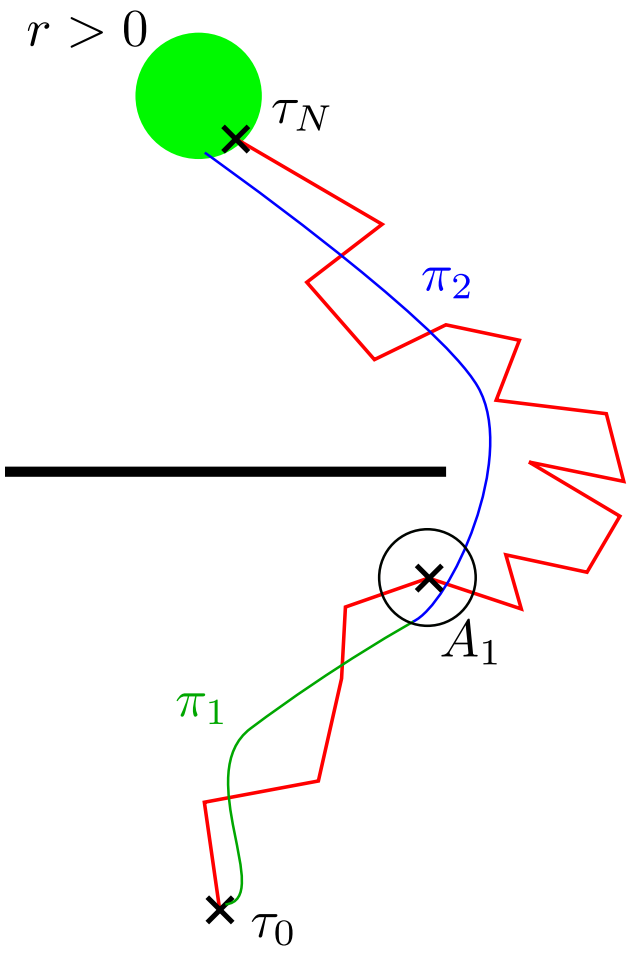
PBCS : *Plan, Backplay, Chain Skills*



On ne produit plus une unique politique mais une suite de politiques à suivre l'une après l'autre :

- π_1 de τ_0 à τ_K puis
- π_2 de τ_K à τ_N.

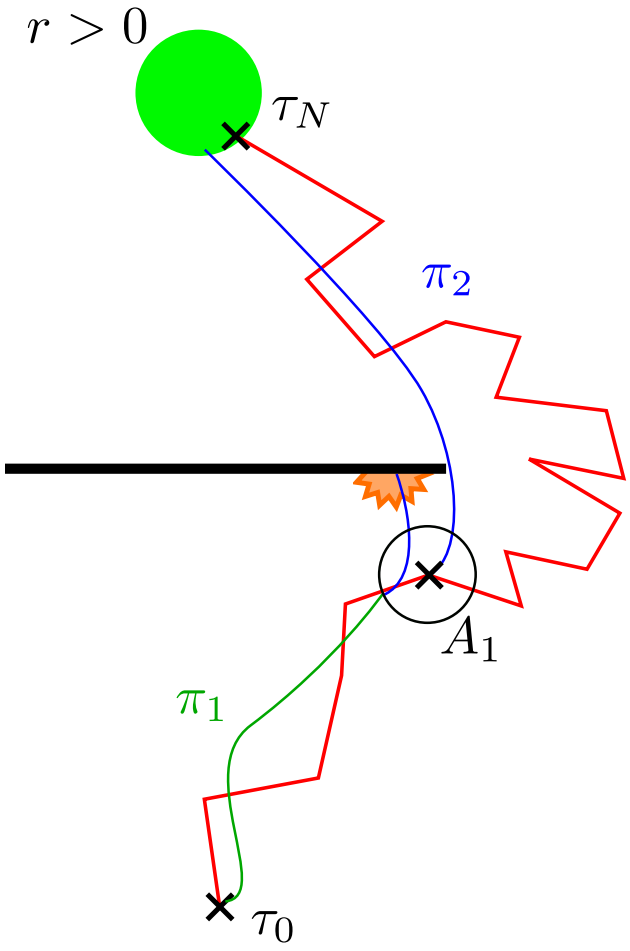
Obstacle 1 : continuité de l'epace d'états



Problème : on n'atteint jamais réellement exactement τ_K en suivant π_1.

