



Faculté
des Sciences

UMONS
Université de Mons

SERVICE D'INFORMATIQUE THÉORIQUE

Plus court chemin stochastique dans les Processus Décisionnels de Markov

Auteur :
Florent Delgrange

Directeurs :
Véronique Bruyère
Mickael Randour

Année académique 2016-2017

Remerciements

Je remercie toutes les personnes qui ont contribué à ce projet et qui m'ont soutenu durant sa réalisation, tout particulièrement Véronique Bruyère et Mickael Randour pour leurs précieux conseils, mais aussi Carolane Gerbehaye pour son appui.

Table des matières

1	Introduction aux probabilités	11
2	Chaînes de Markov	15
2.1	Définitions et propriétés	15
2.2	Chemins dans une chaîne de Markov	19
2.3	Problème d'accessibilité	21
2.3.1	Énoncé du problème	22
2.3.2	Résolution du problème	23
2.3.3	Généralisation matricielle	24
2.4	Chaînes de Markov pondérées	26
2.4.1	Problème de l'espérance du coût de l'accessibilité	28
2.4.2	Problème d'accessibilité limitée par un coût	31
3	Processus Décisionnels de Markov	35
3.1	Définitions et propriétés	35
3.2	Chemins et Stratégies de PDM	39
3.3	Problème d'accessibilité	46
3.3.1	Énoncé du problème	46
3.3.2	Résolution du problème	47
3.4	Le problème du plus court chemin stochastique	51
3.4.1	Minimiser l'espérance de la longueur des chemins	54
3.4.2	Forcer des chemins de taille faible sous une haute probabilité	58
4	Implémentation	61
4.1	Représentation des Processus Décisionnels de Markov	61
4.2	Solveurs	63
4.2.1	Problème d'accessibilité	64
4.2.2	Problème de l'espérance du plus court chemin stochastique	72
4.2.3	Problème des plus courts chemins stochastiques de taille limitée	74
4.3	Benchmarks	78
4.4	Utilisation du programme	81
4.4.1	Importer des Processus Décisionnels de Markov	81
4.4.2	Exporter des Processus Décisionnels de Markov	82
4.4.3	Générer des Processus Décisionnels de Markov	83
4.4.4	Solveurs	83

Définitions et théorèmes

Chapitre 1

1.1	Définition – σ -algèbre	11
1.2	Définition – Mesure de probabilité	11
1.3	Définition – Distribution de probabilité	12
1.4	Définition – Variable aléatoire	14
1.5	Définition – Espérance	14

Chapitre 2

2.1	Définition – Chaîne de Markov à temps discret	15
2.2	Définition – Graphe sous-jacent d'une chaîne de Markov	16
2.3	Définition – Matrice de transition	17
2.4	Définition – Chemin	19
2.5	Définition – Préfixe d'un chemin	19
2.6	Définition – Cylindre d'un chemin fini	19
2.7	Définition – Connexité à un sous ensemble de sommets dans un graphe	23
2.1	Théorème	24
2.8	Définition – Chaîne de Markov pondérée	26
2.9	Définition – Somme tronquée	27
2.10	Définition – Espérance du coût de l'accessibilité	28
2.2	Théorème	30
2.11	Définition – Accessibilité limitée par un coût	31

Chapitre 3

3.1	Définition – Processus décisionnel de Markov	35
3.2	Définition – Graphe sous-jacent d'un processus décisionnel de Markov	38
3.3	Définition – Chemins dans un PDM	39
3.4	Définition – Histoire	40
3.5	Définition – Stratégie	40
3.6	Définition – CM d'un PDM induite par stratégie	40
3.7	Définition – Stratégie à mémoire finie	43
3.8	Définition – Produit d'un PDM par une stratégie à mémoire finie	43
3.9	Définition – Stratégie sans mémoire	45
3.1	Théorème	47
3.2	Théorème	49

3.10 Définition – PDM pondéré	52
3.11 Définition – CMP induite par stratégie	52
3.12 Définition – Somme Tronquée d'un PDMP	52
3.13 Définition – L'espérance du plus court chemin stochastique	54
3.3 Théorème	54
3.14 Définition – Les plus courts chemins stochastiques de taille limitée	58
3.4 Théorème	59

