

# Mise en oeuvre de méthodes d'apprentissage pour le problème de rendez-vous symétrique sur la ligne

Projet de Bachelor

Etudiant: Nguyen Stéphane Liem, Sup. Pierre  
Leone



7 Juillet 2021

# Mise en oeuvre de méthodes d'apprentissage pour le problème de rendez-vous symétrique sur la ligne

Projet de Bachelor

Etudiant : Nguyen Stéphane Liem, Sup. Pierre Leone

## Résumé

Le problème de rendez-vous a été présenté pour la première fois par Alpern. Dans notre cas on se concentre sur la version du problème où deux agents se trouvant initialement sur une ligne doivent se rencontrer dans un temps moyen minimal avec une même stratégie probabiliste pour les deux agents. Les agents se trouvent initialement à une distance  $d = 2$  (au temps  $t = 0$ ) mais ne savent pas si l'autre agent se trouve à gauche ou à droite sur la ligne.

Inspiré par le premier théorème sur les stratégies grilles (grid strategies) proposé par Han et al. 2008 ainsi que les stratégies mixtes distance-preserving, nous utilisons une représentation des stratégies probabilistes sous forme d'un arbre dont la profondeur correspondrait à la longueur du générateur (string) dans Han et al. 2008 qu'on notera  $N$  sauf qu'il y a l'action de ne rien faire en plus (Les actions sont donc gauche, ne rien faire et droite). Bien que nous réutilisons le terme de distance-preserving, leurs théorèmes ne s'appliquent plus forcément à nos stratégies dû à l'action supplémentaire.

Dans ce projet, nous allons considérer une contrainte plus forte sur les stratégies <sup>1</sup> que distance-preserving où les agents se retrouvent soit à leur position initiale, soit se rencontrent entre chaque sous-trajectoires <sup>2</sup>. Intuitivement, les agents se déplacent suivant la même stratégie probabiliste en étant bornés spatialement et reviennent à leurs positions initiales entre chaque pas de temps  $N$  à part s'il y a rencontre pendant la sous-trajectoire.

Nous essayons deux cas, du plus contraint au moins contraint, où le premier utilise deux phases par sous-trajectoire. Dans le premier cas, les agents sont forcés avec probabilité 1 de revenir à leurs positions initiales dans la deuxième phase alors que dans le deuxième cas, ce sera la généralisation à la contrainte citée précédemment, c'est à dire que les agents se retrouvent à leurs positions initiales entre chaque sous-trajectoire mais sont plus libres, il n'y a plus les deux phases.

Pour modifier les probabilités dans l'arbre afin de minimiser le temps moyen de rencontre, nous utilisons deux méthodes, une qui consiste à renforcer uniquement la dernière sous-trajectoire des deux agents (où il y a eu rendez-vous dans la sous-trajectoire) et l'autre qui consiste à pénaliser chaque pas de temps écoulé avec un algorithme inspiré de REINFORCE (voir policy gradients dans Sutton and Barto 2018) qui est une méthode d'apprentissage par renforcement qui permet d'optimiser directement sur les politiques.

---

<sup>1</sup>Et donc il y a moins de stratégies

<sup>2</sup>Une sous-trajectoire est générée par la stratégie probabiliste et si pas de rendez-vous, on répète, voir l'idée du *n-Markovian* dans Han et al. 2008

---

Dans les deux méthodes, nous nous basons majoritairement sur la méthode de projection de gradient dans Luenberger and Ye 2015 pour garder les contraintes d'avoir toujours des probabilités après mise à jour.

Le but principal est donc d'essayer d'optimiser par dessus les différents hyperparamètres et paramètres des arbres (par exemple optimiser sur chaque profondeur de l'arbre similairement à Han et al. 2008 et pour chaque profondeur de l'arbre, obtenir la meilleure stratégie en modifiant les probabilités).

# **Remerciements**

J'aimerai remercier mon superviseur Pierre Leone pour m'avoir guidé à travers tout le projet. Les idées dans le projet ne proviennent pas seulement de moi mais aussi de mon superviseur. Par exemple, l'idée des stratégies avec les deux phases comme premières stratégies à essayer provient de mon superviseur. C'est aussi à travers nos discussions que nous avons découvert plus sur les contraintes des différentes stratégies.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>9</b>
<b>2</b>	<b>Problème et modélisation</b>	<b>11</b>
2.1	Actions et observations . . . . .	12
2.2	Positions relatives, positions absolues et positions initiales absolues . . . . .	12
2.3	Transitions et stratégie . . . . .	13
2.4	Récompenses . . . . .	14
2.5	Arbre . . . . .	15
2.6	Temps moyen de rencontre . . . . .	16
2.6.1	Conditions pour la rencontre . . . . .	17
2.6.2	Temps de rencontre . . . . .	18
2.6.3	Algorithme pour calcul exact du temps moyen de rencontre . . . . .	19
<b>3</b>	<b>Analyse de l'article Improved Bounds</b>	<b>21</b>
3.1	Exemples et simulations avec nos arbres . . . . .	22
3.2	Rencontre . . . . .	25
3.2.1	Temps moyen pour la stratégie avec $R_2^s = 7$ . . . . .	25
<b>4</b>	<b>Stratégies avec retour à position initiale absolue : avec deux phases et sans deux phases</b>	<b>27</b>
4.0.1	Quelques définitions . . . . .	27
4.0.2	Diagramme de simulation avec mise à jour des probabilités . . . . .	28
4.0.3	Stratégies et particularités . . . . .	30
4.0.4	Limites de nos stratégies . . . . .	34
<b>5</b>	<b>Mise à jour des probabilités</b>	<b>35</b>
5.1	Méthode de projection de gradient appliquée à des probabilités . . . . .	36
5.1.1	Projection sans les vecteurs d'update . . . . .	36
5.1.2	Projection avec les vecteurs d'update . . . . .	38
5.2	Renforcer la dernière sous-trajectoire . . . . .	40
5.2.1	Un désavantage de nos méthodes et introduction d'un taux d'exploration $\epsilon$ . . . . .	40
5.3	REINFORCE-like . . . . .	43
5.3.1	Avantages de policy gradient contre value-based . . . . .	43
<b>6</b>	<b>Résultats</b>	<b>44</b>
6.1	Estimation du temps moyen de rencontre . . . . .	44
6.1.1	Estimation du temps moyen de rencontre lors des simulations sans mise à jour des stratégies . . . . .	44

6.1.2	Estimation du temps moyen de rencontre lors des simulations avec mise à jour des stratégies . . . . .	45
6.2	Stratégies avec deux phases . . . . .	47
6.2.1	Renforcer dernière sous-trajectoire . . . . .	47
6.3	Stratégies sans deux phases . . . . .	52
6.3.1	Renforcer dernière sous-trajectoire . . . . .	52
6.3.2	REINFORCE-like . . . . .	55
<b>7</b>	<b>Conclusion</b>	<b>57</b>
7.1	Futures améliorations possibles . . . . .	57
<b>8</b>	<b>Appendice</b>	<b>59</b>
8.1	Stratégies avec deux phases . . . . .	59
8.1.1	Renforcer dernière sous-trajectoire . . . . .	59
8.2	Stratégies sans deux phases . . . . .	60
8.2.1	Renforcer dernière sous-trajectoire . . . . .	60
8.2.2	REINFORCE-like . . . . .	60

# Table des figures

2.1	<i>Boucle action-perception; Transitions de l'agent <math>i</math> avec la stratégie <math>\pi</math> avec états partiellement observables</i>	13
2.2	<i>Un morceau de l'arbre avec notation en vecteurs</i>	15
2.3	<i>Conditions pour la rencontre:</i> Les courbes bleues et rouges sont des exemples de trajectoires absolues des deux agents. Les cercles noirs correspondent à des rendez-vous.	17
3.1	<i>Deux premières stratégies distance-preserving présentées dans l'article Han et al. 2008</i>	22
3.2	<i>Trois stratégies distance-preserving présentées dans l'article Han et al. 2008</i>	23
3.3	<i>Illustration des arbres correspondant à deux stratégies distance-preserving présentées dans l'article Han et al. 2008 avec <math>\mathcal{A} = \{-1, 0, 1\}</math></i>	24
3.4	<i>Illustration des rencontres possibles pour une sous-trajectoire pour trois stratégies distance-preserving présentées dans l'article avec <math>d = 2</math> et avec <math>F = 1</math></i>	25
4.1	<i>Diagramme de simulation avec mise à jour des probabilités pour stratégies avec retour à la position initiale absolue</i>	28
4.2	<i>Exemples de positions relatives possibles pour des stratégies avec retour à la position initiale avec et sans les deux phases</i>	30
4.3	<i>Stratégies sans les deux phases, détails et cas particulier</i>	31
4.4	<i>Actions possibles des stratégies avec retour à position initiale absolue. Comparaison entre avec les deux phases et sans les deux phases</i>	33
5.1	<i>Méthode de projection de gradient sur l'hyperplan <math>\sum_i x_i = 1</math> sans forcément obtenir des probabilités</i>	37
5.2	<i>Méthode de projection de gradient sur l'hyperplan <math>\sum_i x_i = 1</math> en forçant l'obtention des probabilités</i>	37
5.3	<i>Méthode de projection de gradient sur l'hyperplan <math>\sum_i x_i = 1</math> en forçant l'obtention des probabilités et avec <math>\mathbf{u}(o) = [0 \ 1 \ 1]^T</math> et ici <math>\mathbf{m} = \mathbf{u}(o)</math></i>	39
6.1	<i>6 stratégies avec deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement. <math>\epsilon = 0</math>, 10001 épisodes</i>	47
6.2	<i>3 stratégies avec deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement. <math>\epsilon = 0</math>, 10001 épisodes</i>	48

6.3	<i>Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies avec deux phases avec mise à jour en renforçant la dernière sous-trajectoire . . . . .</i>	49
6.4	<i>Temps moyen de rencontre exact comparé à celles estimées pour 2501 épisodes . . . . .</i>	50
6.5	<i>10 simulations pour les stratégies avec les deux phases avec 2501 épisodes, <math>\epsilon = 0</math>, mise à jour en renforçant la dernière sous-trajectoire uniquement . . . . .</i>	51
6.6	<i>5 stratégies sans deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement. <math>\epsilon = 0</math>, 10001 épisodes . . . . .</i>	52
6.7	<i>Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies sans deux phases avec mise à jour en renforçant la dernière sous-trajectoire . . . . .</i>	53
6.8	<i>10 simulations pour les stratégies sans les deux phases avec 10001 épisodes, <math>\epsilon = 0</math>, mise à jour en renforçant la dernière sous-trajectoire uniquement . . . . .</i>	54
6.9	<i>Illustration de la stratégie sans deux phases avec mise à jour par la méthode REINFORCE-like avec <math>N = 4</math>, <math>\epsilon = 0</math>, 50001 épisodes et sans les arcs qui ont des probabilités plus faibles que 0.098 . . . . .</i>	55
6.10	<i>Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies sans deux phases avec mise à jour avec REINFORCE-like . . . . .</i>	56

# Chapitre 1

## Introduction

Le problème de rendez-vous a été présenté pour la première fois par Alpern. Dans notre cas on se concentre sur la version du problème où deux agents se trouvant initialement sur une ligne doivent se rencontrer dans un temps moyen minimal avec une même stratégie probabiliste pour les deux agents. Les agents se trouvent initialement à une distance  $d = 2$  (au temps  $t = 0$ ) mais ne savent pas si l'autre agent se trouve à gauche ou à droite sur la ligne.

Inspiré par le premier théorème sur les stratégies grilles (grid strategies) proposé par Han et al. 2008 ainsi que les stratégies mixtes distance-preserving, nous utilisons une représentation des stratégies probabilistes sous forme d'un arbre dont la profondeur correspondrait à la longueur du générateur (string) dans Han et al. 2008 qu'on notera  $N$  sauf qu'il y a l'action de ne rien faire en plus (Les actions sont donc gauche, ne rien faire et droite). Bien que nous réutilisons le terme de distance-preserving, leurs théorèmes ne s'appliquent plus forcément à nos stratégies dû à l'action supplémentaire.

Dans ce projet, nous allons considérer une contrainte plus forte sur les stratégies<sup>1</sup> que distance-preserving où les agents se retrouvent soit à leur position initiale, soit se rencontrent entre chaque sous-trajectoires<sup>2</sup>. Intuitivement, les agents se déplacent suivant la même stratégie probabiliste en étant bornés spatialement et reviennent à leurs positions initiales entre chaque pas de temps  $N$  à part s'il y a rencontre pendant la sous-trajectoire.

Nous essayons deux cas, du plus contraint au moins contraint, où le premier utilise deux phases par sous-trajectoire. Dans le premier cas, les agents sont forcés avec probabilité 1 de revenir à leurs positions initiales dans la deuxième phase alors que dans le deuxième cas, ce sera la généralisation à la contrainte citée précédemment, c'est à dire que les agents se retrouvent à leurs positions initiales entre chaque sous-trajectoire mais sont plus libres, il n'y a plus les deux phases.

Pour modifier les probabilités dans l'arbre afin de minimiser le temps moyen de rencontre, nous utilisons deux méthodes, une qui consiste à renforcer uniquement la dernière sous-trajectoire des deux agents (où il y a eu rendez-vous dans la sous-trajectoire) et l'autre qui consiste à pénaliser chaque pas de temps écoulé avec un algorithme inspiré de REINFORCE (voir policy gradients dans Sutton and Barto 2018) qui est une méthode d'apprentissage par renforcement qui permet d'optimiser

---

<sup>1</sup>Et donc il y a moins de stratégies

<sup>2</sup>Une sous-trajectoire est générée par la stratégie probabiliste et si pas de rendez-vous, on répète, voir l'idée du *n*-Markovian dans Han et al. 2008

directement sur les politiques.

Dans les deux méthodes, nous nous basons majoritairement sur la méthode de projection de gradient dans Luenberger and Ye 2015 pour garder les contraintes d'avoir toujours des probabilités après mise à jour.

Le but principal est donc d'essayer d'optimiser par dessus les différents hyper-paramètres et paramètres des arbres (par exemple optimiser sur chaque profondeur de l'arbre似ilairement à Han et al. 2008 et pour chaque profondeur de l'arbre, obtenir la meilleure stratégie en modifiant les probabilités).

# Chapitre 2

## Problème et modélisation

Dans notre projet, nous nous concentrons sur le problème de rendez-vous symétrique<sup>1</sup> sur une ligne qui est un problème ouvert où la stratégie doit être forcément probabiliste et la même pour les deux agents. Sans stratégie probabiliste, le temps de rencontre moyen serait infini si les deux agents ont la même stratégie (voir Han et al. 2008 pour plus de détails).

Dans le problème, deux agents se trouvent initialement sur une ligne avec une distance  $d$  entre eux sans savoir si l'autre agent se trouve à sa gauche ou à sa droite et doivent se rencontrer dans un temps moyen minimal avec une même stratégie probabiliste  $\pi$ .

Nous voulons donc minimiser ce temps moyen de rencontre en modifiant  $\pi$  et pour notre cas, nous prenons  $d = 2$  comme dans Han et al. 2008 avec la vitesse des agents qui est bornée en valeur absolue par 1 et nous aurons au maximum 3 actions, gauche, ne rien faire et droite.

Puisqu'il y a deux agents, nous utilisons au cours du projet l'indice  $[i]$  en exposant avec  $i \in \{0, 1\}$  indiquant l'agent  $i$ . De plus, nous traitons le problème comme cas épisodique (voir Sutton and Barto 2018) où chaque épisode<sup>2</sup> se termine à la rencontre des deux agents.

Contrairement à l'article de Han et al. 2008 que nous analyserons plus tard, cela est possible d'avoir des rendezvous à des temps non entiers avec l'ajout de l'action "ne rien faire". De ce fait, les positions relatives/absolues peuvent s'arrêter après rendezvous puisque nous les calculons en ignorant s'il y a des rendezvous. Nous montrerons plus tard comment nous calculons le temps moyen de rencontre avec ces temps qui ne sont plus forcément entiers.

Avant de définir le temps moyen de rencontre que nous voudrons minimiser, il nous faut définir les observations, positions relatives, positions absolues, positions initiales absolues, les actions et la politique/stratégie.

---

<sup>1</sup>Le terme "symétrique" vient du fait que nous avons une même stratégie probabiliste pour les deux agents. Cela est utile pour donner de mêmes instructions à différents agents et donc aussi sans se soucier de devoir choisir quel agent a quelles instructions.

<sup>2</sup>Nous pouvons voir un épisode comme une simulation de trajectoires des deux agents et celle-ci se termine à un moment donné. Après terminaison de l'épisode, la simulation est relancée et les agents recommencent à un des états initiaux possibles avec potentiellement une stratégie qui n'est plus la même

## 2.1 Actions et observations

Une action  $a^{[i]} \in \mathcal{A} = \{-1, 0, 1\}$  ne peut être que gauche, ne rien faire ou droite et une observation  $o^{[i]} \in \mathcal{O} = \{\text{start}\} \cup \mathcal{A} \cup \mathcal{A}^2 \cup \dots \cup \mathcal{A}^N$  est un vecteur contenant toutes les actions prises jusqu'à un certain temps<sup>3</sup> dans une sous-trajetoire de longueur maximale  $N$  et l'observation initiale au temps 0 est *start*<sup>4</sup>.

Pour étendre un peu les notations, nous rajoutons des fonctions de l'action par rapport au temps  $a^{[i]}(t) : \mathbb{N} \rightarrow \mathcal{A}$  et de l'observation par rapport au temps  $o^{[i]}(t) : \mathbb{N} \rightarrow \mathcal{O}$  avec  $o^{[i]}(0) = \text{start}$ .

Une sous-trajetoire (nous reverrons ce que c'est plus en détails plus loin, par exemple dans la figure 4.1) de l'agent  $i$  ressemblerait à

$$(o^{[i]}(0), a^{[i]}(0), r^{[i]}(0), o^{[i]}(1), a^{[i]}(1), r^{[i]}(1) \dots) \quad (2.1)$$

avec  $r^{[i]}(t)$  la récompense au temps  $t \geq 0$  en partant de l'observation  $o^{[i]}(t)$  et en prenant l'action  $a^{[i]}(t)$ .

Nous définissons donc  $o^{[i]}(t)$  comme un vecteur contenant  $t$  actions

$$o^{[i]}(t) = \left[ o_k^{[i]}(t) \right]_{k=0, \dots, t-1} = \left[ o_0^{[i]}(t), o_1^{[i]}(t), \dots, o_{t-1}^{[i]}(t) \right]^T, \quad t \geq 1 \quad (2.2)$$

avec  $o^{[i]}(0) = \text{start}$  et le  $k$ -ième élément de  $o^{[i]}(t)$ ;  $o_{k-1}^{[i]}(t)$  (indices commencent à 0) serait la  $k$ -ième action  $a^{[i]}(k-1)$ .

$$o_{k-1}^{[i]}(t) = a^{[i]}(k-1), \quad t \geq k \geq 1 \quad (2.3)$$

## 2.2 Positions relatives, positions absolues et positions initiales absolues

Nous définissons la position relative à la position initiale de l'agent  $i$  au temps  $t$  comme  $x_{rel}^{[i]}(t)$  avec comme position relative initiale  $x_{rel}^{[i]}(0) = 0$ .

La position absolue est la position relative plus la position initiale absolue de l'agent qui est *initpos*<sup>[i]</sup>:

$$x_{abs}^{[i]}(t) = \text{initpos}^{[i]} + x_{rel}^{[i]}(t) \quad (2.4)$$

Par la définition du problème, nous utiliserons plus tard  $\text{initpos}^{[0]} = 0$  et  $\text{initpos}^{[1]} = d$  ou  $\text{initpos}^{[1]} = -d$  avec probabilité  $\frac{1}{2}$ . De plus, nous avons cette égalité grâce à 2.3

$$x_{rel}^{[i]}(t) = \sum_{t'=0}^{t-1} o_{t'}^{[i]}(t) = \sum_{t'=0}^{t-1} a^{[i]}(t') = x_{rel}^{[i]}(t-1) + a^{[i]}(t-1) \quad (2.5)$$

Qui signifie que la position relative au temps  $t$  peut être vue comme la somme des  $t$  actions prises ou aussi juste comme la dernière position relative plus la dernière action.

---

<sup>3</sup>une action par pas de temps.

<sup>4</sup>*start* est le vecteur vide dans l'implémentation et une sous-trajetoire fait partie d'un épisode

Avec 2.4 et 2.5, on obtient

$$x_{abs}^{[i]}(t) = initpos^{[i]} + \sum_{t'=0}^{t-1} o_{t'}^{[i]}(t) = initpos^{[i]} + \sum_{t'=0}^{t-1} a^{[i]}(t') = x_{abs}^{[i]}(t-1) + a^{[i]}(t-1) \quad (2.6)$$

## 2.3 Transitions et stratégie

Nous supposons qu'en prenant une action depuis une observation, l'observation que nous obtenons après avoir pris l'action est certaine (mais pas forcément les récompenses), c'est-à-dire que  $\mathbb{P}(o'|o, a) = 1$ .

Nous définissons également la stratégie probabiliste <sup>5</sup> de cette façon

$$\pi^{[i]}(a^{[i]}|o^{[i]}) = \mathbb{P}^{[i]}(A_t = a^{[i]}|O_t = o^{[i]}), \quad t \geq 0 \quad (2.7)$$

la probabilité de choisir une action  $a^{[i]}$  au temps  $t$  sachant que l'observation ou l'état au temps  $t$  est  $o^{[i]}$ .

Puisque nous travaillons avec la même stratégie pour les deux joueurs, on va juste désormais écrire  $\pi(a^{[i]}|o^{[i]})$ .

Nous montrons les transitions entre observations dans la figure 2.1 qui est inspirée de la boucle à la page 48 de Sutton and Barto 2018.