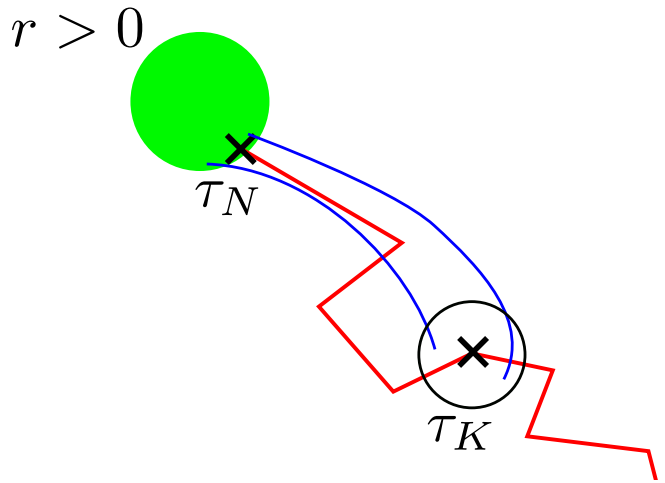
Solution : utiliser des *zones d'activation* $A_1, \ldots, A_n$ comme zones où passer d'une politique à l'autre.

Obstacle 2 : taille des zones d'activation

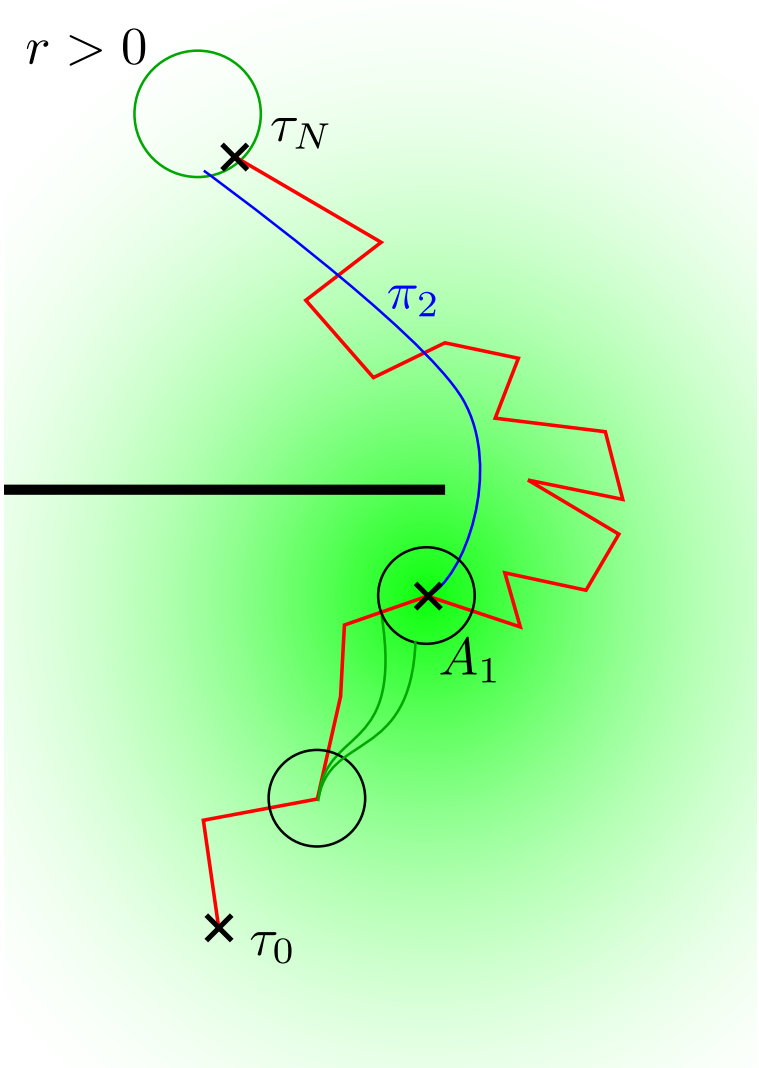


Problème : comme on change de politique avant d'atteindre τ_K, rien ne garantit que π_2 fonctionne à partir du point de changement.

Solution : entraîner les politiques avec des points de départ tirés autour de τ_K, pour être robuste au déplacement du point de départ.



Obstacle 3 : entraînement vers des états arbitraires

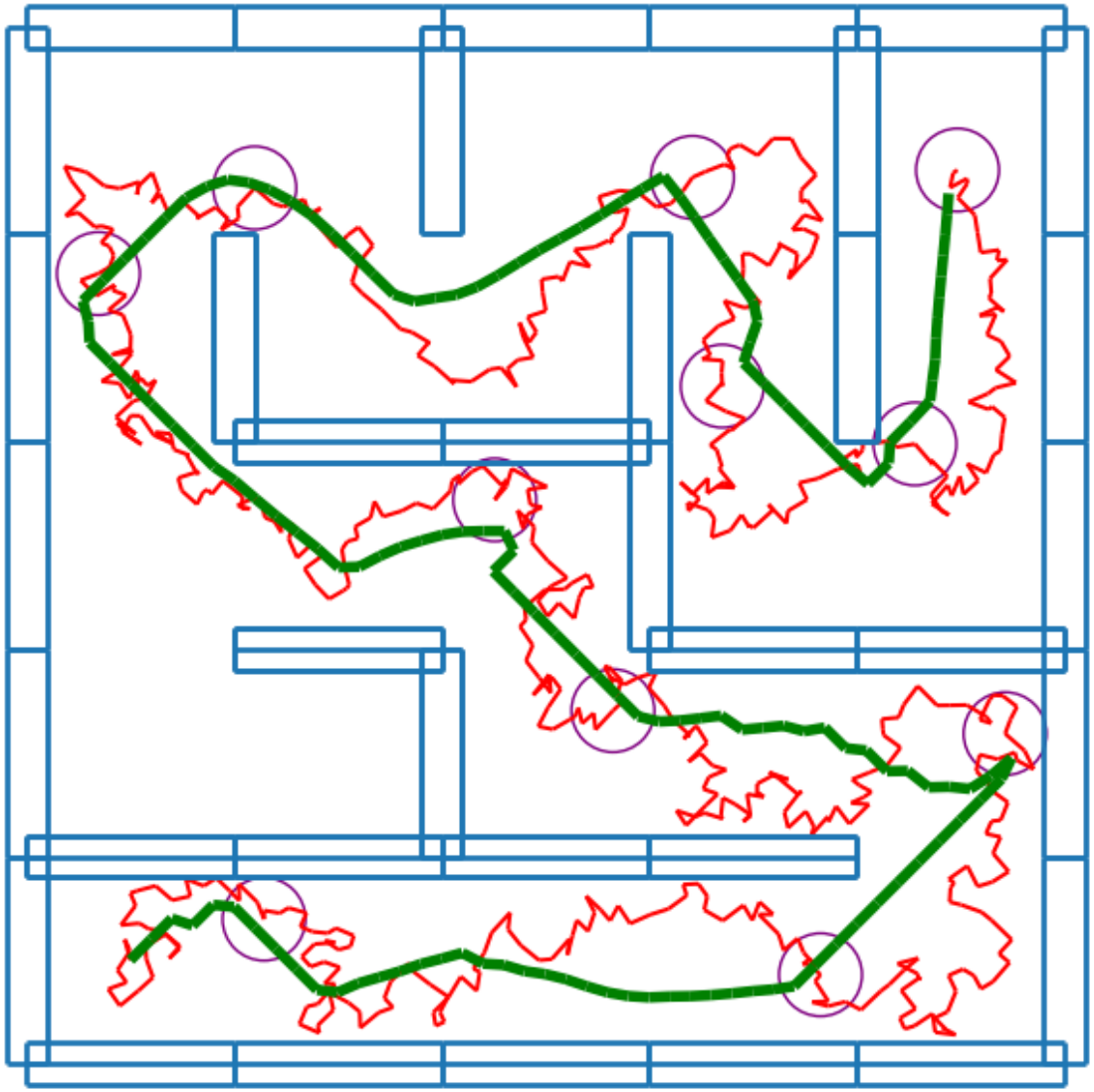


Problème : à partir de la deuxième politique on ne peut plus utiliser la récompense de l'environnement ! On cherche à atteindre une cible τ_T.