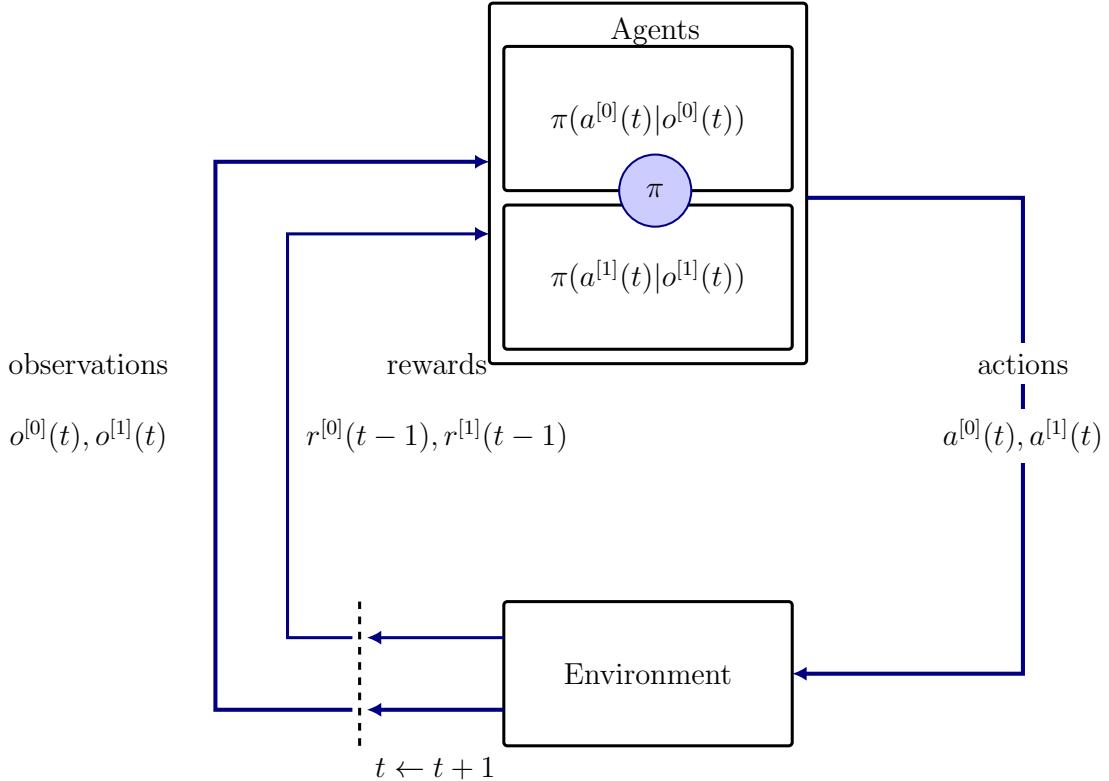


Figure 2.1: Boucle action-perception; Transitions de l'agent  $i$  avec la stratégie  $\pi$  avec états partiellement observables

<sup>5</sup>En particulier, une observation  $o$  sera dans un noeud de l'arbre et  $\pi(\cdot|o)$  sera la distribution sur les actions en partant du noeud. De plus, en prenant une action depuis un noeud, nous n'avons qu'un seul prochain noeud (et donc qu'un seul arc sortant après avoir pris l'action) mais nous pouvons potentiellement avoir une distribution de récompenses

Nos observations  $o^{[i]}(t)$  contiennent aussi les  $t$  anciennes actions de la sous-trajectoire courante. Un vrai état pourrait être noté comme  $s(t) = (x_{abs}^{[0]}(t), x_{abs}^{[1]}(t))$  (ou  $s(t) = (initpos^{[0]}, o^{[0]}(t), initpos^{[1]}, o^{[1]}(t))$ ) où les informations des deux agents sont combinées ainsi que leurs positions absolues initiales. Ces vrais états sont utilisées par l'environnement et ce dernier va vérifier à chaque pas de temps s'il y a rencontre ou pas au pas de temps  $t$  ou entre  $t$  et  $t - 1$ . Les agents ne travailleront explicitement qu'avec leurs propres observations et récompenses et non le vrai état  $s(t)$ .

En plus du schéma, l'environnement renvoie aussi s'il y a rencontre ou non et un  $\Delta t$  qui est le temps de rencontre moins le dernier pas de temps. Ce  $\Delta t$  nous permet d'obtenir les temps de rencontre. Nous rediscuterons plus en détails des conditions pour la rencontre,  $\Delta t$  etc. dans les prochaines sections.

## 2.4 Récompenses

Nous définissons  $r^{[i]}(t) = -1$  pour chaque pas de temps et donc nous avons que la récompense moyenne à partir des états initiaux  $s(0)$ <sup>6</sup> qui vaut moins le temps moyen de rencontre.

Pour notre problème, les algorithmes d'apprentissage par renforcement essaieront de maximiser la récompense moyenne à partir des états initiaux et donc ils essaieront de réduire le temps moyen de rencontre.

Dans notre projet, les récompenses ne seront utilisées que pour l'algorithme inspiré de REINFORCE que nous verrons plus en détails dans sa section dédiée (voir 5.3).

---

<sup>6</sup>Les états initiaux possibles sont définis par les positions absolues initiales possibles des deux agents. Pour nous, nous avons deux états initiaux possibles donnés par les positions absolues initiales possibles de l'agent 1.

## 2.5 Arbre

Nous allons définir nos arbres avant de montrer comment nous calculons les temps de rencontre ainsi que le temps moyen de rencontre pour une stratégie  $\pi$  car nous utilisons les arbres dans l'algorithme 1 bien qu'en général, c'est tout à fait possible d'utiliser d'autres structures de données.

Chaque noeud de l'arbre correspond à une observation  $o \in \mathcal{O}$  (la racine correspond à être au temps initial  $t = 0$  avec *start* comme observation), chaque arc partant d'un noeud vers un noeud enfant est associé à un 3-uple  $(a, p, u) \in \mathcal{A} \times [0, 1] \times \{\text{True}, \text{False}\}$  (pour action, probabilité, update. Update montre si  $p$  peut être modifiée ou non lors de la mise à jour de la politique/stratégie.)

Chaque noeud contient aussi des informations supplémentaires utiles pour la mise à jour de la politique tel que le nombre de fois qu'une action a été prise depuis le noeud (pour chaque arc sortant, nous avons un nombre associé en plus).

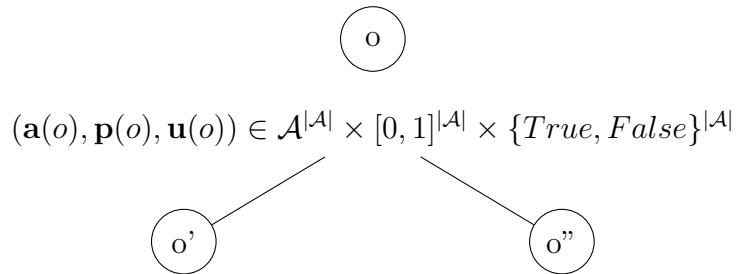


Figure 2.2: *Un morceau de l'arbre avec notation en vecteurs*

Chaque profondeur correspond à un pas de temps en plus. Pour obtenir l'observation d'un noeud enfant à partir de l'observation d'un noeud parent, nous devons rajouter une dimension supplémentaire avec la valeur de l'action prise précédemment, c'est à dire celle de l'action dans l'arc entre le noeud parent et le noeud enfant.

## 2.6 Temps moyen de rencontre

Pour toutes nos stratégies, nous allons calculer le temps moyen de rencontre de la même façon et notre démarche peut s'appliquer aussi aux stratégies de l'article Han et al. 2008<sup>7</sup>.

Nous réutilisons le terme "distance-preserving" de l'article mais certains détails ne s'appliquent plus forcément et en particulier, le temps moyen de rencontre peut être infini avec l'action supplémentaire que nous avons rajouté. Le temps moyen de rencontre peut aussi être non entier mais reste plus grand ou égal à  $\frac{d}{2}$  ( $d$  est la distance initiale en valeur absolue entre les deux agents et la vitesse de déplacement sur la ligne est bornée en valeur absolue par 1).

Le temps moyen de rencontre pour une politique/stratégie  $\pi$  où entre chaque sous-trajectoire, les agents se retrouvent soit à une distance (en valeur absolue) pareille qu'au départ ou se rencontrent dans la sous-trajectoire<sup>8</sup> que nous voulons calculer est:

$$\mathbb{E}_\pi[T] = \frac{1}{2} \sum_{k \in \{d, -d\}} \mathbb{E}_\pi[T | I^{[0]} = initpos^{[0]}, I^{[1]} = initpos^{[0]} + k] \quad (2.8)$$

En notant  $I^{[i]}$  comme la variable aléatoire pour la position initiale de l'agent  $i$  et  $d$  comme la distance initiale en valeur absolue entre les deux agents.

Sans perte de généralité, nous pouvons décaler la ligne de  $-initpos^{[0]}$  pour que la position initiale absolue de l'agent 0 soit 0 et pour que la position initiale absolue de l'agent 1 soit  $k$ .

$$\mathbb{E}_\pi[T] = \frac{1}{2} \sum_{k \in \{d, -d\}} \mathbb{E}_\pi[T | I^{[0]} = 0, I^{[1]} = k] \quad (2.9)$$

De plus, nous jetons l'exposant de  $I$  et nous ne gardons plus que  $I$  pour l'agent 1.

$$\mathbb{E}_\pi[T] = \frac{1}{2} \sum_{k \in \{d, -d\}} \mathbb{E}_\pi[T | I = k] \quad (2.10)$$

Nous voulons donc calculer  $\mathbb{E}_\pi[T | I = k]$  pour  $k \in \{d, -d\}$ .

$$\begin{aligned} \mathbb{E}_\pi[T | I = k] &= \sum_{t=0}^{N-1} \sum_{\tau(t)} p(\tau(t) | I = k) \cdot (t + \Delta t(\tau(t) | I = k))^{met(\tau(t) | I = k)} \\ &\quad \cdot (\mathbb{E}_\pi[T] + N)^{(1 - met(\tau(t) | I = k)) \wedge (t = N-1)} \end{aligned} \quad (2.11)$$

avec  $\tau(t) = a^{[0]}(t), a^{[1]}(t), o^{[0]}(t), o^{[1]}(t)$  et nous définirons juste après les conditions de rencontre et comment obtenir  $\Delta t$  qui est utilisée pour obtenir le temps de rencontre associé à la trajectoire des deux agents. Puisque dans la formule,  $t$  est

<sup>7</sup>Nous discuterons plus tard de comment passer de leur notations de string/générateur à nos arbres et vice-versa

<sup>8</sup>Cela signifie qu'à la fin d'une sous-trajectoire, si pas de rendez-vous, les agents repartent avec la même stratégie  $\pi$  depuis une distance identique à l'initiale, donc comme si nous repartions du tout début sauf que le temps de la dernière sous-trajectoire s'est déjà écoulée, augmentant le temps de rencontre

incrémentée après avoir fait les actions des deux agents, un temps de rencontre est  $t + \Delta t$  au lieu de  $t - 1 + \Delta t$  (si  $t$  était incrémentée après que les actions soient faites.)

Nous montrons dans la section avec l'article Han et al. 2008 deux calculs à la main du temps moyen de rencontre mais en décomposant différemment.

### 2.6.1 Conditions pour la rencontre

La rencontre des deux agents se fait s'ils se trouvent intuitivement à la même position absolue mais seulement si cette dernière est calculée en tenant compte des rendez-vous ce qui n'est pas le cas pour nous. Le temps de rencontre n'est plus forcément entier à cause de l'action "ne rien faire".

Les positions relatives et absolues sont des entiers relatifs car nous les calculons en additionnant des actions sans tenir compte des rencontres et donc comme condition de rencontre des deux agents, nous regardons si un agent (par exemple 0) qui était précédemment au temps  $t - 1$  à gauche d'un autre (par exemple 1) est passé à droite du deuxième agent (selon l'exemple, le deuxième agent est le 1) après une action (donc au temps  $t$ ) ou s'ils se trouvent à la même position absolue au temps entier  $t$ . Voir figure 2.3 pour des exemples.

Plus formellement, il y a rencontre si

$$\operatorname{sgn}(x_{abs}^{[0]}(t - 1) - x_{abs}^{[1]}(t - 1)) \neq \operatorname{sgn}(x_{abs}^{[0]}(t) - x_{abs}^{[1]}(t)) \quad (2.12)$$

avec

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (2.13)$$

Nous rappelons juste aussi que les équations 2.4 et 2.6 existent et que nous les utilisons fortement et aussi pour le temps de rencontre que nous allons montrer.

La formule 2.12 est utilisée à chaque fois que nous voulons voir s'il y a eu rencontre ou pas, soit dans la simulation ou dans le calcul du temps moyen de rencontre.

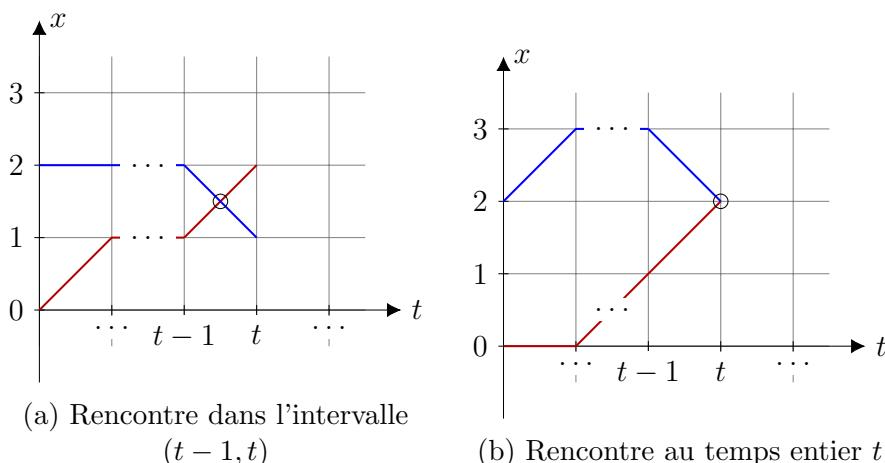


Figure 2.3: *Conditions pour la rencontre*: Les courbes bleues et rouges sont des exemples de trajectoires absolues des deux agents. Les cercles noirs correspondent à des rendez-vous.

## 2.6.2 Temps de rencontre

Pour calculer le temps moyen de rencontre, il faut calculer le temps de rencontre pour chaque trajectoire possible des deux agents et donc nous devons calculer, pour chaque rencontre, le temps écoulé entre  $t - 1$  et le temps de rencontre. Nous noterons cette différence comme  $\Delta t$  que nous utilisons pour obtenir le temps de rencontre en sommant  $\Delta t$  et  $t - 1$ .

Dans l'intervalle  $[t - 1, t]$ , les trajectoires absolues sont des fonctions affines de pente  $-1, 0$  ou  $1$  correspondant aux actions possibles  $-1, 0, 1$  mais l'ordonnée à l'origine peut être différente.

Sans perte de généralité, nous décalons l'axe du temps pour que  $t - 1$  soit l'origine juste pour l'ordonnée à l'origine dans les équations des droites. La pente de chaque droite est donnée par la dernière action de chaque agent  $a^{[i]}(t - 1)$  et son ordonnée à l'origine est donnée par la position absolue de chaque agent  $x_{abs}^{[i]}(t - 1) = initpos^{[i]} + x_{rel}^{[i]}(t - 1)$ .

Nous avons donc cette équation à une inconnue  $\Delta t$  pour l'intersection des deux droites <sup>9</sup>

$$a^{[0]}(t - 1) \cdot \Delta t + x_{abs}^{[0]}(t - 1) = a^{[1]}(t - 1) \cdot \Delta t + x_{abs}^{[1]}(t - 1) \quad (2.14)$$

$$x_{abs}^{[0]}(t - 1) = (a^{[1]}(t - 1) - a^{[0]}(t - 1)) \cdot \Delta t + x_{abs}^{[1]}(t - 1) \quad (2.15)$$

Nous obtenons donc cette formule qui est utilisée après avoir regardé si  $\text{sgn}(x_{abs}^{[0]}(t - 1) - x_{abs}^{[1]}(t - 1)) \neq \text{sgn}(x_{abs}^{[0]}(t) - x_{abs}^{[1]}(t))$  est vrai<sup>10</sup>

$$\Delta t = \frac{x_{abs}^{[0]}(t - 1) - x_{abs}^{[1]}(t - 1)}{a^{[1]}(t - 1) - a^{[0]}(t - 1)} \quad (2.16)$$

qui est à la fois utilisée pour calculer exactement le temps moyen de rencontre mais aussi pour calculer les temps de rencontre dans la simulation. L'environnement, en plus de retourner s'il y a eu rencontre ou non, il renvoie donc aussi  $\Delta t$ .

---

<sup>9</sup>nous supposons avoir déjà vérifié s'il y a eu un rendez-vous avant, c'est-à-dire en vérifiant si  $\text{sgn}(x_{abs}^{[0]}(t - 1) - x_{abs}^{[1]}(t - 1)) \neq \text{sgn}(x_{abs}^{[0]}(t) - x_{abs}^{[1]}(t))$  est vrai et donc nous calculons  $\Delta t$  seulement quand nous sommes sûrs qu'il y a une intersection

<sup>10</sup>donc nous n'allons pas diviser par 0 et donc que la formule n'est jamais utilisée pour  $a^{[0]}(t - 1) = a^{[1]}(t - 1)$

### 2.6.3 Algorithme pour calcul exact du temps moyen de rencontre

Nous allons montrer dans l'algorithme 1 comment calculer le temps moyen de rencontre pour un arbre.

**Rappel:**

$$\mathbb{E}_\pi[T] = \frac{1}{2} \sum_{k \in \{d, -d\}} \sum_{t=0}^{N-1} \sum_{\tau(t)} p(\tau(t)|I=k) \cdot (t + \Delta t(\tau(t)|I=k))^{met(\tau(t)|I=k)} \cdot (\mathbb{E}_\pi[T] + N)^{(1-met(\tau(t)|I=k)) \wedge (t=N-1)} \quad (2.17)$$

avec  $\tau(t) = a^{[0]}(t), a^{[1]}(t), o^{[0]}(t), o^{[1]}(t)$ . Puisque dans la formule,  $t$  est incrémentée après avoir fait les actions des deux agents, un temps de rencontre est  $t + \Delta t$  au lieu de  $t - 1 + \Delta t$  (si  $t$  était incrémentée après que les actions soient faites.)

**Dans l'algorithme,** le temps moyen de rencontre est donnée sous cette forme

$$E = E'_1 + E_2 \cdot (N + E) = \frac{E'_1 + E_2 \cdot N}{1 - E_2} = \frac{E_1}{1 - E_2} \quad (2.18)$$

avec  $E_1 = E'_1 + E_2 \cdot N$  où  $E'_1$  est le temps moyen de rencontre dans la sous-trajectoire et  $E_2$  est la probabilité de rencontre après la sous-trajectoire.

Si nous remplaçons les paramètres de  $met(\text{prev-}x_{abs}^{[0]}, a^{[0]}, \text{prev-}x_{abs}^{[1]}, a^{[1]})$  à la ligne 9 de l'algorithme dans la fonction  $met$ , la sortie de  $met$  serait pareille que d'écrire

$$\text{sgn}(\text{prev-}x_{abs}^{[0]} - \text{prev-}x_{abs}^{[1]}) \neq \text{sgn}(\text{prev-}x_{abs}^{[0]} + a^{[0]} - \text{prev-}x_{abs}^{[1]} - a^{[1]}) \quad (2.19)$$

où cela sera vrai ou faux.

Dans l'algorithme, nous utilisons des notations python pour les indices comme dans

$$p_0 = \pi(a^{[0]}|n^{[0]}.o) \cdot \prod_{k=0}^{t-1} \pi(n^{[0]}.o[k]|n^{[0]}.o[0:k]) \quad (2.20)$$

Le produit signifie la probabilité jointe des actions prises pour une trajectoire de l'agent. C'est le produit entre la probabilité de la dernière action prise avec la probabilité jointe de la séquence d'actions passées.

---

**Algorithm 1:** Exact expected meeting time given policy tree  $\pi$  with tree depth  $N$  and initial distance between agents of  $d$

---

```

input : policy tree  $\pi$  with tree depth  $N$  and initial distance between
agents of  $d$ 
output: expected meeting time  $E$ 
 $\mathcal{I} = \{d, -d\};$ 
 $E_1 = 0;$ 
 $E_2 = 0;$ 
 $e_1 = 0;$ 
 $e_2 = 0;$ 
/* for each absolute initial position of agent 1 */*
1 for  $initpos^{[1]} \in \mathcal{I}$  do
2   for  $t \in \{0, \dots, N - 1\}$  do /* for each depth of the tree */
3     for each pair of node  $(n^{[0]}, n^{[1]})$  from depth  $t$  do
4       for each pair of action  $(a^{[0]}, a^{[1]})$  from a pair of node  $(n^{[0]}, n^{[1]})$  do
          /* previous abs. position by summing elements in
           the observation *//
5          $prev-x_{abs}^{[0]} = \sum_k n^{[0]}.o;$ 
6          $prev-x_{abs}^{[1]} = \sum_k n^{[1]}.o + initpos^{[1]};$ 
        /* prob. of the sequence of actions with
          $o[0 : 0] = start$  *//
7          $p_0 = \pi(a^{[0]} | n^{[0]}.o) \cdot \prod_{k=0}^{t-1} \pi(n^{[0]}.o[k] | n^{[0]}.o[0 : k]);$ 
8          $p_1 = \pi(a^{[1]} | n^{[1]}.o) \cdot \prod_{k=0}^{t-1} \pi(n^{[1]}.o[k] | n^{[1]}.o[0 : k]);$ 
9         if  $met(prev-x_{abs}^{[0]}, a^{[0]}, prev-x_{abs}^{[1]}, a^{[1]})$  then
10            $\Delta t = \frac{prev-x_{abs}^{[0]} - prev-x_{abs}^{[1]}}{a^{[1]} - a^{[0]}},$ 
11            $e_1 \leftarrow e_1 + p_0 \cdot p_1 \cdot (t + \Delta t);$ 
12         else
13           if  $t == N - 1$  then /* no meeting for the last
            action allowed in the sub-trajectory */
14              $e_1 \leftarrow e_1 + p_0 \cdot p_1 \cdot N;$ 
15              $e_2 \leftarrow e_2 + p_0 \cdot p_1;$ 
16           end
17         end
18       end
19     end
20   end
/* weighted sum by the probability of the absolute initial
position of agent 1 */*
21    $E_1 \leftarrow E_1 + e_1 / |\mathcal{I}|;$ 
22    $E_2 \leftarrow E_2 + e_2 / |\mathcal{I}|;$ 
23    $e_1 = 0;$ 
24    $e_2 = 0;$ 
25 end
26  $E = E_1 / (1 - E_2);$ 

```

---

# Chapitre 3

## Analyse de l'article Improved Bounds

Tout au long du projet, toutes les fois où nous écrivons "dans l'article", l'article dont nous parlons sera toujours celui d'Han et al. 2008.

Les stratégies string distance-preserving dans l'article sont des stratégies de la forme  $g_1 = \{1, -1, -1, 1\}$ ,  $g_2 = \{1, 1, -1, -1\}$ ,  $x_1 \approx 0.1118$ ,  $x_2 \approx 0.8882$  où  $g_i$  contient des actions avec uniquement  $\mathcal{A} = \{-1, 1\}$  sans l'action "ne rien faire" et les  $x_i$  définissent une distribution sur les générateurs  $g_i$ .

Pour chaque sous-trajectoire, pour chaque agent, un générateur  $g_i$  est choisi avec probabilité  $x_i$  et le forward  $F \in \{1, -1\}$  est choisi avec probabilité  $\frac{1}{2}$ . Chaque agent suit ensuite chaque action de  $g_i \cdot F$  durant la sous-trajectoire. Le terme *distance-preserving* indique que les agents se retrouvent entre chaque sous-trajectoire soit toujours à la même distance entre eux que celle initiale, soit se sont rencontrés pendant la sous-trajectoire donc distance de 0.

Autrement dit, pour toutes les combinaisons de chemins possible des deux agents dans une sous-trajectoire<sup>1</sup>, les agents se rencontrent ou se retrouvent à une distance entre eux identique qu'au début de la sous-trajectoire.

Dans l'article, toutes les stratégies string distance-preserving commencent par l'action 1 donc droite mais la direction  $F$  (pour forward, soit 1, soit  $-1$ ) vers laquelle regarde l'agent change avec probabilité  $\frac{1}{2}$  entre chaque sous-trajectoire. Cela signifie que toutes les stratégies string écrites dans l'appendice de l'article ne contiennent qu'un côté et qu'il ne faut pas oublier la symétrie dans nos arbres lors du passage des stratégies sous forme string à arbre ou sinon il faut rajouter le forward  $F$  qui change avec probabilité  $\frac{1}{2}$  à chaque sous-trajectoire et n'apprendre/n'utiliser que la moitié de l'arbre. Nous choisissons la méthode sans le forward  $F$  pour ainsi voir plus tard la symétrie dans les probabilités lors de l'apprentissage d'une stratégie/politique. Dans cette section, nous nous concentrerons uniquement sur les simulations sans mise à jour des probabilités.

---

<sup>1</sup>aussi en tenant compte des positions initiales absolues possibles des deux agents

### 3.1 Exemples et simulations avec nos arbres

Nous allons montrer quelques exemples pour passer des stratégies sous forme string en forme d'arbre et vice-versa. Certaines stratégies tel que la stratégie d'Alpern (voir dans la figure 3.1) peuvent être simulée mais pas obtenable par apprentissage à travers nos méthodes car les positions absolues ne sont pas bornées <sup>2</sup>.

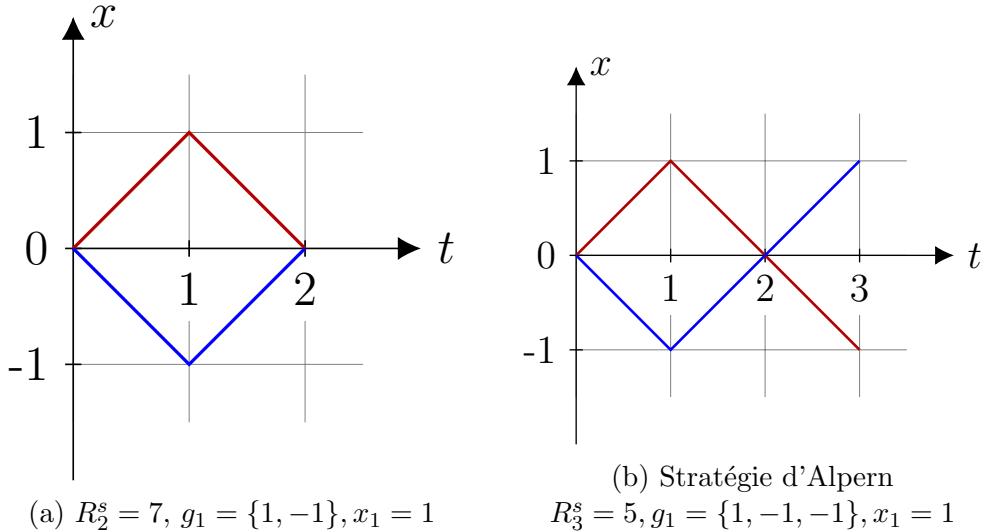


Figure 3.1: Deux premières stratégies *distance-preserving* présentées dans l'article Han et al. 2008

Nous montrons dans la figure 3.1 des actions en bleues qui ne sont pas incluses dans les stratégies écrites sous forme de string dans l'article car ils utilisent l'idée du *forward F*. Nous les montrons juste pour montrer qu'elles existent et qu'elles sont symétriques (en probabilités) à celles en rouge. Pour le reste du projet, nous ne les afficherons plus pour l'agent 0.

---

<sup>2</sup>Voir dans les prochains chapitres pour plus de détails sur nos méthodes

Dans la figure 3.2, nous ne montrons que forward  $F = 1$  à part pour la dernière stratégie où nous montrons aussi son arbre mais sans les arcs/actions qui ont 0 de probabilité (nous montrons tous les arcs juste pour les deux premières stratégies dans la figure 3.3). Les actions rouges ou bleues sans valeurs au dessus ont 1 comme probabilité.

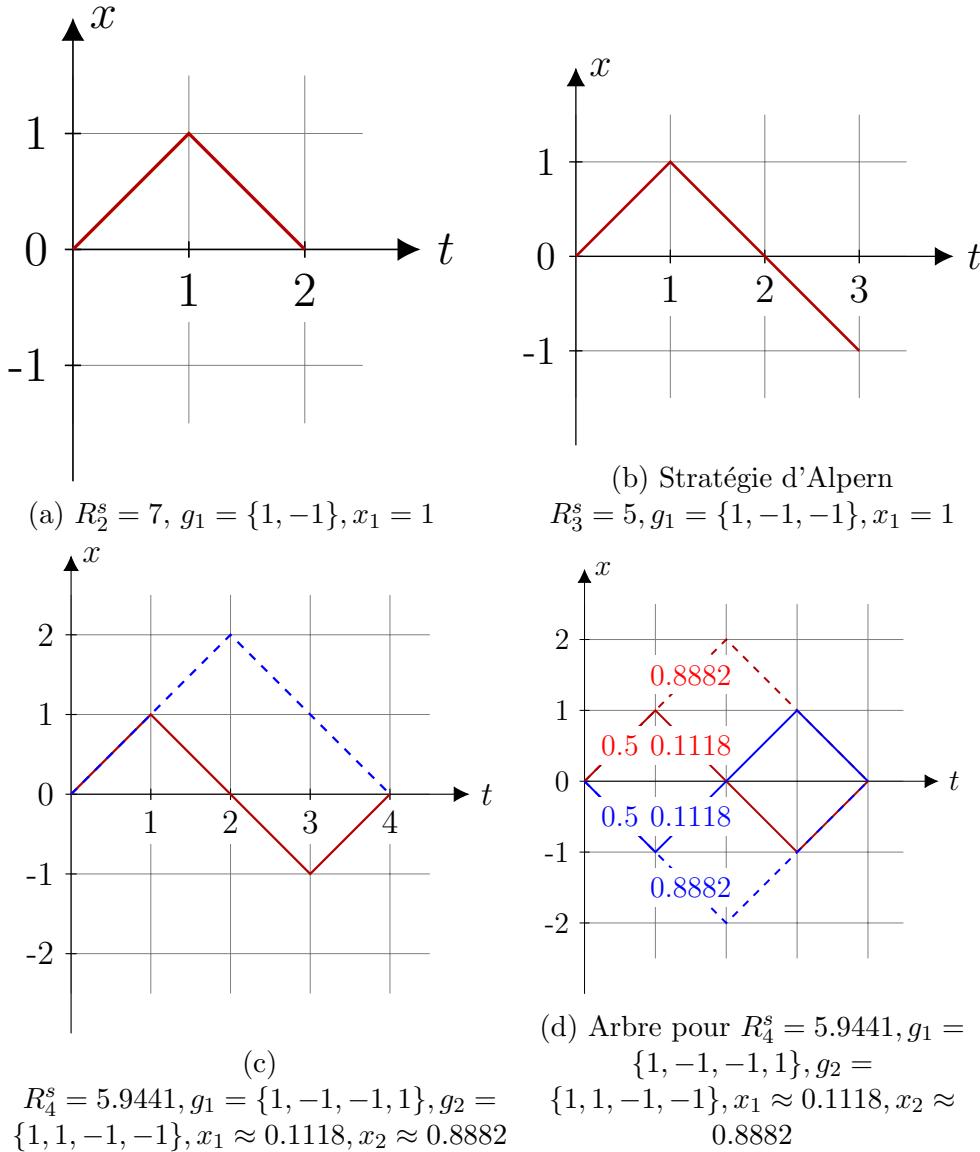


Figure 3.2: Trois stratégies distance-preserving présentées dans l'article Han et al. 2008

Nous illustrons dans la figure 3.3 des exemples d'arbres sous une version compacte où des noeuds se chevauchent.

La racine de l'arbre commence à l'origine et chaque pas de temps correspond à une nouvelle profondeur de l'arbre. Les feuilles sont donc tout à la fin à droite. Dans les deux figures, les arbres ont des noeuds qui se superposent/chevauchent (même position relative) et il y a 3 actions/arcs sortants par noeud avec leurs probabilités associées. Les actions où les arcs sont gris ont 0 de probabilité et les actions sans valeurs labellées et non grises ont 1 de probabilité. Le nombre de  $p$  correspond au nombre de noeuds avec la même position relative. Ce nombre indique le nombre de

chemins (paths) sortant du noeud pour les positions successives directes (une action en plus). Ce nombre est calculé en additionnant le nombre de chemins parents qui arrivent à la même position relative. Nous voudrons voir une certaine symétrie dans les probabilités lors de l'apprentissage car la position initiale absolue de l'agent 1 est choisie au hasard avec probabilité  $\frac{1}{2}$  pour  $d$  et idem pour  $-d$ .

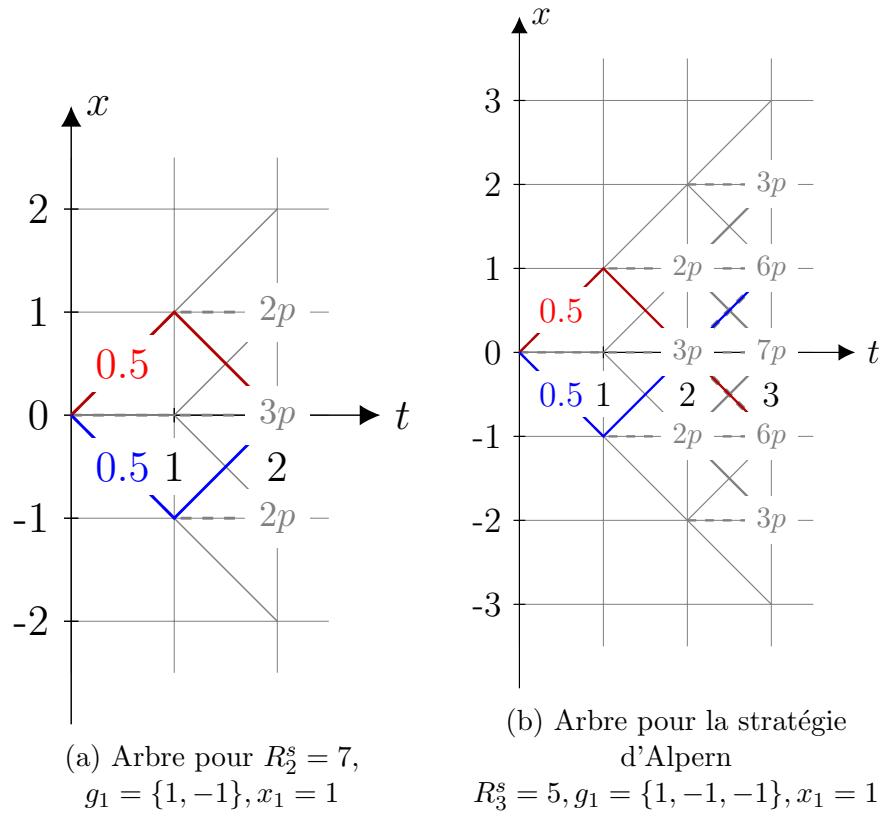


Figure 3.3: Illustration des arbres correspondant à deux stratégies distance-preserving présentées dans l'article Han et al. 2008 avec  $\mathcal{A} = \{-1, 0, 1\}$

## 3.2 Rencontre

Nous illustrons dans la figure 3.4 des rencontres possibles pour une sous-trajectoire pour trois stratégies distance-preserving présentées dans l'article avec  $d = 2$  et  $F = 1$ . L'autre sens  $F = -1$  est symétrique. La couleur noire représente l'agent 0 et les deux autres couleurs pour l'agent 1 avec position initiale  $d$  ou  $-d$ . Les cercles noirs indiquent les rendez-vous.

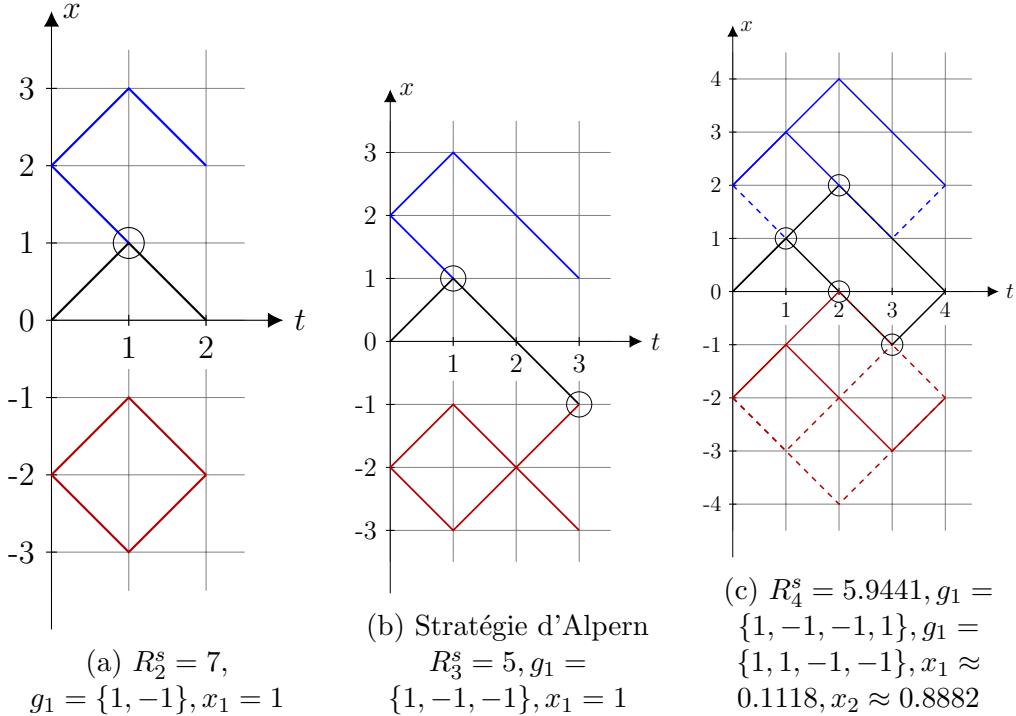


Figure 3.4: Illustration des rencontres possibles pour une sous-trajectoire pour trois stratégies distance-preserving présentées dans l'article avec  $d = 2$  et avec  $F = 1$

Nous allons montrer un exemple de calcul de temps moyen juste avec les schémas ci-dessus et

$$\mathbb{E}_\pi[T] = \frac{1}{2} \sum_{f \in \{1, -1\}} \mathbb{E}_\pi[T|F^{[0]} = f] \quad (3.1)$$

avec  $F^{[i]}$  indiquant le forward de l'agent  $i$ . Puisqu'il y a la symétrie, nous pouvons simplifier la formule en

$$\mathbb{E}_\pi[T] = \mathbb{E}_\pi[T|F^{[0]} = 1] \quad (3.2)$$

Cette formule n'est pas utilisable pour n'importe quel arbre car en général, un arbre n'a pas forcément la symétrie dans les probabilités mais lors de l'apprentissage, nous voudrons voir cette symétrie apparaître.

### 3.2.1 Temps moyen pour la stratégie avec $R_2^s = 7$

$$E_\pi[T] = 2 \cdot \frac{1}{2} \left( \frac{1}{2} \left( \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot [E_\pi[T] + 2] \right) + \frac{1}{2} \cdot [E_\pi[T] + 2] \right) = \frac{1}{4} + \frac{1}{2} + 1 + \frac{3}{4} E_\pi[T] \quad (3.3)$$

$$E_\pi[T] = \frac{7}{4} + \frac{3}{4} E_\pi[T] \iff E_\pi[T] = \frac{\frac{7}{4}}{1 - \frac{3}{4}} = 7 \quad (3.4)$$

Le premier  $\frac{1}{2}$  donne la probabilité que  $F^{[0]} = 1$  mais puisqu'il y a la symétrie, nous multiplions par 2. Ensuite nous regardons les probabilités que l'agent 1 se trouve à  $d$  ou à  $-d$ .

$E_\pi[T] + 2$  pour dire que s'il n'y a pas rendez-vous dans la sous-trajectoire, le rendez-vous se trouvera en moyenne à 2 qui est le temps écoulé de la sous-trajectoire additionné au temps moyen de rencontre. Nous pouvons le faire car les stratégies sont distance-preserving.

# Chapitre 4

## Stratégies avec retour à position initiale absolue : avec deux phases et sans deux phases

Dans notre projet, nous utilisons deux types de stratégies où les agents reviennent toujours à leur position initiale absolue entre chaque sous-trajectoire. Cela signifie aussi que tous les déplacements sont bornés.

Dans ce chapitre, nous allons illustrer le schéma global de toutes nos simulations avec mise à jour des probabilités puis nous présenterons graphiquement les deux types de stratégies et quelques particularités pour les stratégies sans les deux phases. Finalement, nous parlerons des limites de nos stratégies.

### 4.0.1 Quelques définitions

En rappelant que  $N$  est la longueur d'une sous-trajectoire et est aussi le nombre d'actions que les agents peuvent effectuer au maximum dans une sous-trajectoire, la stratégie avec deux phases consiste à forcer les agents à revenir avec probabilité 1 vers la position initiale absolue après  $K \leq N$  actions<sup>1</sup>.

Pour être certain que les agents aient le temps de revenir à leur position initiale absolue, nous imposons une borne  $M$  pour indiquer que les agents ne peuvent pas aller  $M$  cases plus loin de leur position initiale absolue. Puisque les agents peuvent donc être au plus loin à  $M$  pas/actions de leur position initiale absolue, il faut donc au plus  $M$  pas/actions<sup>2</sup> pour revenir à leur position initiale absolue et cela donne donc  $K + M = N$ . De plus, nous n'utilisons que  $1 \leq M \leq K \leq N$  car si la borne  $M$  était plus grande que  $K$ , cela voudrait dire que la borne ne serait jamais atteinte et que les agents reviennent tout le temps à leur position initiale absolue avant le dernier pas de temps dans la sous-trajectoire. Cela implique donc qu'il y aurait l'action "ne rien faire" pendant  $M - K$  pas de temps après le retour des deux agents à leurs positions initiales absolues ce qui accroît considérablement le temps moyen de rencontre alors qu'il aurait suffit d'avoir  $N = 2 \cdot K < K + M$ .

---

<sup>1</sup>Cette contrainte de faire revenir à la position initiale absolue après  $K$  actions est très forte car les agents sont très limitées dans leurs déplacements après la  $K$ -ième action, c'est la raison pourquoi nous avons les stratégies sans deux phases mais c'est intéressant d'observer les résultats des stratégies avec deux phases.

<sup>2</sup>Dans la deuxième phase où les agents sont forcés de revenir

Comme remarque, puisque le problème est symétrique et que les agents ne savent pas si l'autre agent est déjà revenu à sa position initiale absolue ou non, cela est impossible de sauter à la prochaine sous-trajectoire si jamais les deux agents sont déjà à leur position initiale absolue avant le dernier pas de temps de la sous-trajectoire.

Pour laisser plus de liberté aux agents, nous avons notre deuxième type de stratégie sans les deux phases. Pour les fois où nous utilisons  $K$  et  $M$  pour les stratégies sans les deux phases, leurs significations ne sont plus les mêmes que pour les stratégies avec les deux phases.