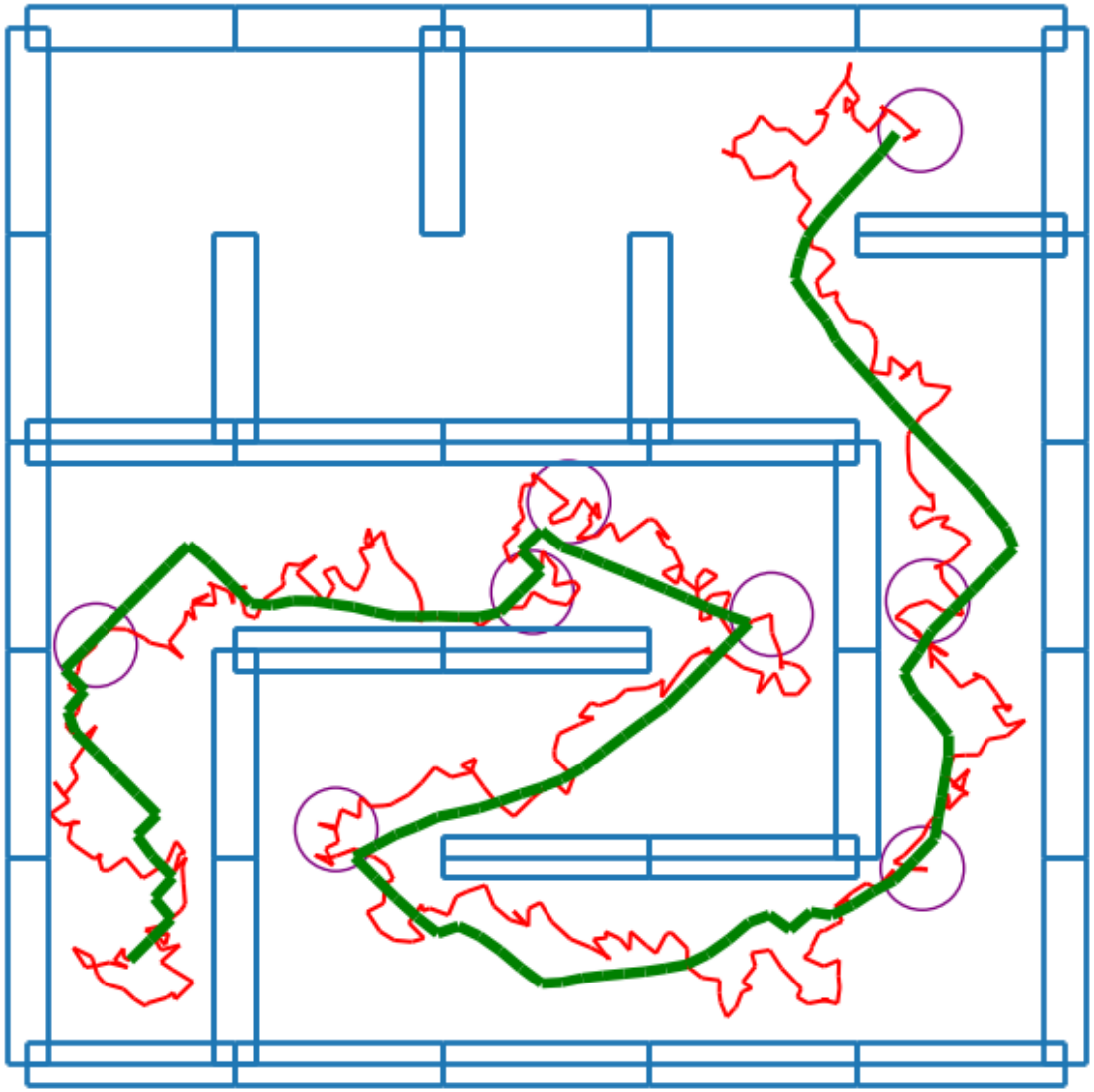
Solution : *façonnage de récompense (reward shaping)*.
$R(s, a, s') = \frac{1}{d(s',\tau_T)} - \frac{1}{d(s,\tau_T)}$.

Bonus : puisque la récompense n'est plus éparse, on élimine le "problème de gradient de DDPG" !

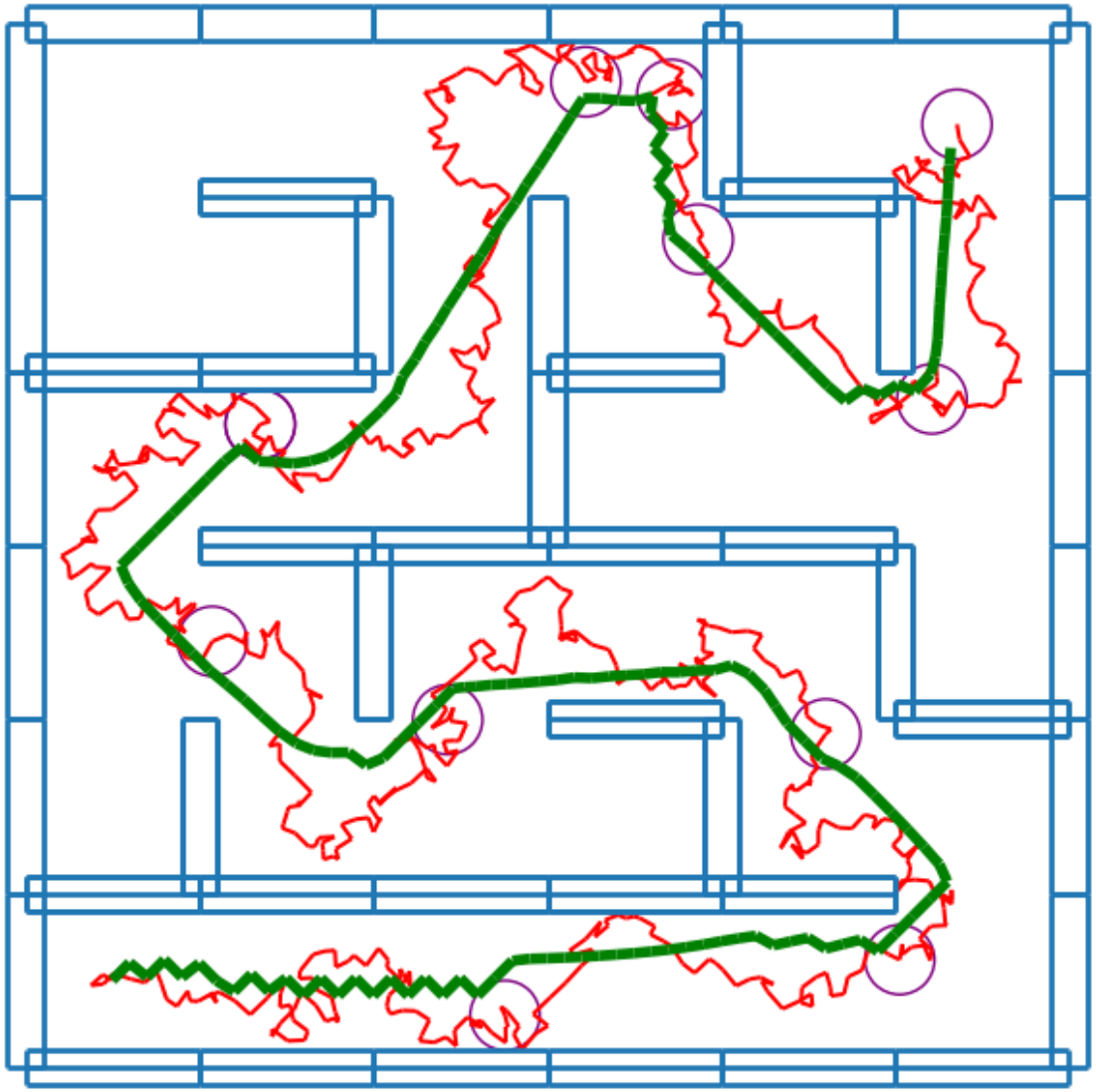
Résultats



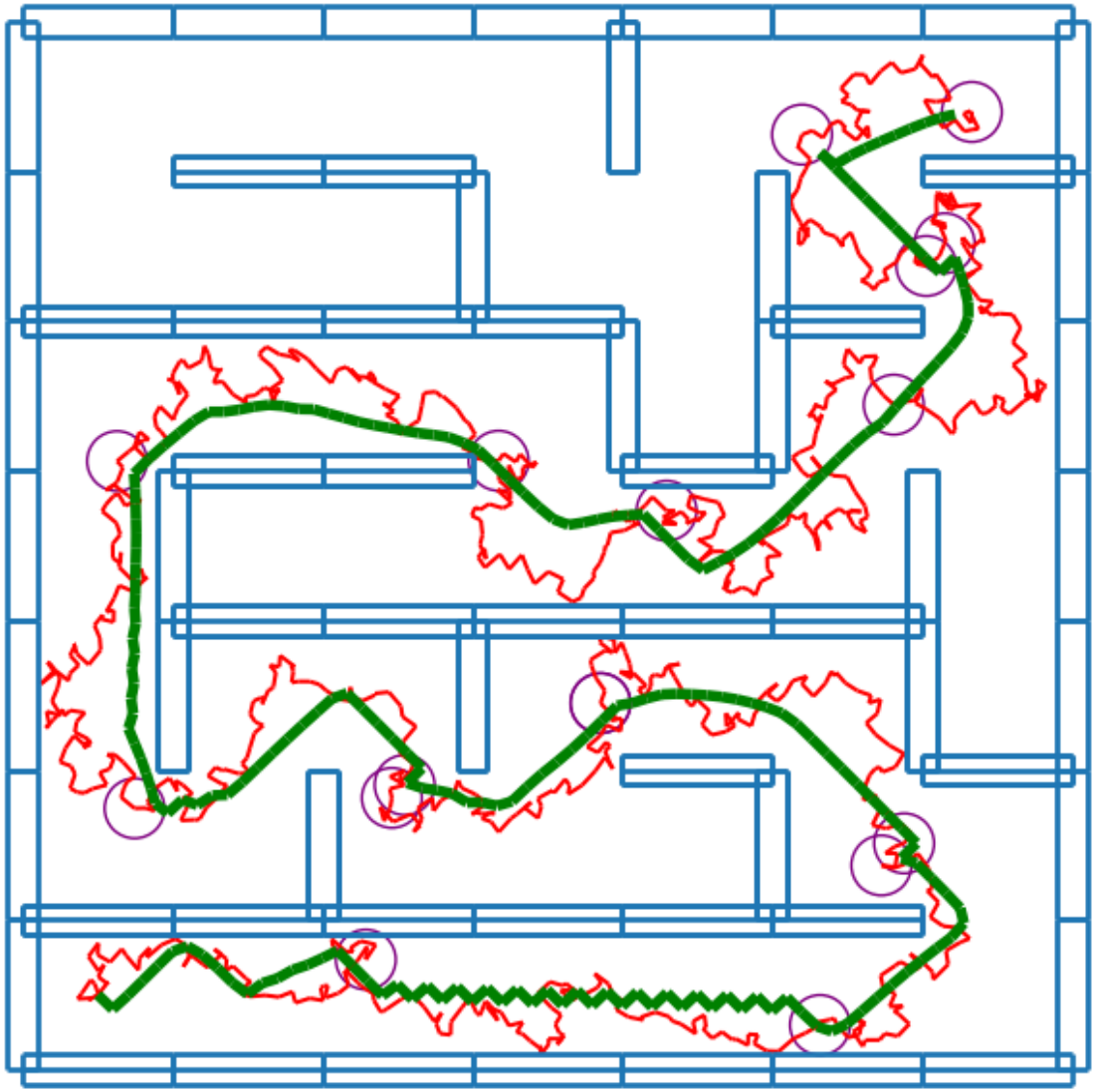
Résultats



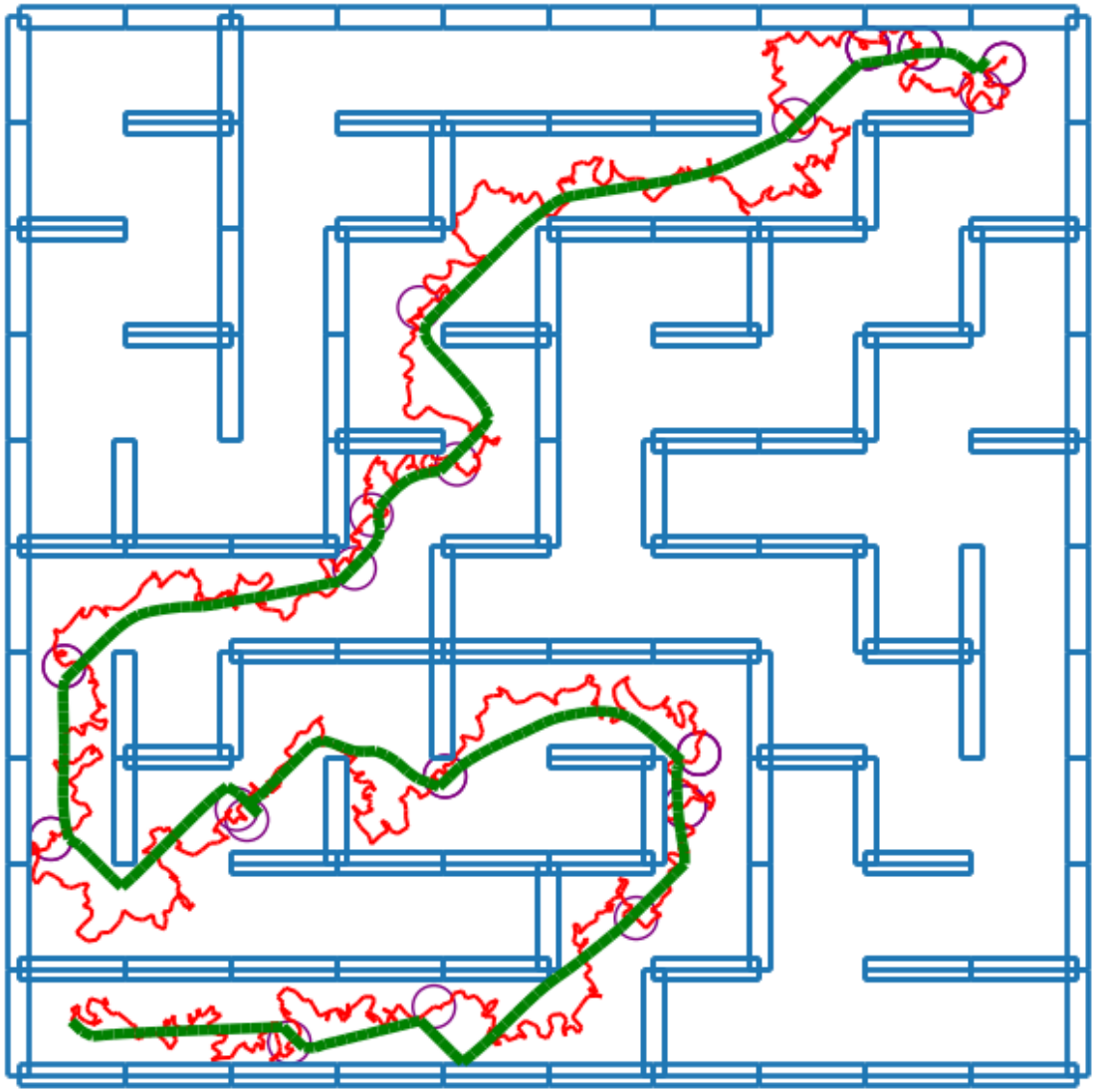
Résultats



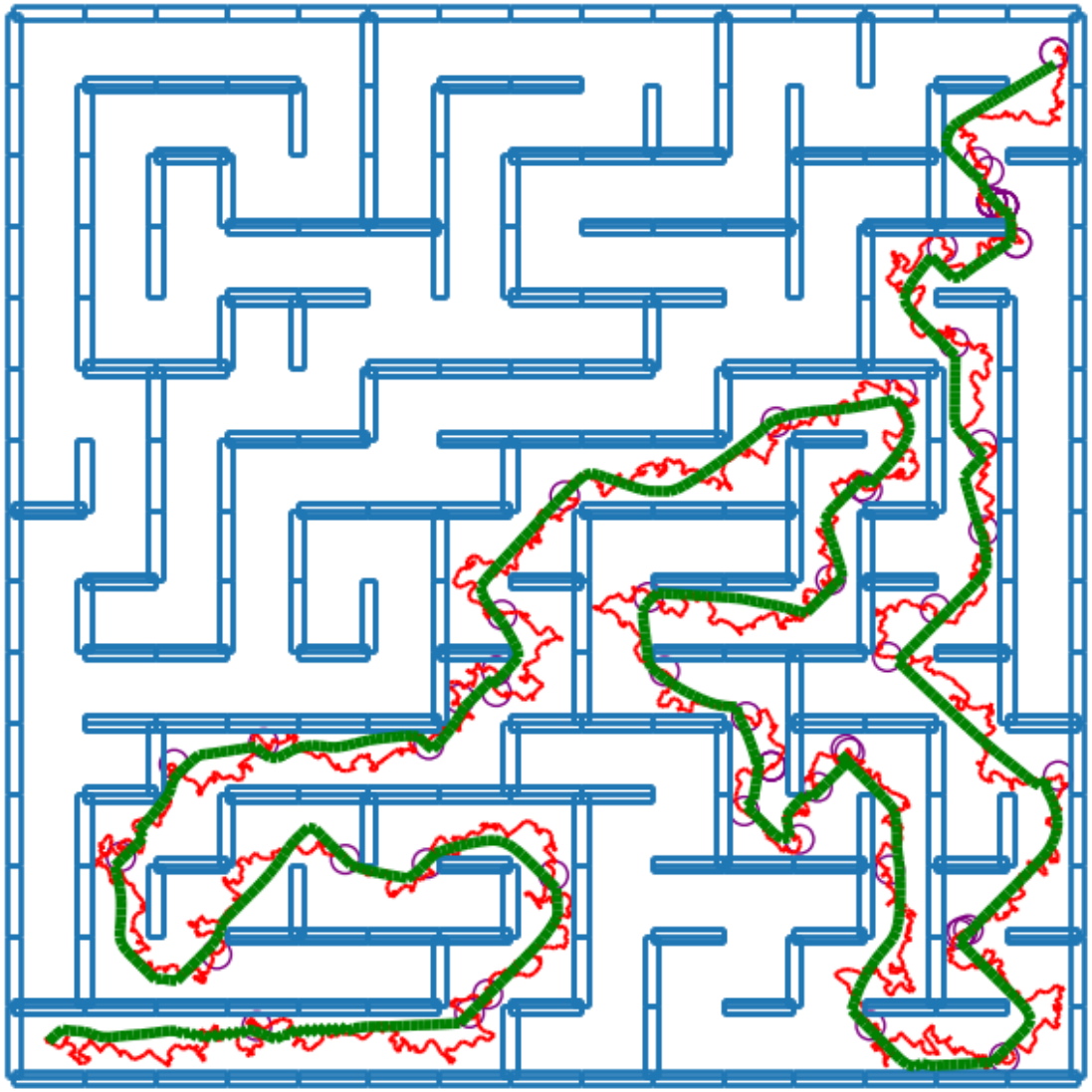
Résultats



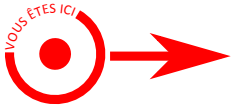
Résultats



Résultats



- Introduction
  - Cadre
  - Apprentissage par renforcement
  - Planification de mouvement
- Phase 1 : planification de mouvement