Pour les stratégies sans les deux phases,  $N = K$  et  $M = 0$  et la vraie borne serait  $\lfloor \frac{N}{2} \rfloor = \lfloor \frac{K}{2} \rfloor$ .

Les illustrations expliquent plus clairement les deux types de stratégies.

#### 4.0.2 Diagramme de simulation avec mise à jour des probabilités

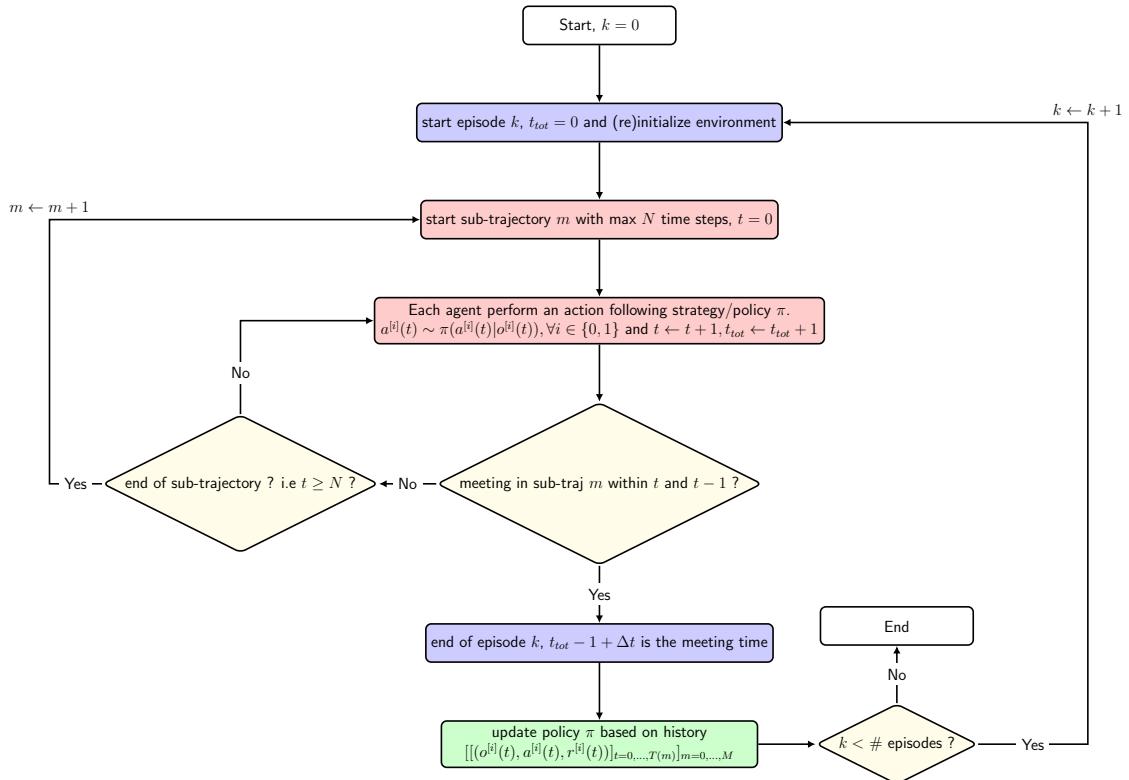


Figure 4.1: Diagramme de simulation avec mise à jour des probabilités pour stratégies avec retour à la position initiale absolue

A chaque début de sous-trajetorie et si pas de rendez-vous dans la sous-trajetorie précédente, nous supposons que les agents se trouvent toujours à distance identique que celle au tout début de l'épisode. Le diagramme sans mise à jour des probabilités est similaire mais sans la partie *update* qui est faite à chaque fin d'épisode.

Le choix des actions est basé sur la boucle d'action-perception de la figure 2.1. Si nous rajoutions un taux d'exploration  $ε ∈ [0, 1]$ , il faudrait le prendre en compte pour la prise d'actions (voir 5.2.1 pour plus d'explications).

La mise à jour des probabilités à la fin de chaque épisode est basée sur l'historique des transitions des deux agents pour chaque sous-trajetorie dans l'épisode, i.e

$[(o^{[i]}(t), a^{[i]}(t), r^{[i]}(t))]_{t=0, \dots, T(m)}]_{m=0, \dots, M}$  où  $T(m) = N - 1$  pour les sous-traj ectoires  $m$  qui ne sont pas la dernière et  $T(m) \leq N - 1$  pour la dernière sous-traj ectoire. Pour obtenir le temps moyen empiriquement, il y a plusieurs méthodes que nous utilisons pour l'estimer (voir 6.1) bien que nous pouvons le calculer exactement à partir de  $\pi$  (voir 2.6.3).

### 4.0.3 Stratégies et particularités

Nous illustrons graphiquement dans 4.2 des exemples de nos stratégies avec ou sans deux phases.

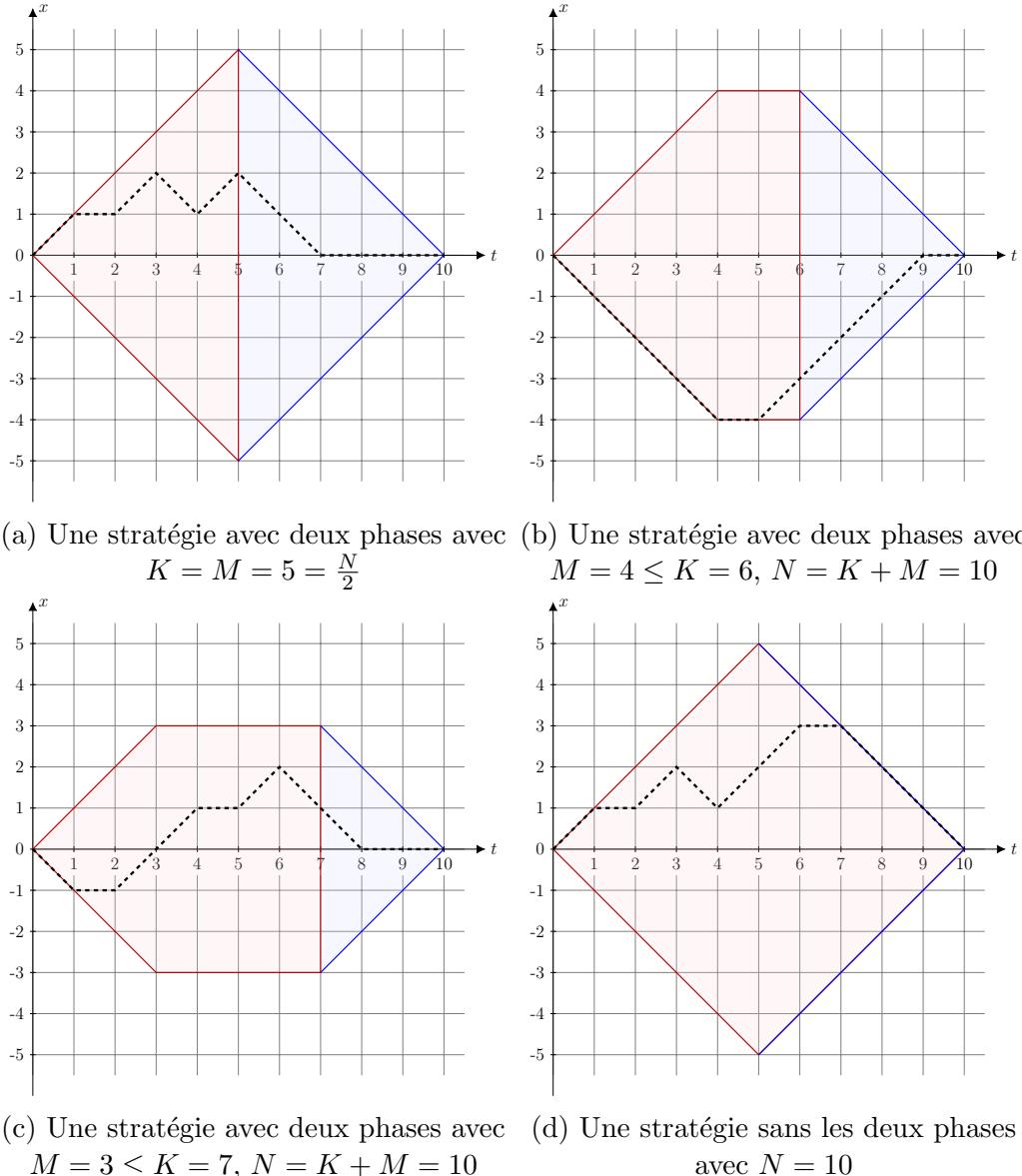


Figure 4.2: Exemples de positions relatives possibles pour des stratégies avec retour à la position initiale avec et sans les deux phases

Pour un agent  $i$  quelconque, nous montrons des exemples qui montrent les positions relatives  $x_{rel}^{[i]}(t)$  possibles en fonction du temps pour la première sous-trajectoire pour des stratégies avec retour à la position initiale pour chaque agent après  $N = 10$  pas de temps. Nous montrons aussi en noir des exemples/échantillons de premières sous-traj ectoires complètes. Les stratégies présentées sont avec les deux phases ou sans les deux phases et avec les actions  $a \in \mathcal{A} = \{-1, 0, 1\}$ . La couleur bleue représente les positions relatives où l'agent est forcé de revenir vers sa position initiale (0 en position relative signifie revenir à sa position initiale absolue) alors que la couleur rouge veut juste dire que l'agent peut se déplacer comme il veut (avec les actions qu'il peut faire) sans dépasser les bornes/bords. Nous pouvons observer que

si l'agent est sur un des deux bords (ou bornes/lignes) bleus, celui-ci est forc   de longer la ligne sur lequel il se trouve pour arriver    sa position initiale absolue car il n'a pas d'actions qui le ferait revenir plus rapidement. Nous verrons prochainement des cas particuliers ou d  tails pour la strat  gie sans les deux phases car le derni  re figure n'est pas totalement correcte avec les bords bleus et nous n'avons pas montr   le cas o   N est impair.

Pour faciliter les choses, nous d  finissons ces contraintes bleues (incluant les d  tails des strat  gies sans deux phases et le cas particulier N impair) sur la politique initiale dans les distributions sur les actions en partant d'une observation (en indiquant par exemple avec probabilit   1 de ne pas aller plus loin depuis une observation);  $\pi(\cdot|o) = \mathbf{p}(o) \in [0, 1]^{|\mathcal{A}|}$ ,  $\sum_k \mathbf{p}_k(o) = 1$  et dans les vecteurs  $\mathbf{u}(o) \in \{\text{True}, \text{False}\}^{|\mathcal{A}|}$  indiquant quelles probabilit  s peuvent   tre modifi  es lors de la mise    jour de la strat  gie. Les ments dans ces vecteurs sont li  s aux actions  $-1, 0, 1$  dans le m  me ordre.

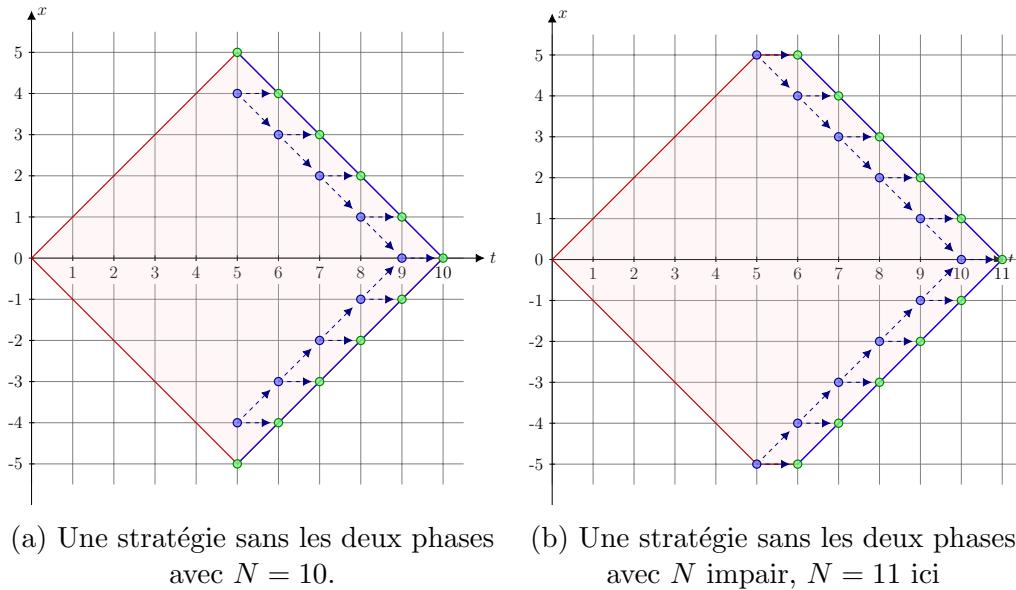


Figure 4.3: *Strat  gies sans les deux phases, d  tails et cas particulier*

Nous montrons dans la figure 4.3, deux illustrations montrant les actions possibles depuis certaines positions pour le cas  $N$  pair et impair.

Nous rappelons que si pas de rendezvous, les agents se trouvent que sur le quadrillage    chaque pas de temps avec les actions permises. Si l'agent se trouve sur un point vert, il suivra la ligne bleue non traitill  e pour revenir    sa position initiale. Si l'agent se trouve sur un point bleu, ses actions seront restreintes    ne pas bouger pendant un pas de temps ou aller vers la direction oppos  e au bord (donc aller vers sa position initiale) pour un pas de temps. Autrement dit, depuis un point bleu, l'agent ne pourra se d  placer que vers le point vert le plus proche du pas de temps suivant ou vers le point bleu le plus proche du pas de temps suivant. Pour toutes les strat  gies sans les deux phases (cela traite aussi le cas avec  $N$  impair), pour savoir si l'agent peut prendre une action, il se posera la question de si apr  s avoir fait l'action, il pourra revenir    sa position initiale avec le temps restant et si c'est le cas, il pourra faire l'action, sinon il ne pourra pas faire l'action.

Plus formellement, si la valeur absolue de la position relative au temps  $t$  de l'agent est égale à  $N - t$  (donc l'agent se trouve sur une des lignes bleues non traitillées), alors l'agent revient avec probabilité 1 vers sa position initiale absolue (position relative de 0).

S'il ne reste plus qu'un seul pas de temps restant, c'est-à-dire si  $N - t$  vaut 1 et que l'agent se trouve déjà à sa position initiale (les autres cas sont déjà traités par le cas précédent car l'agent ne peut pas dépasser la borne), il ne peut que ne rien faire pendant un pas de temps (sinon il dépasse la borne).

Si faire une action supplémentaire au temps  $t$  pour aller plus loin de la position initiale rend impossible le retour de l'agent à sa position initiale avant ou au temps  $N$ , alors les seules actions permises au temps  $t$  sont de ne rien faire dans un pas de temps ou l'action qui ramène l'agent plus proche de sa position initiale.

La condition précédente est équivalente à comparer la position relative en valeur absolue plus 1 à  $N - t$  pour voir si c'est égal (c'est comme si nous décalions les points bleus sur les points verts pour comparer). C'est l'approche que nous utiliserons dans l'implémentation.

Si aucune condition précédente n'est satisfaite, les probabilités peuvent être quelconques et dans la stratégie initiale, pour ces observations, nous mettons une distribution uniforme sur les actions depuis l'observation.

Voici, avec des formules, à quoi cela ressemble pour une certaine observation  $o(t)$  pour l'agent  $i$  (nous enlevons les  $[i]$  en exposant ici pour rendre plus clair et nous partons du principe que les positions relatives, le temps  $t$  étaient déjà calculées à partir de l'observation et que  $N$  est donné. Pour rappel,  $o(t)$  est un vecteur de  $t$  actions.)

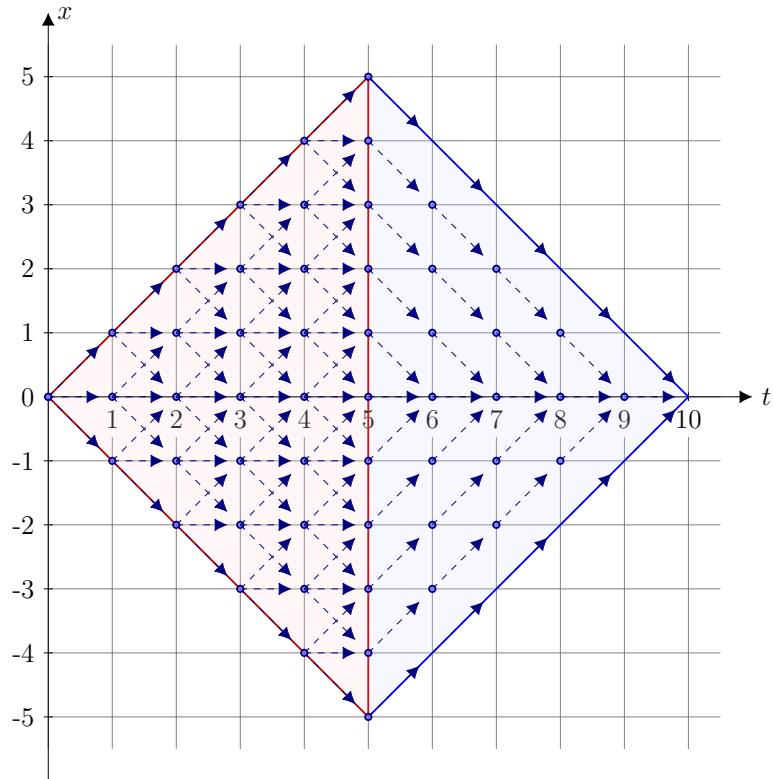
$$\mathbf{p}(o(t)) = \begin{cases} e_{2-(1+\text{sgn}(x_{\text{rel}}(t)))} & \text{if } |x_{\text{rel}}(t)| = N - t \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T & \text{if } N - t = 1 \text{ and } x_{\text{rel}}(t) = 0 \\ \frac{1}{2}e_{2-(1+\text{sgn}(x_{\text{rel}}(t)))} + \frac{1}{2}e_1 & \text{if } |x_{\text{rel}}(t)| + 1 = N - t \\ \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}^T & \text{else} \end{cases} \quad (4.1)$$

$$\mathbf{u}(o(t)) = \begin{cases} \begin{bmatrix} \text{False} & \text{False} & \text{False} \end{bmatrix}^T & \text{if } |x_{\text{rel}}(t)| = N - t \\ \begin{bmatrix} \text{False} & \text{False} & \text{False} \end{bmatrix}^T & \text{if } N - t = 1 \text{ and } x_{\text{rel}}(t) = 0 \\ \text{to\_bool}(e_{2-(1+\text{sgn}(x_{\text{rel}}(t)))} + e_1) & \text{if } |x_{\text{rel}}(t)| + 1 = N - t \\ \begin{bmatrix} \text{True} & \text{True} & \text{True} \end{bmatrix}^T & \text{else} \end{cases} \quad (4.2)$$

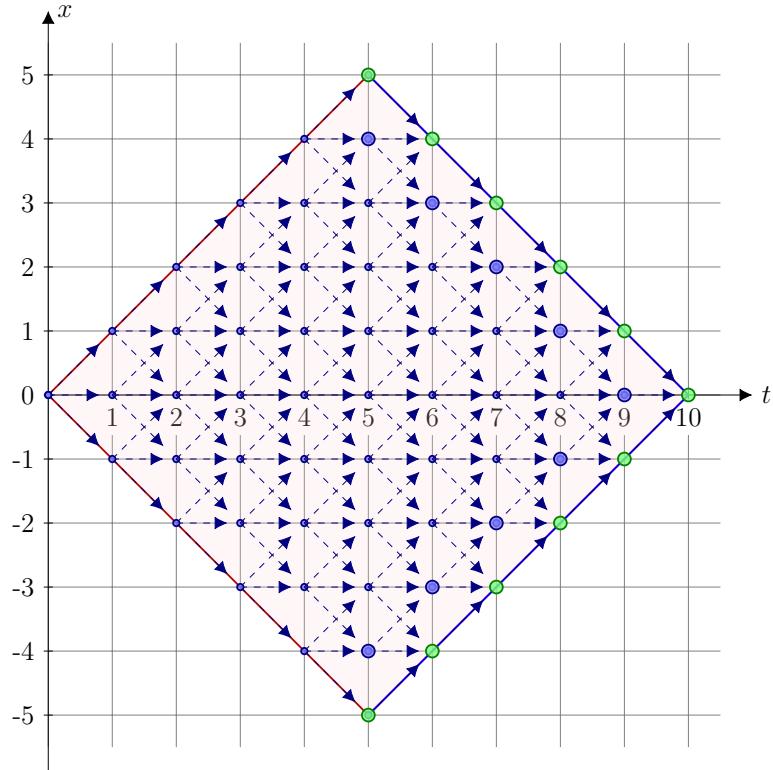
où  $e_i \in \mathbb{R}^{|\mathcal{A}|} = \mathbb{R}^3$  ( $i$  variant de 0 à 2) un vecteur de la base canonique dans  $\mathbb{R}^{|\mathcal{A}|} = \mathbb{R}^3$ . La troisième ligne, nous avions noté `to_bool` pour transformer les 1 en *True* et 0 en *False*.

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (4.3)$$

$2 - (1 + \text{sgn}(x_{\text{rel}}(t)))$  donne le mapping de l'action à prendre pour revenir plus proche de la position initiale.



(a) Actions possibles d'une stratégie avec deux phases avec  
 $K = M = 5 = \frac{N}{2}$



(b) Actions possibles d'une stratégie sans les deux phases avec  
 $N = 10$ .