  - RRT, EST, Ex
- Phase 2 : apprentissage par renforcement
  - DDPG
  - Transfert d'expérience
  - Plan, Backplay
  - Cas de blocage de DDPG
  - Plan, Backplay, Chain Skills



- Conclusion

Avancées

1. Algorithme Ex pour l'exploration d'espaces d'état.
2. Problème de gradient dans DDPG [18].
3. PB - adaptation automatique de la vitesse d'apprentissage.
4. PBCS - synergie entre apprentissage en sens inverse et chaînage de compétences [19].

18. Matheron, Perrin et Sigaud, « Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards », 2020, ICANN.
19. Matheron, Perrin et Sigaud, « PBCS », 2020, ICANN.

Limites

1. *Reset-anywhere* ne fonctionne pas dans le cas général (états inaccessibles ou invalides)
2. La forme circulaire de A_i est arbitraire
3. Grand nombre de politiques générées, on aimerait un meilleur algo sous-jacent.

Perspectives

- À court terme
  - Utiliser un simulateur en marche arrière à la place de *reset-anywhere*.
  - Utiliser un classifieur pour définir les zones d'activation.
- À moyen terme
  - Améliorer la stabilité des algorithmes sous-jacents (DDPG, TD3)
  - Faire fonctionner PBCS sur `AntMaze`
- À long terme
  - Transfert au réel
  - Généraliser les compétences

Merci pour votre écoute !