Figure 4.4: *Actions possibles des stratégies avec retour à position initiale absolue. Comparaison entre avec les deux phases et sans les deux phases*

#### 4.0.4 Limites de nos stratégies

Pour les stratégies avec deux phases, à cause de sa structure, des stratégies tel que le 4-générateur de Han et al. 2008 avec  $g_1 = \{1, -1, -1, 1\}$ ,  $g_2 = \{1, 1, -1, -1\}$ ,  $x_1 \approx 0.1118$ ,  $x_2 \approx 0.8882$  ne sont pas atteignables dans l'apprentissage bien que cette stratégie fait partie des stratégies avec retour à la position initiale. Ce problème peut être causé par la borne  $M$  qui est trop petite ou car nous forçons les agents à revenir dans la deuxième phase.

Pour palier à ce problème d'avoir des contraintes trop fortes, nous avons utilisé les stratégies sans les deux phases où les agents sont un peu plus libres.

Malgré cela, certaines stratégies tel que la stratégie d'Alpern (voir dans la figure 3.1) restent non obtenables par apprentissage à travers nos méthodes (avec ou sans les deux phases) bien que simulables. La stratégie d'Alpern n'est pas apprennable car les positions absolues ne sont pas bornées. S'il n'y a pas de rendez-vous, les deux agents peuvent glisser indéfiniment sur la ligne en gardant la même distance entre eux et ne reviennent jamais à la fin d'une sous-trajectoire à la même position que celle d'au début de la sous-trajectoire.

Nos stratégies forçant les agents à revenir à leur positions initiales entre chaque sous-trajectoire fait partie des stratégies bornées spatialement et nos stratégies font aussi parties des stratégies "*distance-preserving*". Les stratégies bornées spatialement intersectent les stratégies "*distance-preserving*" mais ce n'est pas une inclusion. Cela montre donc que dépendant de comment nous définissons la structure de nos stratégies, nous ne pourrons jamais converger vers des stratégies hors de nos définitions.

# Chapitre 5

## Mise à jour des probabilités

La stratégie/politique  $\pi$  est mise à jour à chaque fin d'épisode (voir figure 4.1) et cette dernière est basée sur l'historique des transitions des deux agents pour chaque sous-trajectoire dans l'épisode, i.e  $[(o^{[i]}(t), a^{[i]}(t), r^{[i]}(t))]_{t=0, \dots, T(m)}_{m=0, \dots, M}$  où  $T(m) = N - 1$  pour les sous-trajectoires  $m$  qui ne sont pas la dernière et  $T(M) \leq N - 1$  pour la dernière sous-trajectoire.

Pour la mise à jour des probabilités, nous avons choisi de modifier directement les probabilités au lieu de paramètres mais c'est une amélioration future possible et probablement meilleure<sup>1</sup> pour éviter d'utiliser la méthode de projection de gradient (voir Luenberger and Ye 2015) appliquée à des contraintes pour les probabilités que nous allons présenter en premier. Toutes nos méthodes vont donc utiliser cette projection de gradient pour garder les contraintes d'avoir des probabilités après mise à jour.

Deuxièmement, nous allons montrer notre première méthode qui consiste à renforcer uniquement la dernière sous-trajectoire des deux agents sans utiliser les récompenses.

Troisièmement, nous allons aussi montrer notre deuxième méthode inspirée de l'algorithme Monte-Carlo Policy Gradient autrement appelé REINFORCE qui est un algorithme d'apprentissage par renforcement qui fait apprendre la politique stochastique directement et ne fait la mise à jour qu'à la fin de chaque épisode.

Pour notre deuxième méthode, nous présentons d'abord l'algorithme que nous utilisons qui est inspiré de REINFORCE. Après avoir présenté ce que nous utilisons, nous expliquons l'intuition pour les algorithmes de policy gradient ainsi que leurs avantages pour le projet par rapport aux méthodes basées sur une fonction de valeur (*state-value function* par exemple, voir Sutton and Barto 2018 pour plus de détails). REINFORCE n'est pas le meilleur algorithme policy gradient mais nous l'utilisons pour pouvoir comparer non formellement avec notre première méthode et pour apprendre davantage sur l'apprentissage par renforcement.

De nos résultats que nous montrons dans le chapitre 6, nous observons que c'est

---

<sup>1</sup>Comme amélioration future possible: utiliser une fonction d'approximation pour toute la stratégie  $\pi$  avec un softmax en sortie pour la distribution sur les actions et des observations en entrée puis faire la mise à jour par rapport aux paramètres d'un potentiel réseau de neurones. Cela permettrait potentiellement d'améliorer les performances car chaque poids/paramètre peut avoir un impact petit ou grand sur les choix des actions alors que pour nos méthodes, modifier pour les probabilités pour une distribution sur les actions à partir d'une observation ne modifient pas les probabilités d'une autre distribution sur les actions, en tout cas lors de la mise à jour des probabilités.

compliqué d'analyser numériquement. Nos méthodes peuvent donner différentes stratégies en lançant plusieurs fois l'algorithme et le temps de rencontre peut fluctuer beaucoup. De plus, nous ne savons pas si la solution est unique ou non.

Dans ce projet, nous n'avons pas essayé d'analyser la convergence des algorithmes (aussi combiné à la méthode de projection de gradient) et pas utilisé de méthodes rigoureuses pour comparer la performance des algorithmes.

Nous présenterons plus tard des graphes montrant le temps moyen de rencontre estimé pour différents hyperparamètres pour les stratégies avec les deux phases, sans les deux phases et avec la méthode de renforcer la dernière sous-trajectoire ou REINFORCE-like.

## 5.1 Méthode de projection de gradient appliquée à des probabilités

Etant donné que nous modifions directement les probabilités au lieu de paramètres, nous devons faire en sorte de garder des probabilités après mise à jour, donc que le vecteur somme à 1 et que chaque élément soit positif entre 0 et 1. Dans nos illustrations, l'espace des probabilités est indiquée en bleue.

### 5.1.1 Projection sans les vecteurs d'update

En partant d'un vecteur de probabilité  $\mathbf{p}(o)$  que nous allons modifier par la mise à jour, nous définissons  $\mathbf{z} \in \mathbb{R}^n$  à partir d'un vecteur  $\mathbf{a} \in \mathbb{R}^n$ .

$$\mathbf{z} = \mathbf{p}(o) + \mathbf{a} \quad (5.1)$$

La méthode de projection de gradient sans les vecteurs d'update va essayer projeter  $\mathbf{z} \in \mathbb{R}^n$  sur l'hyperplan défini par  $\sum_{i=1}^n x_i = 1$  en faisant en sorte que ce soit des probabilités, donc que les  $x_i$  soient positifs et plus petits que 1.

Et pour projeter sans faire en sorte d'avoir des probabilités, nous utilisons le gradient de cet hyperplan qui est un vecteur rempli de 1 et utilise aussi  $\lambda \in \mathbb{R}$  de sorte que  $\mathbf{x}$  soit sur l'hyperplan

$$\mathbf{x} = \mathbf{z} - \lambda \mathbf{1} \quad (5.2)$$

et pour trouver ce  $\lambda$ , nous faisons

$$\sum_i \mathbf{x}_i = \sum_i \mathbf{z}_i - n \cdot \lambda \iff \lambda = \frac{\sum_i \mathbf{z}_i - \sum_i \mathbf{x}_i}{n} \quad (5.3)$$

avec  $\sum_i \mathbf{x}_i = 1$ ,  $\sum_i \mathbf{z}_i = \sum_i \mathbf{p}(o)_i + \mathbf{a}_i = 1 + \sum_i \mathbf{a}_i$  donc nous avons

$$\lambda = \frac{\sum_i \mathbf{a}_i}{n} \quad (5.4)$$

Nous avons donc, en remettant dans 5.2

$$\mathbf{x} = \mathbf{z} - \lambda \mathbf{1} = \mathbf{p}(o) + \mathbf{a} - \frac{\sum_i \mathbf{a}_i}{n} \cdot \mathbf{1} \quad (5.5)$$

Nous pouvons voir ces opérations dans la figure 5.1

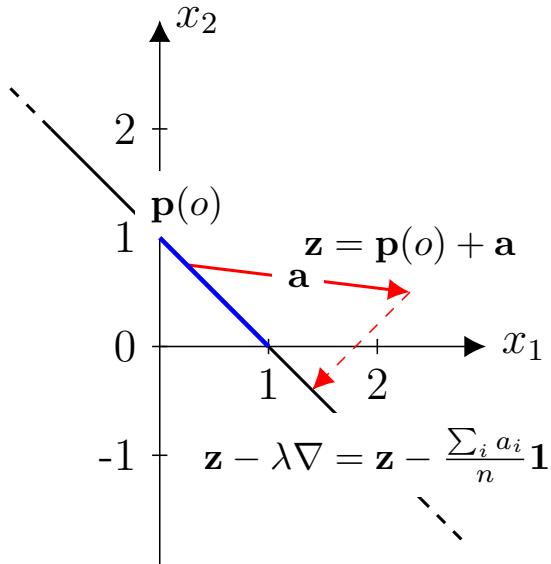


Figure 5.1: Méthode de projection de gradient sur l'hyperplan  $\sum_i x_i = 1$  sans forcément obtenir des probabilités

Comme remarque  $\sum_j \left( \mathbf{a}_j - \frac{\sum_i \mathbf{a}_i}{n} \right) = 0$  donc nous pouvons tout à fait faire

$$\mathbf{x} = \mathbf{p}(o) + \beta \left( \mathbf{a} - \frac{\sum_i \mathbf{a}_i}{n} \cdot \mathbf{1} \right) \quad (5.6)$$

avec  $\beta \leq 1$  et cela serait comme ci nous avions  $\mathbf{a}$  qui aurait une norme  $\beta$  fois celle d'avant. Cela permettrait ainsi de déplacer le  $\mathbf{x}$  pour avoir des probabilités (voir figure 5.2). Il faut aussi que ce  $\beta$  reste le plus proche possible de 1.

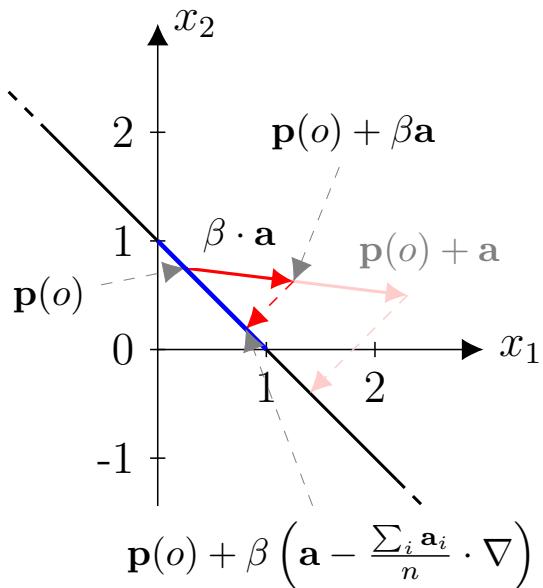


Figure 5.2: Méthode de projection de gradient sur l'hyperplan  $\sum_i x_i = 1$  en forçant l'obtention des probabilités

### 5.1.2 Projection avec les vecteurs d'update

Etant donné que nous ne voulons pas modifier certaines probabilités lors de la mise à jour, il nous faut donc ne faire que la méthode de projection avec un espace de dimension plus petite ou égale à  $n$  et ne pas toucher aux probabilités que nous ne voulons pas modifier. Nous illustrons cela dans la figure 5.3 où  $\mathbf{p}(o)_1$  n'est pas modifié.

Nous pouvons rappeler que nous avons un vecteur qui indique si nous pouvons modifier certaines probabilités ou non dans  $\mathbf{u}(o) \in \{0, 1\}^n$  (ou dans  $\{\text{False}, \text{True}\}^n$ ) mais nous voulons aussi être sûr que les probabilités à 0 ne soient pas diminuées et que les probabilités à 1 ne soient pas augmentées. Pour cela, nous pouvons définir un vecteur de masque  $\mathbf{m} \in \{0, 1\}^n$  où nous aurions les 1 uniquement pour les probabilités que nous voulons modifier.

La formule 5.6 devient alors

$$\mathbf{p}(o) \leftarrow \mathbf{p}(o) + \beta \left( \mathbf{a} \circ \mathbf{m} - \frac{\mathbf{a}^T \mathbf{m}}{\sum_j \mathbf{m}_j} \cdot \mathbf{1} \right) \circ \mathbf{m} \quad (5.7)$$

avec  $\circ$  pour les produits éléments par éléments, produit d'Hadamard. Dans l'implémentation, nous appellons  $\Delta = \left( \mathbf{a} \circ \mathbf{m} - \frac{\mathbf{a}^T \mathbf{m}}{\sum_j \mathbf{m}_j} \cdot \mathbf{1} \right) \circ \mathbf{m}$  et  $step = \beta$  ce qui donnerait

$$\mathbf{p}(o) \leftarrow \mathbf{p}(o) + step \cdot \Delta \quad (5.8)$$

Ces deux dernières formules sont celles que nous utilisons pour la projection du gradient dans nos algorithmes. Nous pouvons à vrai dire simplifier  $\Delta$  avec  $\Delta = \mathbf{a} \circ \mathbf{m} - \frac{\mathbf{a}^T \mathbf{m}}{\sum_j \mathbf{m}_j} \cdot \mathbf{m}$

Nous pouvons vérifier que nous obtenons bien des probabilités théoriquement en vérifiant d'abord si  $\mathbf{p}(o)$  après mise à jour somme encore à 1.

Pour vérifier cela, nous pouvons juste vérifier si  $\sum_k \Delta_k = 0$

$$\sum_k \left( \mathbf{a}_k \cdot \mathbf{m}_k - \frac{\mathbf{a}^T \mathbf{m}}{\sum_j \mathbf{m}_j} \cdot \mathbf{m}_k \right) = \sum_k \mathbf{a}_k \cdot \mathbf{m}_k - \frac{\mathbf{a}^T \mathbf{m}}{\sum_j \mathbf{m}_j} \cdot \sum_k \mathbf{m}_k = 0 \quad (5.9)$$

Il nous reste plus qu'à déterminer/définir  $\beta$  tel que chaque élément soit dans  $[0, 1]$ . Pour cela, nous choisissons de diviser par 2 le step  $\beta$  à chaque fois que la condition n'est pas satisfaite et nous partons de  $\beta = 1$ .

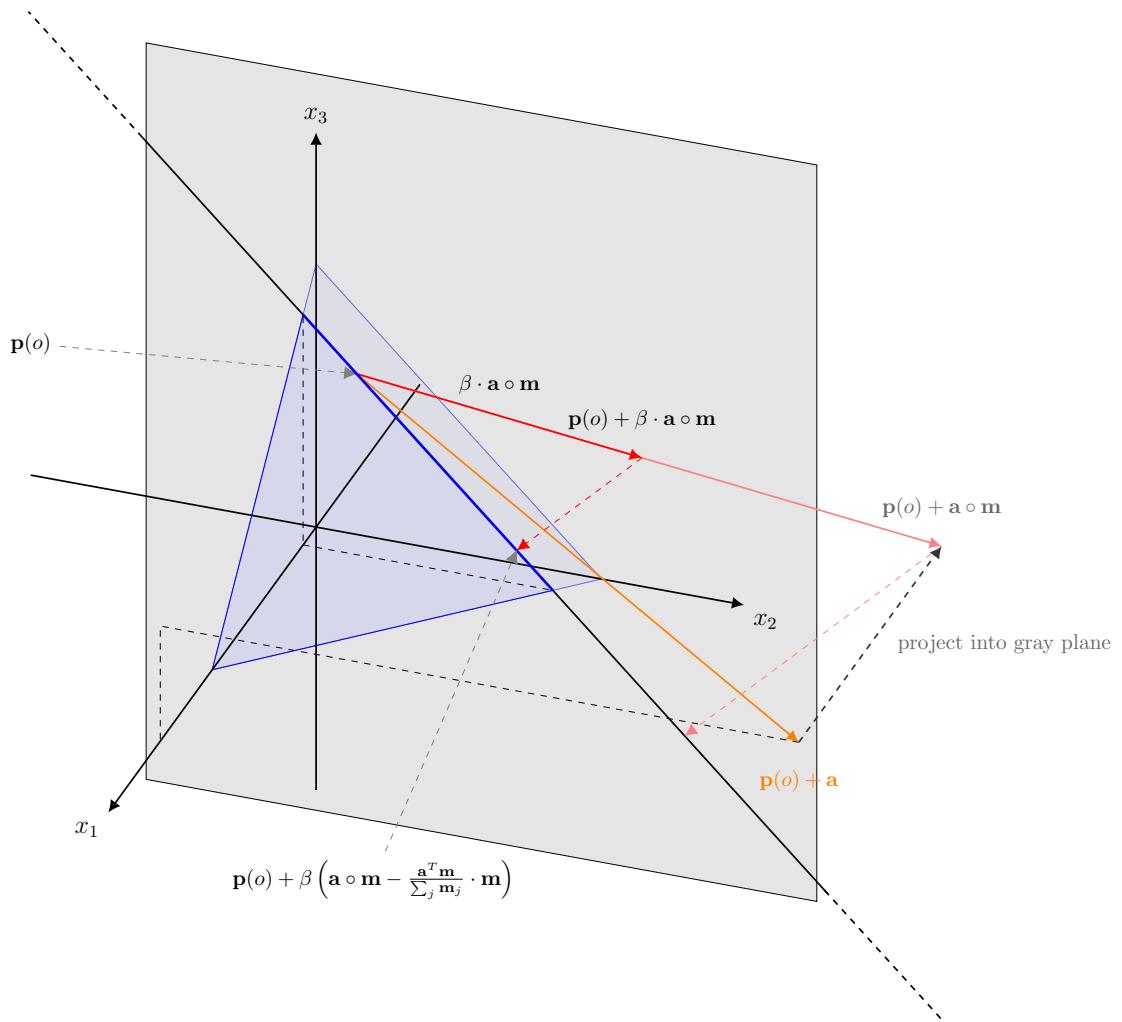


Figure 5.3: Méthode de projection de gradient sur l'hyperplan  $\sum_i x_i = 1$  en forçant l'obtention des probabilités et avec  $\mathbf{u}(o) = [0 \ 1 \ 1]^T$  et ici  $\mathbf{m} = \mathbf{u}(o)$

## 5.2 Renforcer la dernière sous-trajectoire

Dans cette section, nous allons montrer notre première méthode qui consiste à renforcer uniquement la dernière sous-trajectoire des deux agents sans utiliser les récompenses.

Pour nos deux méthodes, la stratégie/politique  $\pi$  est toujours mise à jour à chaque fin d'épisode (voir figure 4.1) et notre première méthode est uniquement basée sur l'historique des transitions des deux agents pour la dernière sous-trajectoire de chaque épisode, i.e  $[(o^{[i]}(t), a^{[i]}(t), r^{[i]}(t))]_{t=0, \dots, T(M)}$ .

Nous modifions les probabilités  $\mathbf{p}(o^{[i]}(t))$  de la stratégie  $\pi$  avec la méthode de projection de gradient avec

$$\mathbf{a} = e_{a^{[i]}(t)+1} \cdot \frac{1}{20 + N(o^{[i]}(t), a^{[i]}(t))} \quad (5.10)$$

à chaque fois que nous voyons  $(o^{[i]}(t), a^{[i]}(t), r^{[i]}(t))$  dans la sous-trajectoire. Nous rappelons que nous modifions la même stratégie pour les deux agents et que  $a^{[i]}(t)+1$  donne l'indice dans  $\{0, 1, 2\}$  associé à l'action  $a^{[i]}(t) \in \{-1, 0, 1\}$ . Par exemple pour l'action  $-1$ , nous avons l'indice 0.

$N(o^{[i]}(t), a^{[i]}(t))$  est le nombre de fois qu'un agent est passé sur l'arc entre  $o^{[i]}(t)$  et  $a^{[i]}(t)$ . Cela veut aussi dire le nombre de fois que l'action  $a^{[i]}(t)$  a été prise depuis l'observation  $o^{[i]}(t)$  ou son noeud correspondant. Dans l'implémentation, nous gardons ces valeurs dans les noeuds de l'arbre et nous l'appelons  $nbA$ . Ce nombre est incrémenté de 1 à chaque fois qu'un agent passe dessus et n'est jamais remis à 0, même entre sous-trajectoires et entre épisodes. Cela a pour effet de diminuer l'impact de la mise à jour pour les chemins explorées plus souvent mais les modifications deviennent faible très rapidement.

De plus, pour essayer d'améliorer la convergence, nous donnons à l'algorithme à la fois la dernière sous-trajectoire des deux agents mais aussi sa symétrique dans l'arbre, autrement dit, la même sous-trajectoire mais en multipliant toutes les actions par  $-1$ .

### 5.2.1 Un désavantage de nos méthodes et introduction d'un taux d'exploration $\epsilon$

Comme remarque assez importante, nos deux méthodes sont *on-policy*<sup>2</sup> et peuvent être coincées dans l'apprentissage car des probabilités peuvent être à 0 ou à 1. Cela signifie que l'agent ne ferait plus que passer par certains chemins sans explorer d'autres. Pour essayer de contrer ce problème, nous avons ajouté un taux  $\epsilon \in [0, 1]$  pour la probabilité de choisir uniformément une action parmi celles que nous pouvons modifier en regardant  $\mathbf{u}(o)$ . Cette probabilité  $\epsilon$  vaut  $\epsilon_{init} \in [0, 1]$  au départ puis diminue de moitié à chaque épisode pour faire décroître le taux d'exploration et comme remarque, si nous utilisons  $\epsilon \in (0, 1)$ , nous n'aurions plus *on-policy* car la stratégie apprise (*target policy*) n'utiliserait plus ce choix uniforme (dans la *behavior policy*). A la place de *on-policy*, cela serait *off-policy*.

Une autre solution possible que ce taux d'exploration et en restant *on-policy* serait de limiter les stratégies (directement la *target policy* avec *target policy*=*behavior policy*)

---

<sup>2</sup>voir Sutton and Barto 2018. Pour *on-policy*, la stratégie apprise *target policy* est la même que la stratégie utilisée lors de la simulation *behavior policy*.

*policy*) pour qu'il n'y ait plus de probabilité exactement à 0 ou à 1 (voir Sutton and Barto 2018 dans le chapitre policy gradient) pour les probabilités que nous pouvons modifier avec  $\mathbf{u}(o)$  bien sûr. Un désavantage de cette solution serait que nous ne pouvons plus converger vers une stratégie avec des probabilités à 1 (par exemple dans l'arbre).

---

**Algorithm 2:** Choosing actions for the two agents in one time step of a sub-trajectory with  $\epsilon$  probability of exploring,  $1 - \epsilon$  of following  $\pi$

---

```

input :  $\mathbf{u}(o^{[0]}), \mathbf{u}(o^{[1]}), \pi(\cdot|o^{[0]}), \pi(\cdot|o^{[1]}), \epsilon \in [0, 1)$ 
output:  $\mathbf{p}_0, \mathbf{p}_1$  the probability vectors over actions for the behavior policy
doRandomAction =  $u \sim \mathcal{U}(0, 1) \leq \epsilon$ ;
if  $\sum_i \mathbf{u}(o^{[0]})_i > 0$  and  $\sum_i \mathbf{u}(o^{[1]})_i > 0$  then
     $\mathbf{p}_0 \leftarrow (1 - \text{doRandomAction}) \cdot \pi(\cdot|o^{[0]}) + \frac{\text{doRandomAction}}{\sum_i \mathbf{u}(o^{[0]})} \cdot \mathbf{u}(o^{[0]})$ ;
     $\mathbf{p}_1 \leftarrow (1 - \text{doRandomAction}) \cdot \pi(\cdot|o^{[1]}) + \frac{\text{doRandomAction}}{\sum_i \mathbf{u}(o^{[1]})} \cdot \mathbf{u}(o^{[1]})$ ;
end

```

---

Ce taux d'exploration  $\epsilon$  était inspiré d' $\epsilon$ -greedy dans les algorithmes d'apprentissage par renforcement basées sur les fonctions de valeurs uniquement. Cette méthode n'est pas la seule pour le *trade-off* entre exploiter les connaissances de l'agent avec ses fonctions de valeurs et explorer l'environnement afin de découvrir potentiellement de meilleures actions et états.

**Dans ces algorithmes**, il y a très souvent un argmax (la partie gloutonne des algorithmes) par rapport aux actions. Un argmax de la fonction de valeur état-action notée  $q(s, a)$ <sup>3</sup> qui est une estimation/approximation de  $q^\pi(s, a)$  qui est la récompense moyenne<sup>4</sup> future cumulée (avec ou sans *discount factor*) en partant de l'état courant  $s$  et en prenant une action  $a$ . Donc nous avons une formule de la forme ci-dessous pour le choix d'action de façon gloutonne avec probabilité  $1 - \epsilon$  à partir d'un état  $s \in \mathcal{S}$  et  $\mathcal{A}$  est l'ensemble des actions possibles depuis l'état.

$$\operatorname{argmax}_{a \in \mathcal{A}} q(s, a) \quad (5.11)$$

L'étape ci-dessus serait liée à celle de *policy improvement* où  $\pi \rightsquigarrow \text{greedy}(q)$  (voir la chapitre de programmation dynamique dans Sutton and Barto 2018 avec *Generalized Policy Iteration*) et l'évaluation de la politique courante est faite en essayant de faire converger la fonction de valeur vers la vraie, par exemple faire tendre  $q(s, a)$  vers  $q^\pi(s, a)$  ( $q(s, a) \rightsquigarrow q^\pi(s, a)$ ) pour bien évaluer la stratégie  $\pi$ .

Voici ci-dessous à quoi ressemblerait la stratégie  $\pi$  pour la partie gloutonne si jamais nous prenions  $q^\pi(s, a)$  au lieu de l'estimation.

$$\operatorname{argmax}_{a \in \mathcal{A}} q^\pi(s, a) = \operatorname{argmax}_{a \in \mathcal{A}} \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \quad (5.12)$$

$G_t = R_{t+1} + G_{t+1}$  est le *return* et c'est la variable aléatoire de la récompense future cumulée (ici, elle n'est pas pondérée avec un *discount factor*  $\gamma$ ). C'est possible

<sup>3</sup>Intuitivement, cette fonction donne une estimation de à quelle point c'est bien de prendre une action depuis un état avec la politique courante  $\pi$

<sup>4</sup>Moyenne en suivant la politique  $\pi$  pour le choix des actions à partir des états et dynamique de l'environnement  $\mathbb{P}(s', r|s, a)$  qui indique les états et les récompenses obtenues après avoir pris une action depuis un état.

de rajouter l'utilisation de  $\gamma$  dans les cas épisodiques mais dans les cas continus, nous serions forcés<sup>5</sup> de l'utiliser pour obtenir un return  $G_t$  fini (le cas continu est opposé aux cas épisodiques, voir Sutton and Barto 2018)).

Comme remarqué, les fonctions de valeur d'état-action  $q$  sont utilisées au lieu des fonctions de valeur d'états  $v$  pour éviter de connaître la dynamique de l'environnement (les probabilités) dans les méthodes Monte-Carlo ou Temporal-Difference qui peuvent être vues comme des méthodes basées sur des échantillons de trajectoires au lieu de tout calculer explicitement avec la programmation dynamique si la dynamique de l'environnement est connue. Les méthodes en programmation dynamique peuvent donc être infaisable soit car la dynamique de l'environnement est inconnue, soit elle est connue mais trop lourde en complexité.

Ces algorithmes basées sur les fonctions de valeurs uniquement ne peuvent apprendre que des stratégies déterministes ou quasi-déterministes (*Greedy in the Limit with Infinite Exploration* par exemple où le taux d'exploration décroît afin d'obtenir des stratégies déterministes à la limite.) et donc il faut balancer entre vouloir prendre tout le temps l'action menant à un meilleur<sup>6</sup> état et entre explorer d'autres actions d'où les algorithmes  $\epsilon$ -greedy.

Dans notre problème, ces algorithmes basées sur les fonctions de valeurs uniquement ne sont pas applicables car nous savons, par connaissance du problème que la stratégie optimale ne peut être que stochastique. Cependant, nous réutilisons tout de même cette idée d'avoir un taux d'exploration pour les causes citées précédemment.

---

<sup>5</sup>David Silver, dans son cours, explique que c'est la solution la plus simple

<sup>6</sup>meilleur: c'est-à-dire en terme de récompenses moyennes futures cumulées

## 5.3 REINFORCE-like

Dans cette section, nous allons montrer notre deuxième méthode inspirée de l'algorithme Monte-Carlo Policy Gradient autrement appelé REINFORCE qui est un algorithme d'apprentissage par renforcement qui fait apprendre la politique stochastique directement et ne fait la mise à jour qu'à la fin de chaque épisode.

Notre méthode utilise tout l'historique des transitions ainsi que les récompenses contrairement à notre première méthode et nous modifions les probabilités  $\mathbf{p}(o^{[i]}(t))$  de la stratégie  $\pi$  avec la méthode de projection de gradient avec

$$\mathbf{a} = e_{a^{[i]}(t)+1} \cdot \frac{\text{lr} \cdot G^{[i]}(t)}{\max(0.1, \mathbf{p}(o^{[i]}(t))_{a^{[i]}(t)+1})} \quad (5.13)$$

où  $a^{[i]}(t) + 1$  donne l'indice dans  $\{0, 1, 2\}$  associé à l'action  $a^{[i]}(t) \in \{-1, 0, 1\}$ . Par exemple pour l'action  $-1$ , nous avons l'indice  $0$ . Le maximum est utilisé pour éviter que la modification soit trop grande et le lr pour learning rate pour essayer de contrôler le poids de la modification.  $G^{[i]}(t)$  est le return de l'agent  $i$  (Comme remarque, les récompenses ont un indice décalé de 1 comparé au livre de Sutton (voir page 48 et 54 de Sutton and Barto 2018)).

$$G^{[i]}(t) = \sum_{k=t}^{T-1} R^{[i]}(k) \quad t = 0, \dots, T-1 \quad (5.14)$$

$$G^{[i]}(t) = R^{[i]}(t) + G^{[i]}(t+1), \quad t = 0, \dots, T-1, \quad G^{[i]}(T) = 0 \quad (5.15)$$

Nous avons mis en majuscule les récompenses car cette fois ci, ce sont les récompenses pendant toute la trajectoire, ce n'est plus une seule sous-trajetoire, les indices partent de  $0$  jusqu'à la fin de l'épisode. La dernière action commence au temps  $T-1$  avec  $T = t_{tot}$ .

Dans l'implémentation, nous avons choisi manuellement  $\text{lr} = \frac{1}{2} \cdot 10^{-5}$  en ayant essayé quelques autres valeurs par puissance de  $10$  mais cela pouvait faire exploser le temps moyen de rencontre. De plus, nous verrons que nous n'arrivons pas à obtenir la stratégie avec temps de rencontre moyen de  $7$  pour  $N = 2$ .

### 5.3.1 Avantages de policy gradient contre value-based

Les algorithmes basées sur les fonctions de valeurs uniquement ne peuvent apprendre que des stratégies déterministes ou quasi-déterministes (*Greedy in the Limit with Infinite Exploration* par exemple où le taux d'exploration  $\epsilon$  dans  $\epsilon$ -greedy décroît afin d'obtenir des stratégies déterministes à la limite.) comme expliqué précédemment.

Par connaissance du problème, nous savons que la stratégie optimale ne peut être que stochastique et nous ne voulons donc pas utiliser des algorithmes *value-based*.

Les algorithmes *policy-based* ou utilisant à la fois les fonctions de valeurs et la politique explicitement (*actor-critic*) peuvent faire apprendre des stratégies probabilistes et c'est la raison pourquoi nous avons choisi de nous inspirer de l'algorithme REINFORCE.

# Chapitre 6

## Résultats

Dans ce chapitre, nous allons premièrement montrer comment nous estimons les temps moyen de rencontre à travers la simulation.

Deuxièmement, nous présenterons des graphes montrant le temps moyen de rencontre estimé pour différents hyperparamètres ( $K, M, N$  par exemple) pour les stratégies avec les deux phases, sans les deux phases et avec la méthode de renforcer la dernière sous-trajectoire ou REINFORCE-like. En plus de ces graphes, nous montrons bien sûr quelles stratégies nous avons obtenues.

De nos résultats que nous allons montrer dans ce chapitre, nous pourrons observer que c'est compliqué d'analyser numériquement. Nos méthodes peuvent donner différentes stratégies en lançant plusieurs fois l'algorithme et le temps de rencontre peut fluctuer beaucoup. De plus, nous ne savons pas si la solution est unique ou non.

Dans ce projet, nous n'avons pas essayé d'analyser la convergence des algorithmes (aussi combiné à la méthode de projection de gradient) et pas utilisé de méthodes rigoureuses pour comparer la performance des algorithmes.

### 6.1 Estimation du temps moyen de rencontre

#### 6.1.1 Estimation du temps moyen de rencontre lors des simulations sans mise à jour des stratégies

Pour les simulations sans mise à jour de la stratégie, nous utilisons juste une moyenne empirique implémentée de façon incrémentale car la politique/stratégie ne change pas entre les épisodes (voir Robbins-Monro pour plus de détails sur les conditions dans Gosavi 2014).

$$E \leftarrow E + \alpha_k \cdot (m_k - E) = (1 - \alpha_k) \cdot E + \alpha_k \cdot m_k \quad (6.1)$$

où  $E$  est l'estimation du temps moyen de rencontre,  $\alpha_k = \frac{1}{k}$ ,  $k = 1, \dots, N$  où  $N = \#$  épisodes et  $m_k$  est le temps de rencontre pour l'épisode  $k - 1$ .

La formule est obtenue en décomposant la moyenne empirique

$$E_N = \frac{1}{N} \sum_{k=1}^N m_k = \frac{1}{N} \cdot m_N + \frac{1}{N} \frac{N-1}{N-1} \cdot \sum_{k=1}^{N-1} m_k = \frac{1}{N} \cdot m_N + \left(1 - \frac{1}{N}\right) \cdot E_{N-1} \quad (6.2)$$

### 6.1.2 Estimation du temps moyen de rencontre lors des simulations avec mise à jour des stratégies

Pour les simulations avec mise à jour de la stratégie, nous utilisons une autre formule car les stratégies changent entre épisodes. Généralement, pour une estimation d'une quantité non stationnaire,  $\alpha_k$  est une constante  $\alpha$  et cela donne une moyenne mouvante. Pour  $\alpha$  proche de 1, cette moyenne mouvante ne prend en compte que très peu d'échantillons (par échantillons, nous pouvons prendre plusieurs  $m_k$  par exemple) dans le passé ( $\alpha = 1$  donne une estimation basée uniquement sur le dernier  $m_k$ ) et donc plus le  $\alpha$  est petit, plus le passé est pris en compte pour chaque estimation.

Pourquoi avoir un  $\alpha_k$  constant ? Intuitivement, ne pas avoir un  $\alpha_k$  qui décroît toujours permet de s'adapter plus rapidement quand les stratégies changent (surtout si le temps moyen de rencontre change beaucoup quand les stratégies changent).

Par exemple lorsque  $\alpha_k$  est trop petit et que le temps moyen de rencontre exact change soudainement, l'estimation du temps moyen de rencontre est très mauvaise car  $\alpha_k$  étant petit, la mise à jour de l'estimation est très faible. Cependant, si le temps moyen de rencontre ne change pas trop pour  $\alpha_k$  petit ( $k$  grand), l'estimation du temps moyen de rencontre fluctuerait plus que le vrai temps moyen de rencontre. Cela est le cas pour nos stratégies mais c'est aussi une indication que le temps moyen de rencontre exact ne saute pas dans tous les sens quand nous faisons la mise à jour de nos stratégies et donc que cela pourrait nous montrer que les stratégies sont apprennables.

**Formules:** Pour nos estimations, nous utilisons une méthode dans le livre de Sutton and Barto 2018 à la page 35 qui permet d'enlever l'impact de l'estimation initiale du temps moyen de rencontre et à la fois avoir des avantages d'une estimation avec  $\alpha_k$  constant pour des problèmes non stationnaires.

Notons  $\bar{o}_0 = 0$  et soit  $\alpha \in (0, 1]$ . Nous définissons

$$\bar{o}_k \leftarrow \bar{o}_{k-1} + \alpha \cdot (1 - \bar{o}_{k-1}) \quad (6.3)$$

et  $\beta_k = \frac{\alpha}{\bar{o}_k}$  avec  $k = 1, \dots, N$ . Nous pouvons déjà observer que  $\beta_1 = 1$  et  $\bar{o}_1 = \alpha$ . Cela permettra ainsi d'enlever l'impact de l'estimation initiale du temps moyen de rencontre comme nous allons le montrer prochainement.

Nous utilisons cette formule ci-dessous pour l'estimation du temps moyen de rencontre

$$E \leftarrow E + \beta_k \cdot (m_k - E) \quad (6.4)$$

où  $\beta_k$  finit par tendre vers le *constant step size*  $\alpha$  (et  $\bar{o}_k$  finit par tendre vers 1.)

Puisque  $\beta_1 = 1$ , nous avons que

$$E_1 = E_0 + 1 \cdot (m_1 - E_0) = m_1 \quad (6.5)$$

et donc  $E_0$  n'est plus pris en compte. Dans notre projet, nous allons utiliser arbitrairement  $\alpha = 10^{-3}$

**Moyenne pondérée des échantillons passés:** Nous rajoutons juste mathématiquement les formules pour soutenir notre explication intuitive.

$$\begin{aligned}
 E_N &= E_{N-1} + \beta_N \cdot (m_N - E_{N-1}) = \beta_N m_N + (1 - \beta_N) \cdot E_{N-1} \\
 &= \beta_N m_N + (1 - \beta_N) \cdot [\beta_{N-1} m_{N-1} + (1 - \beta_{N-1}) \cdot E_{N-2}] \\
 &= \sum_{k=1}^N \beta_k m_k \prod_{k'=k+1}^N (1 - \beta_{k'}) + \prod_{k''=1}^N (1 - \beta_{k''}) \cdot E_0
 \end{aligned} \tag{6.6}$$

Si  $\beta_k = \beta$  constant, nous obtenons

$$E_N = \sum_{k=1}^N \beta m_k (1 - \beta)^{N-k} + (1 - \beta)^N \cdot E_0 \tag{6.7}$$

et donc nous pouvons observer que l'impact du passé est faible pour  $\beta$  proche de 1 et que l'estimation est biaisée par  $E_0$  qui ne provient pas de la distribution qui génère les  $m_k$ .

Si  $\beta_k = \frac{\alpha}{\bar{o}_k}$  avec  $\beta_1 = 1$  en utilisant les formules que nous avons définies plus tôt, nous obtenons

$$\begin{aligned}
 E_N &= \sum_{k=1}^N \beta_k m_k \prod_{k'=k+1}^N (1 - \beta_{k'}) \\
 &= \sum_{k=1}^N \frac{1}{\bar{o}_k} \alpha m_k \prod_{k'=k+1}^N \left(1 - \frac{1}{\bar{o}_k} \alpha\right)
 \end{aligned} \tag{6.8}$$

avec  $\bar{o}_k \rightarrow 1$  et  $\beta_k \rightarrow \alpha$  donc nous aurions une moyenne mouvante *exponential recency-weighted average* lorsque  $N \rightarrow \infty$  sans biais initial causé par  $E_0$ .

## 6.2 Stratégies avec deux phases

### 6.2.1 Renforcer dernière sous-trajectoire

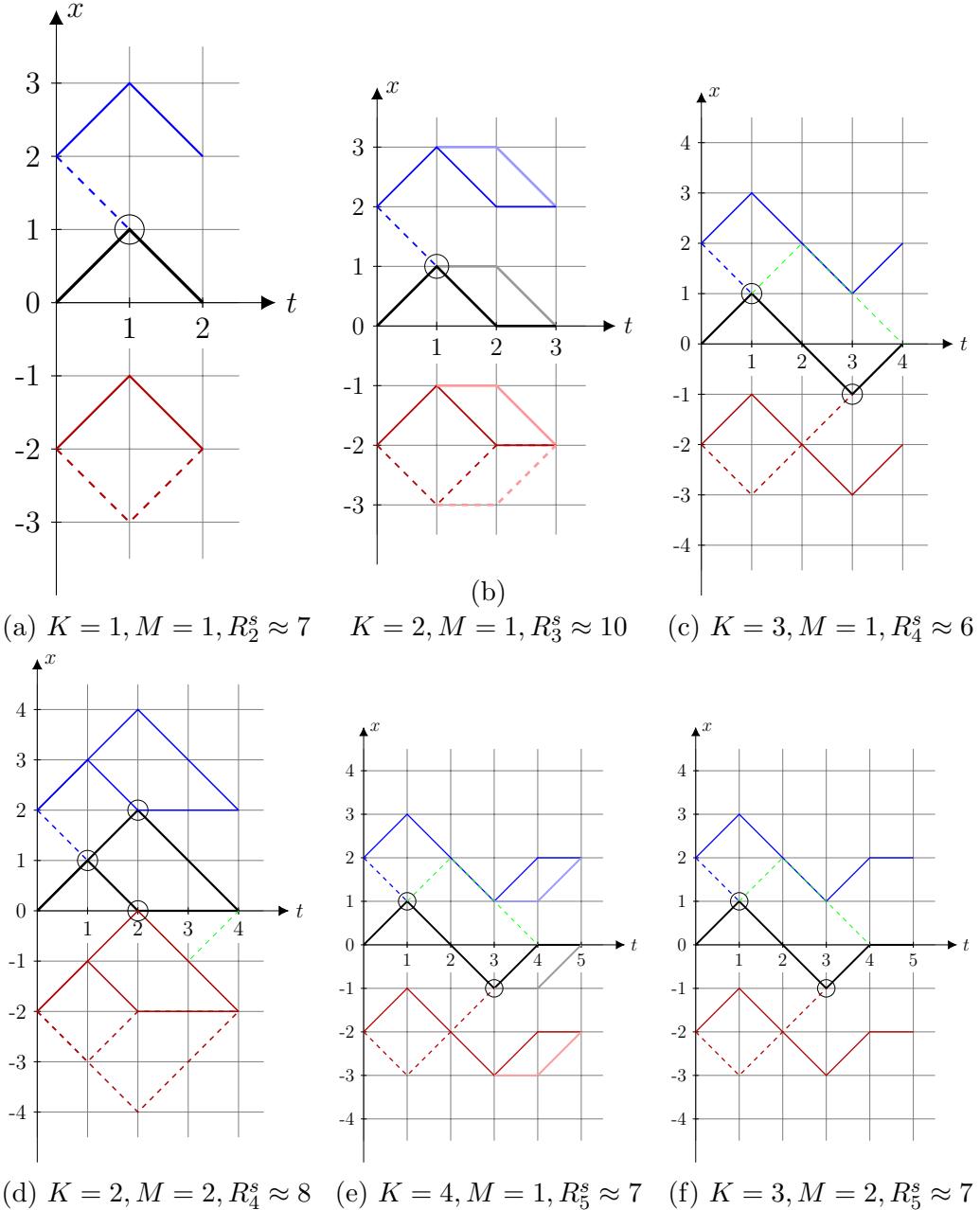


Figure 6.1: 6 stratégies avec deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement.  $\epsilon = 0, 10001$  épisodes

Dans les illustrations, toutes les fois où nous utilisons vert traitillé, cela sera pour souligner une stratégie de l'article. Par exemple, dans la figure 6.1d, la première action verte est impossible à apprendre pour les stratégies avec deux phases pour  $K = 2, M = 2$  car nous forçons les agents de revenir à leur position initiale directement dans la deuxième phase après  $K = 2$  pas. Cependant cela est possible de l'obtenir pour nos stratégies sans les deux phases.

Nous pouvons remarquer que dans 6.1b les chemins plus clairs sont inutiles car

il n'y a jamais de rencontre en les faisant (et le temps de rencontre moyen reste inchangé), il peut donc y avoir plusieurs stratégies avec le même temps de rencontre ! De plus, à cause de la borne, cela est impossible d'apprendre la stratégie d'Alpern.

Pour  $K = 3, M = 1$ , dans la figure 6.1c, à cause de la borne qui est trop petite, nous n'arrivons pas à obtenir le morceau vert mais on obtient l'autre chemin que 6.1d n'a pas.

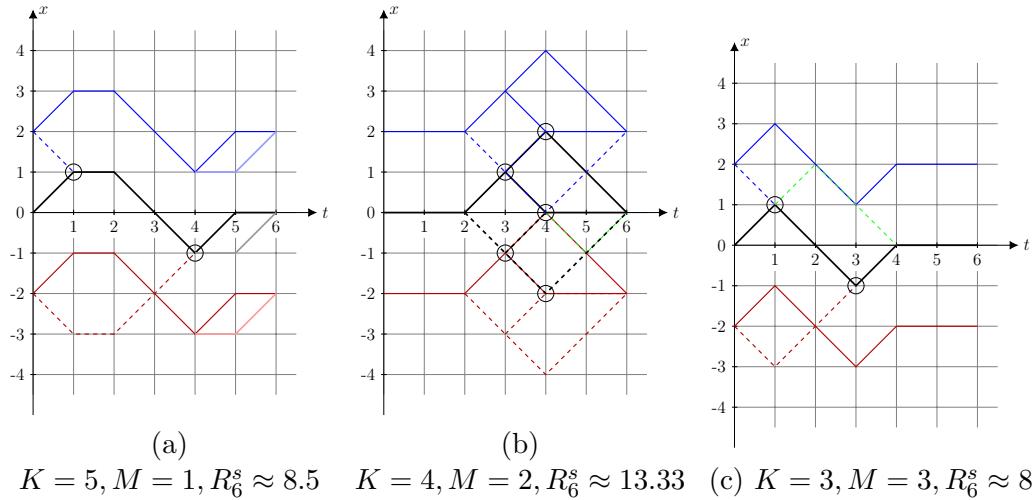


Figure 6.2: 3 stratégies avec deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement.  $\epsilon = 0$ , 10001 épisodes

Nous pouvons observer que des stratégies essaient de se rapprocher des stratégies de l'article mais soit cela converge mal, soit les stratégies que nous voulons sont en dehors des stratégies obtenables pour des  $K$  et  $M$  donnés.

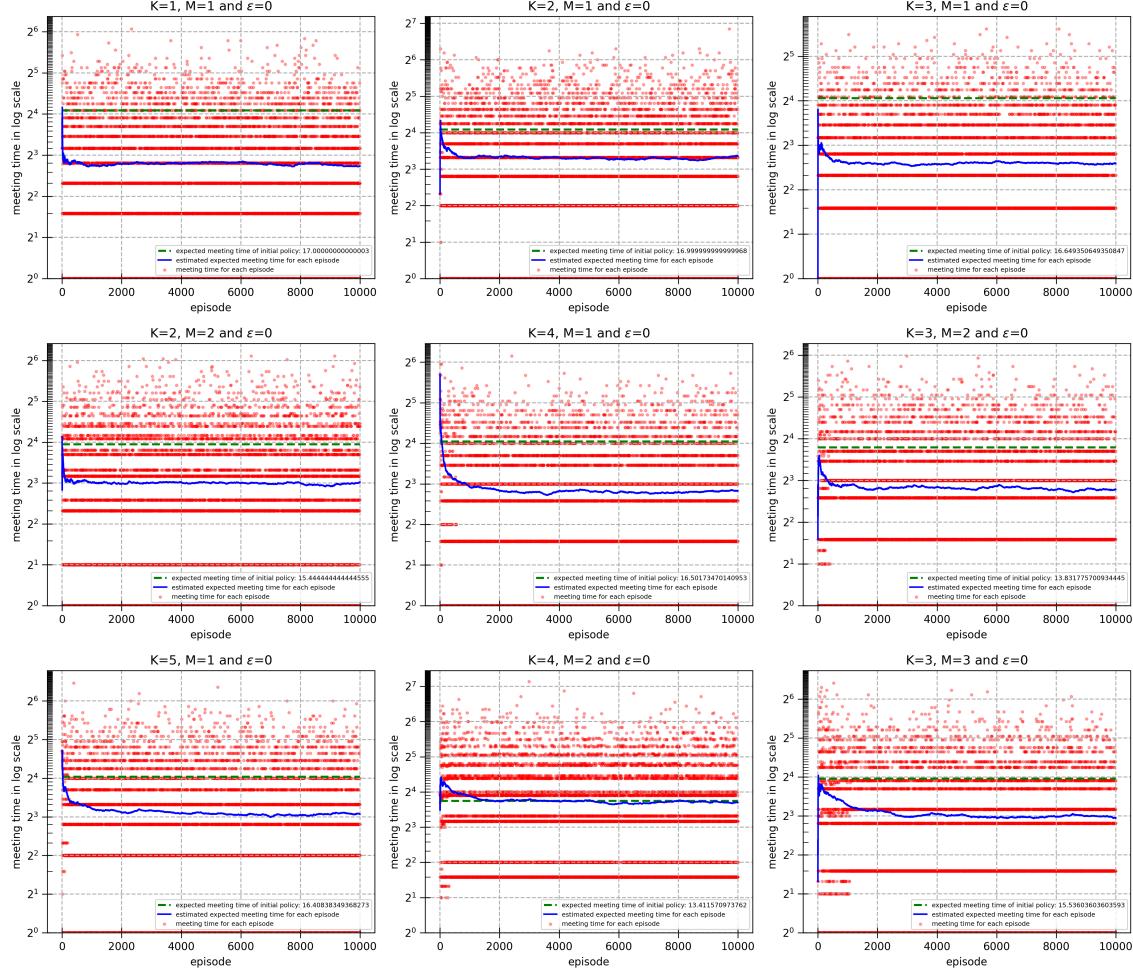


Figure 6.3: Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies avec deux phases avec mise à jour en renforçant la dernière sous-trajetoire

Les courbes bleues dans la figure 6.3 représentent l'estimation du temps moyen de rencontre avec une politique/stratégie qui est différente pour chaque épisode (voir 6.1 pour les détails de comment nous avons fait l'estimation). Ces courbes sont juste des estimations basées sur les temps de rencontre à chaque épisode (points rouges), les vrais temps moyen de rencontre fluctuent moins (voir figure 6.4 juste avec pour certains  $K$  et  $M$  car calculer le temps moyen de rencontre exact est lourd en complexité). Les droites vertes indiquent le temps moyen de rencontre exact pour les stratégies initiales.

Nous pouvons aussi observer dans les graphes de la figure 6.3 que le temps de rencontre varie extrêmement beaucoup.

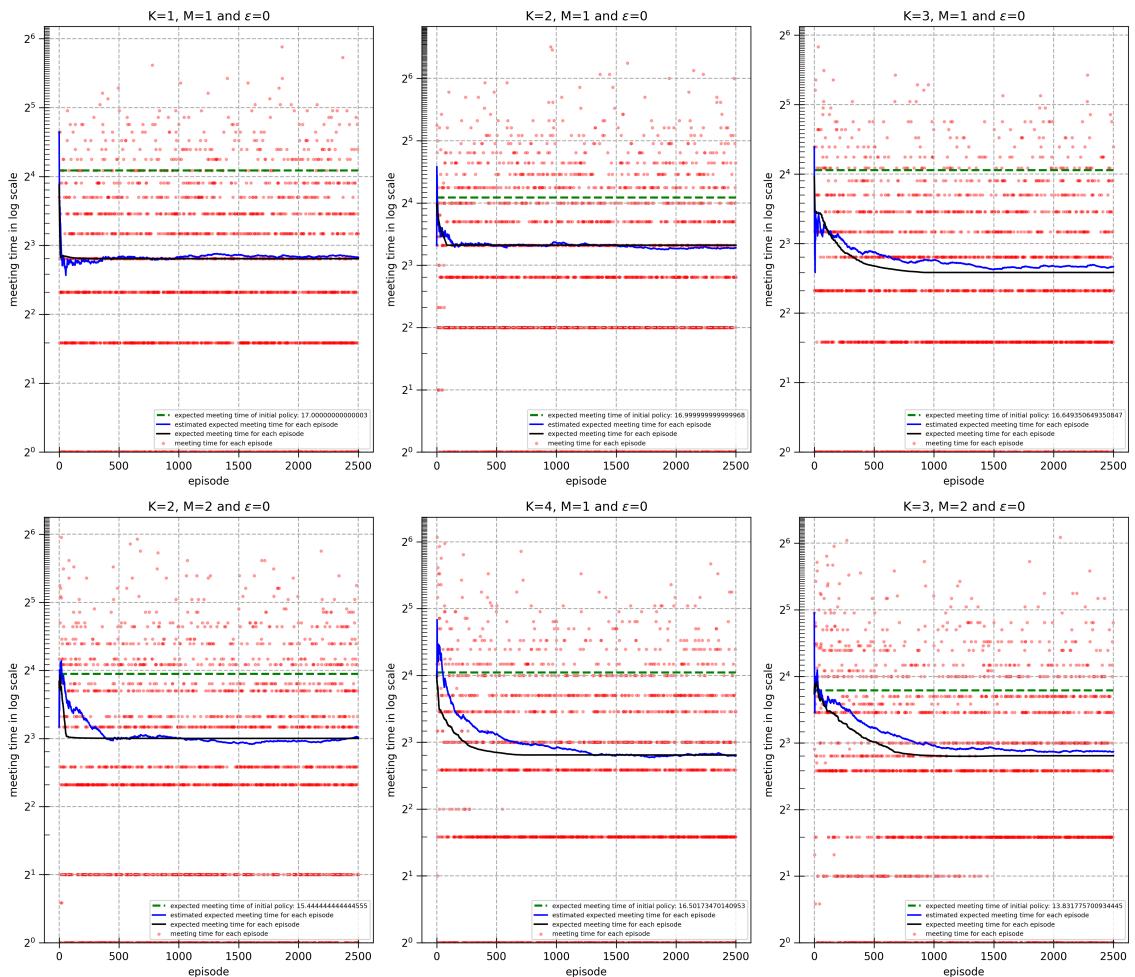


Figure 6.4: Temps moyen de rencontre exact comparé à celles estimées pour 2501 épisodes

La figure 6.5 montre que nous pouvons obtenir différentes stratégies en lançant plusieurs simulations.

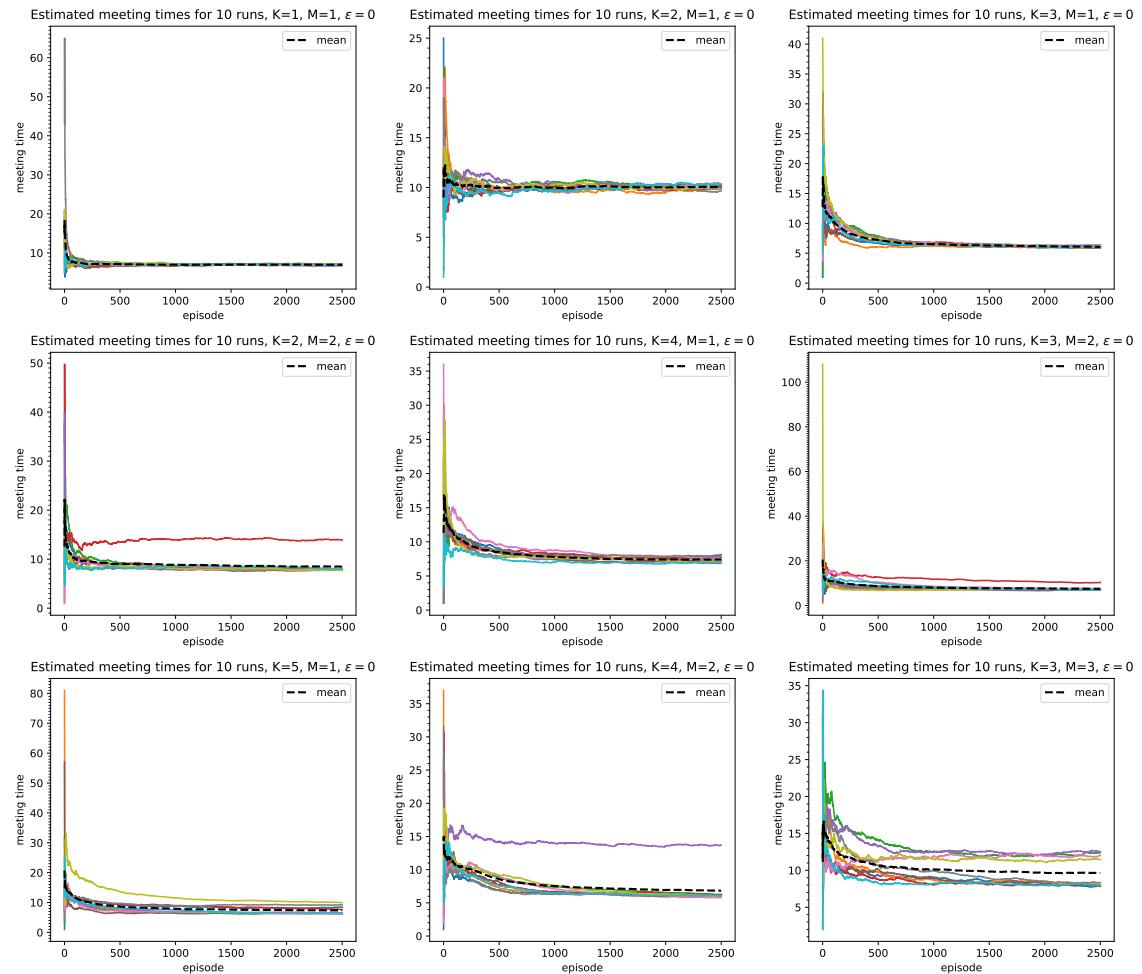


Figure 6.5: 10 simulations pour les stratégies avec les deux phases avec 2501 épisodes,  $\epsilon = 0$ , mise à jour en renforçant la dernière sous-trajectoire uniquement

## 6.3 Stratégies sans deux phases

### 6.3.1 Renforcer dernière sous-trajectoire

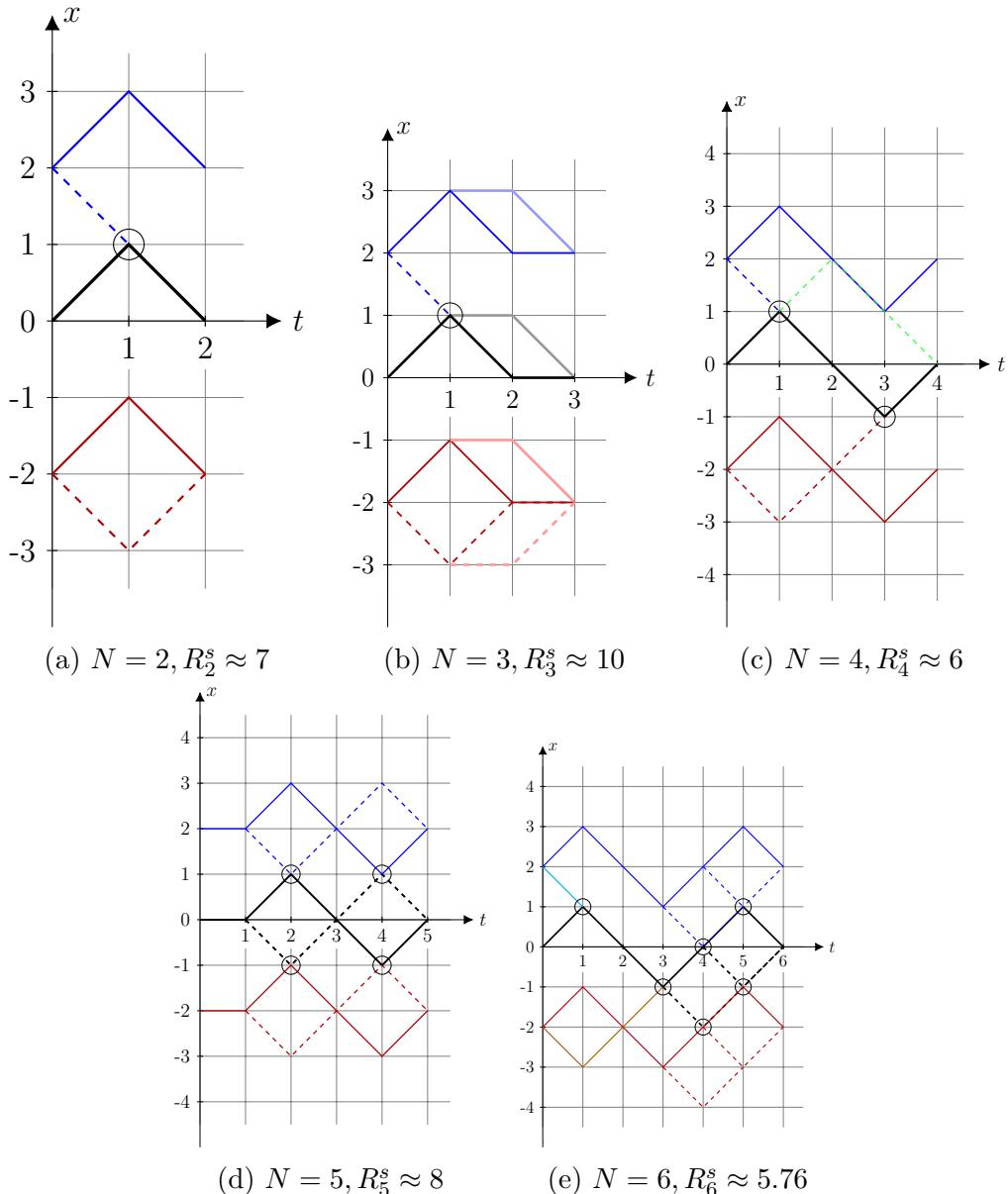


Figure 6.6: 5 stratégies sans deux phases obtenues avec mise à jour des probabilités en renforçant la dernière sous-trajectoire uniquement.  $\epsilon = 0$ , 10001 épisodes

La même stratégie que celle de l'article pour  $N = 4$  est obtenable mais nous ne l'avons pas obtenu ici.

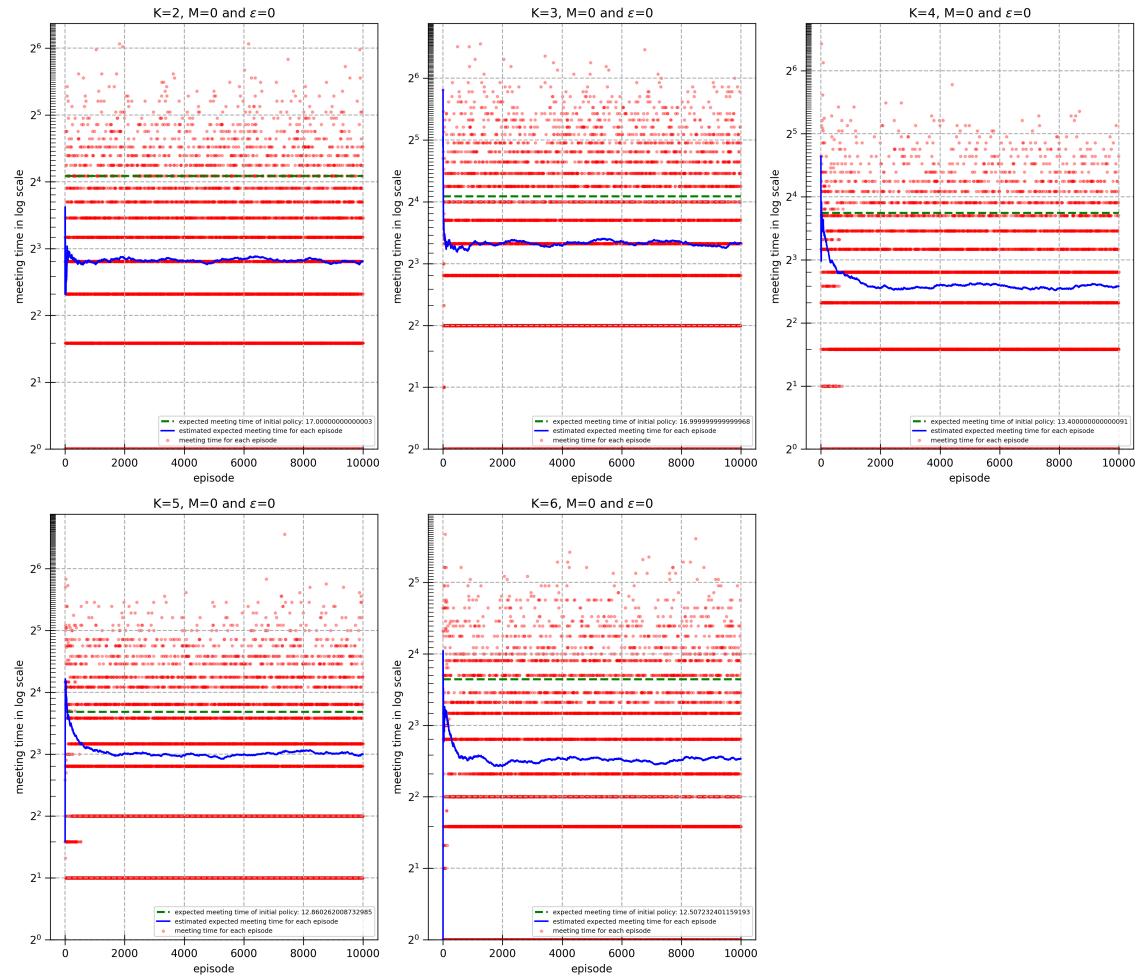


Figure 6.7: Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies sans deux phases avec mise à jour en renforçant la dernière sous-trajectoire

La figure 6.8 nous montre que nous pouvons obtenir différentes stratégies en lançant plusieurs simulations.

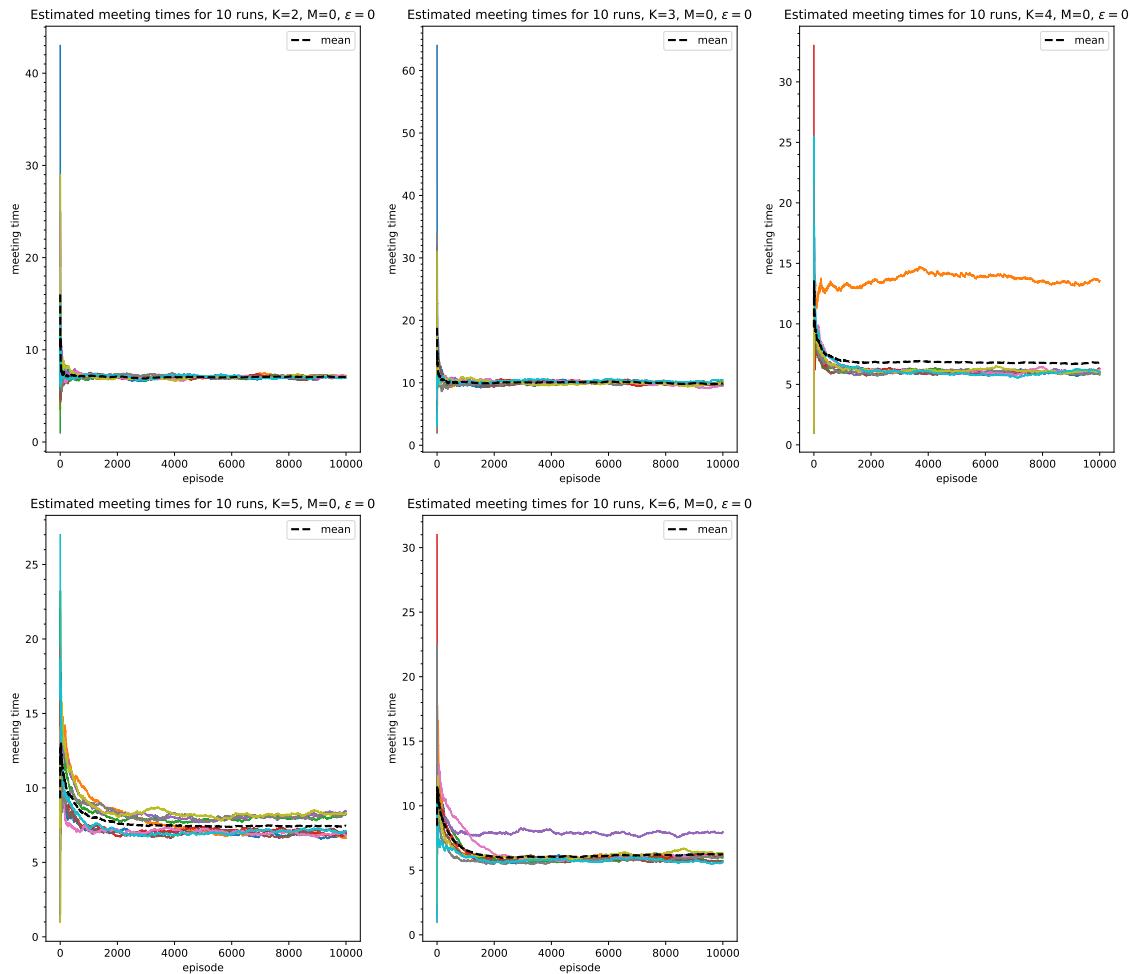


Figure 6.8: 10 simulations pour les stratégies sans les deux phases avec 10001 épisodes,  $\epsilon = 0$ , mise à jour en renforçant la dernière sous-trajectoire uniquement

### 6.3.2 REINFORCE-like

Pour cette méthode, nous ne montrons pas les stratégies comme que précédemment car nous ne verrions presque plus rien à cause de probabilités qui ne convergent pas très bien.

Après 50001 épisodes, pour  $N = 2$  et  $N = 3$ , l'algorithme essaie de converger vers la même stratégie que nous avons obtenu quand nous avions modifier les stratégies en renforçant uniquement dernière sous-trajectoire mais peine un peu à rendre des probabilités exactement à 0 (par exemple 0.07638655 ou 0.09707489 pour l'action de ne rien faire dans le premier pas de temps pour  $N = 2$  et  $N = 3$  respectivement). Le temps moyen de rencontre exact pour  $N = 2$  est de 8.378 environ (au lieu d'être proche de 7) et celui pour  $N = 3$  est de 11.1097 environ (au lieu de 10). Le temps moyen de rencontre exact semble diminuer lentement et/ou osciller plus que notre autre méthode. REINFORCE-like a une grande variance et nous pouvons l'observer à travers les temps de rencontre pour chaque épisode et cela a un impact direct sur le return qui varie beaucoup mais nous ne l'avons pas estimé. De plus, si l'agent prend une action qui a une probabilité faible, cela peut faire exploser la modification (de l'ordre de centaines de fois  $2^6 \cdot \frac{1}{0.1} = 640$  si le learning rate n'était pas pris en compte).

Pour  $N = 4$ , nous avons environ 8.5138 comme temps moyen de rencontre exact au lieu de 6 ou environ 5.9441 de l'article. Si nous enlevons les arcs avec probabilité inférieure à 0.098, nous obtenons une stratégie proche de celle de l'article visuellement.

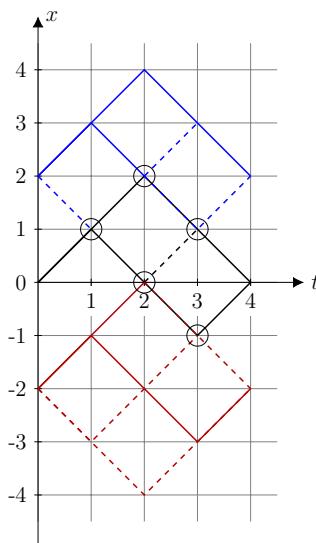


Figure 6.9: Illustration de la stratégie sans deux phases avec mise à jour par la méthode REINFORCE-like avec  $N = 4$ ,  $\epsilon = 0$ , 50001 épisodes et sans les arcs qui ont des probabilités plus faibles que 0.098

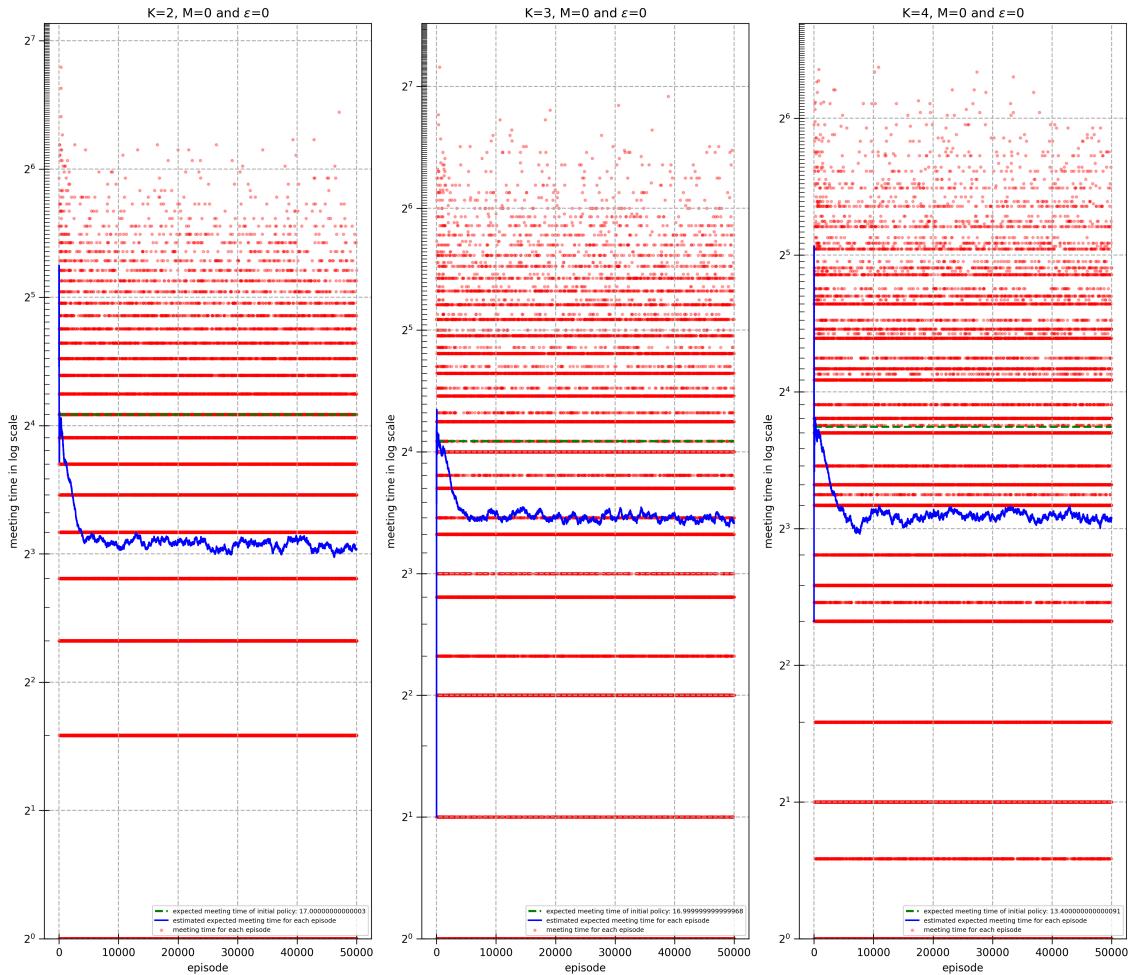


Figure 6.10: Temps moyen de rencontre estimé et temps de rencontre pour chaque épisode pour des stratégies sans deux phases avec mise à jour avec REINFORCE-like

# Chapitre 7

## Conclusion

### 7.1 Futures améliorations possibles

Nos stratégies forçant les agents à revenir à leur positions initiales entre chaque sous-trajectoire fait partie des stratégies bornées spatialement et nos stratégies font aussi parties des stratégies "*distance-preserving*". Une stratégie est "*distance-preserving*" si les agents se rencontrent pendant une sous-trajectoire ou reviennent à une distance initiale pareille entre eux entre chaque sous-trajectoire. Les stratégies bornées spatialement intersectent les stratégies "*distance-preserving*" mais ce n'est pas une inclusion.

Un problème fondamental est donc que dépendant de comment nous définissons la structure de nos stratégies, nous ne pourrons jamais converger vers des stratégies hors de nos définitions<sup>1</sup>. Comme amélioration possible serait donc enlever encore plus de contraintes mais il y a quelques problèmes. Moins il y a de contraintes, plus c'est compliqué d'interpréter les stratégies (ou les résultats) avec le nombre de chemins possibles qui augmente exponentiellement (par exemple quand la borne augmente pour les stratégies bornées spatialement).

C'est un des problèmes que mon superviseur a rencontré quand il a travaillé sur des stratégies avec moins de contraintes que nos stratégies.

De plus, il faut utiliser une autre structure que juste l'arbre (nous ne voulons pas un arbre de profondeur infini aussi). Lorsque le nombre d'observations est trop grand, cela serait probablement plus judicieux d'utiliser une fonction d'approximation pour les politiques. Les fonctions d'approximations seraient utiles par exemple pour avoir peu de paramètres à stocker comparé à une "structure exacte".

Des soucis avec les réseaux de neurones est qu'il faut optimiser beaucoup d'hyperparamètres (pour juste un *fully connected feedforward*, il y a par exemple le learning rate, le nombre de couches, choisir les fonctions d'activations, le nombre de neurones par couches, la méthode d'optimisation, etc.) et cela peut prendre beaucoup de temps pour entraîner et peut aussi ne pas converger (par exemple si le learning rate est trop grand).

Malgré cela, en terme d'algorithmes ou méthodes, cela serait intéressant de recommencer avec REINFORCE avec un réseau de neurones à la place et sans la projection de gradient pour voir les performances. Pour aller encore plus loin, il y a bien sûr des méthodes d'apprentissage par renforcement dans le *deep RL* plus poussées que REINFORCE.

---

<sup>1</sup>voir la section sur les limites de nos stratégies pour les exemples

Du côté implémentation, le code pourrait être amélioré, par exemple le rendre plus compréhensible et peut être même utiliser OpenAI Gym.

# Chapitre 8

## Appendice

Dans cette appendice, nous mettons nos résultats sous forme de string avec les probabilités. Contrairement aux notations de l'article, nous écrivons aussi les stratégies commençant avec l'action  $-1$  si la probabilité de la prendre est non nulle. Dans nos notations, nous ignorons donc l'idée du *forward* qui change avec probabilité  $\frac{1}{2}$ .

Cela est possible que les probabilités ne somment pas à 1 car cela a été tronquée et nous ignorons les chemins qui ont de très faibles probabilités.

### 8.1 Stratégies avec deux phases

#### 8.1.1 Renforcer dernière sous-trajectoire

- $K = 1, M = 1, R_2^s \approx 7.00015 : g_1 = \{-1, 1\}, g_2 = \{1, -1\}, x_1 \approx 0.4978589, x_2 = 1 - x_1$
- $K = 2, M = 1, R_3^s \approx 10.0009 : g_1 = \{-1, 1, 0\}, g_2 = \{-1, 0, 1\}, g_3 = \{1, -1, 0\}, g_4 = \{1, 0, -1\}, x_1 \approx 0.193186915043, x_2 \approx 0.311140754957, x_3 \approx 0.191961164544, x_4 \approx 0.303711165456$
- $K = 3, M = 1, R_4^s \approx 6.00029 : g_1 = \{-1, 1, 1, -1\}, g_2 = \{1, -1, -1, 1\}, x_1 \approx 0.49700384, x_2 \approx 0.50299616$
- $K = 2, M = 2, R_4^s \approx 8.00019 : g_1 = \{-1, -1, 1, 1\}, g_2 = \{-1, 1, 0, 0\}, g_3 = \{1, 1, -1, -1\}, g_4 = \{1, -1, 0, 0\}, x_1 \approx 0.250185999911, x_2 \approx 0.250458160089, x_3 \approx 0.247048227256, x_4 \approx 0.252307612744$
- $K = 4, M = 1, R_5^s \approx 7.00151 : g_1 = \{-1, 1, 1, -1, 0\}, g_2 = \{-1, 1, 1, 0, -1\}, g_3 = \{1, -1, -1, 1, 0\}, g_4 = \{1, -1, -1, 0, 1\}, x_1 \approx 0.128986494581, x_2 \approx 0.377155175419, x_3 \approx 0.11841298466, x_4 \approx 0.37544534534$
- $K = 3, M = 2, R_5^s \approx 7.00271 : g_1 = \{-1, 1, 1, -1, 0\}, g_2 = \{1, -1, -1, 1, 0\}, x_1 \approx 0.508224932, x_2 \approx 0.491775068$
- $K = 5, M = 1, R_6^s \approx 8.50361 : g_1 = \{-1, 0, 1, 1, -1, 0\}, g_2 = \{-1, 0, 1, 1, 0, -1\}, g_3 = \{1, 0, -1, -1, 1, 0\}, g_4 = \{1, 0, -1, -1, 0, 1\}, x_1 \approx 0.132908577585, x_2 \approx 0.358414792415, x_3 \approx 0.162217541938, x_4 \approx 0.346459088062$

- $K = 4, M = 2, R_6^s \approx 13.33467 : g_1 = \{0, 0, -1, -1, 1, 1\}, g_2 = \{0, 0, -1, 1, 0, 0\}, g_3 = \{0, 0, 1, 1, -1, -1\}, g_4 = \{0, 0, 1, -1, 0, 0\}, x_1 \approx 0.248432580099, x_2 \approx 0.245230179901, x_3 \approx 0.250461037306, x_4 \approx 0.255876202694$
- $K = 3, M = 3, R_6^s \approx 8.00001 : g_1 = \{-1, 1, 1, -1, 0, 0\}, g_2 = \{1, -1, -1, 1, 0, 0\}, x_1 \approx 0.49942286, x_2 \approx 0.50057714$

## 8.2 Stratégies sans deux phases

### 8.2.1 Renforcer dernière sous-trajectoire

- $N = 2, R_2^s \approx 7.000576 : g_1 = \{-1, 1\}, g_2 = \{1, -1\}, x_1 \approx 0.4957577, x_2 = 1 - x_1$
- $N = 3, R_3^s \approx 10.000446 : g_1 = \{-1, 1, 0\}, g_2 = \{-1, 0, 1\}, g_3 = \{1, -1, 0\}, g_4 = \{1, 0, -1\}, x_1 \approx 0.164670745168, x_2 \approx 0.332280084832, x_3 \approx 0.163221350925, x_4 \approx 0.339827819075$
- $N = 4, R_4^s \approx 6.000414 : g_1 = \{-1, 1, 1, -1\}, g_2 = \{1, -1, -1, 1\}, x_1 \approx 0.49640187, x_2 \approx 0.50359813$
- $N = 5, R_5^s \approx 8.000466 : g_1 = \{0, -1, 1, 1, -1\}, g_2 = \{0, 1, -1, -1, 1\}, x_1 \approx 0.50341171, x_2 \approx 0.49658829$
- $N = 6, R_6^s \approx 5.763796 : g_1 = \{-1, 1, 1, -1, -1, 1\}, g_2 = \{-1, 1, 1, -1, 1, -1\}, g_3 = \{-1, 1, 1, 1, -1, -1\}, g_4 = \{1, -1, -1, 1, 1, -1\}, g_5 = \{1, -1, -1, 1, -1, 1\}, g_6 = \{1, -1, -1, -1, 1, 1\}, x_1 \approx 0.20503399947, x_2 \approx 0.218336713715, x_3 \approx 0.0639650700443, x_4 \approx 0.214068816129, x_5 \approx 0.202055506669, x_6 \approx 0.0851557372022$

### 8.2.2 REINFORCE-like

- $N = 2, R_2^s \approx 8.37848 : g_1 = \{-1, 1\}, g_2 = \{0, 0\}, g_3 = \{1, -1\}, x_1 \approx 0.46515888, x_2 \approx 0.07638655, x_3 \approx 0.45845457$
- $N = 3, R_3^s \approx 11.10966 : g_1 = \{-1, 1, 0\}, g_2 = \{-1, 0, 1\}, g_3 = \{0, -1, 1\}, g_4 = \{0, 0, 0\}, g_5 = \{0, 1, -1\}, g_6 = \{1, -1, 0\}, g_7 = \{1, 0, -1\}, x_1 \approx 0.171290321372, x_2 \approx 0.284019398628, x_3 \approx 0.0441991409849, x_4 \approx 0.00942353426851, x_5 \approx 0.0434522147466, x_6 \approx 0.157092384725, x_7 \approx 0.290523005275$
- $N = 4, R_4^s \approx 8.51376 : g_1 = \{-1, -1, 1, 1\}, g_2 = \{-1, 0, 0, 1\}, g_3 = \{-1, 0, 1, 0\}, g_4 = \{-1, 1, -1, 1\}, g_5 = \{-1, 1, 0, 0\}, g_6 = \{-1, 1, 1, -1\}, g_7 = \{0, -1, 0, 1\}, g_8 = \{0, -1, 1, 0\}, g_9 = \{0, 0, -1, 1\}, g_{10} = \{0, 0, 0, 0\}, g_{11} = \{0, 0, 1, -1\}, g_{12} = \{1, 1, -1, -1\}, g_{13} = \{1, 0, 0, -1\}, g_{14} = \{1, 0, -1, 0\}, g_{15} = \{1, -1, 1, -1\}, g_{16} = \{1, -1, 0, 0\}, g_{17} = \{1, -1, -1, 1\}, x_1 \approx 0.214196909671, x_2 \approx 0.0195390214803, x_3 \approx 0.0181910527028, x_4 \approx 0.0976015464765, x_5 \approx 0.0171921882281, x_6 \approx 0.0960934081779, x_7 = 0.0194866943951, x_8 \approx 0.0182433797881, x_9 \approx 0.00307587525175, x_{10} \approx 0.000541805209069, x_{11} \approx 0.00302834685249, x_{12} \approx 0.207628514398, x_{13} \approx 0.0188930143482, x_{14} \approx 0.0182540550371, x_{15} \approx 0.0960934081779, x_{16} \approx 0.0171921882281, x_{17} \approx 0.0976015464765$

# Références

- Han, Qiaoming et al. (June 2008). “Improved Bounds for the Symmetric Rendezvous Value on the Line”. In: *Operations Research* 56.3, pp. 772–782. DOI: [10.1287/opre.1070.0439](https://doi.org/10.1287/opre.1070.0439).
- Gosavi, Abhijit (2014). *Simulation-based optimization: parametric optimization techniques and reinforcement learning*. New York: Springer. ISBN: 978-1-4899-7490-7.
- Luenberger, David G. and Yinyu Ye (July 2015). *Linear and Nonlinear Programming*. en. 4th ed. 2016 edition. Springer. ISBN: 978-3-319-18841-6.
- Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: an introduction*. en. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-03924-6.