Introduction

La résolution du problème du plus court chemin dans un graphe possède une quantité indénombrable d'applications, comme par exemple l'optimisation des réseaux de télécommunications ou la détermination du chemin le plus court entre deux villes via un réseau routier. Mais qu'en est-il lorsque l'environnement du problème que l'on étudie est stochastique et sujet à des phénomènes aléatoires divers ? En pratique, très peu de situations se déroulent dans un environnement parfait et cela est rare de pouvoir affirmer l'exactitude du contexte du problème que l'on traite. Les *processus décisionnels de Markov* permettent de modéliser ces situations évoluant dans un milieu stochastique et dans lesquelles des prises de décisions sont requises.

Les processus décisionnels de Markov sont des automates probabilistes dans lesquels une prise de décision est requise à chaque étape. Chaque décision a un coût et, lorsque le système est en un état, la prise de décision entraîne un comportement probabiliste du système, qui évolue ensuite dans l'état suivant. On va donc définir des *stratégies* qui détermineront, à chaque étape, la décision à prendre afin que le système évolue dans l'état suivant. Par le fait que l'environnement du système est probabiliste, la stratégie ne peut assurer au système d'évoluer vers un ou plusieurs états avec un chemin de coût fixe et ne peut donc pas résoudre le problème du plus court chemin à proprement parler. On parle donc de *plus court chemin stochastique*. Ce problème peut être abordé par différentes approches. Dans ce document, on va définir des stratégies qui vont maximiser l'*espérance de la longueur du plus court chemin* et la *probabilité d'accessibilité avec des chemins de taille inférieure à un seuil*. De cette façon, lorsqu'on souhaitera voyager entre deux villes via un réseau routier, de par le caractère stochastique du voyage, il sera plus intéressant d'étudier ces problèmes de plus court chemin stochastique dans le processus décisionnel de Markov qui lui est associé.

Le but final de ce travail est d'implémenter les processus décisionnels de Markov, le problème d'accessibilité et les deux problèmes de plus court chemin stochastique qui leur sont associés. Pour ce faire, on va tout d'abord définir ce qu'est une probabilité, comment la mesurer et dans quelles conditions. Ensuite, nous allons étudier des systèmes qui modélisent des phénomènes aléatoires (i.e., les *chaînes de Markov*) qui seront indispensables pour mesurer la probabilité des *chemins* d'un processus décisionnel de Markov. En effet, on va voir que le non-déterminisme lié à la prise de décision ne permet pas de mesurer la probabilité des chemins des processus décisionnels de Markov sans les notions de stratégies et de chaîne de Markov. C'est ce qui va nous amener à

traiter les problèmes *d'accessibilité, d'espérance du coût de l'accessibilité* ainsi que celui de *l'accessibilité limitée par un coût* dans une chaîne de Markov. Finalement, cela nous permettra d'étudier les processus décisionnels de Markov et les stratégies qui vont résoudre les problèmes *d'accessibilité, d'espérance du plus court chemin stochastique* ainsi que celui des *plus courts chemins stochastiques de taille limitée*.

Chapitre 1

Introduction aux probabilités

Les processus décisionnels de Markov sont des systèmes qui modélisent des situations aléatoires enrichis par des probabilités. Avant de définir formellement de tels modèles, il est utile d'introduire brièvement quelques notions de probabilités qui seront indispensables à la compréhension de la suite du document.

Définition 1.1 (σ -algèbre). Une σ -algèbre est une paire (Ω, σ) où Ω est un ensemble non-vide et $\sigma \subseteq \mathcal{P}(\Omega)$ qui respecte les 3 conditions suivantes :

1. $\emptyset \in \sigma$
2. Si $E \in \sigma$, alors $\overline{E} = \Omega \setminus E$ et $\overline{\overline{E}} \in \sigma$
3. Si $E_1, E_2, \dots \in \sigma$, alors $\bigcup_{n \geq 1} E_n \in \sigma$

Les éléments de Ω sont appelés *résultats* et les éléments de σ sont appelés *événements*.

Remarque 1.1. Ces conditions sur la σ -algèbre mènent au fait que

- $\Omega \in \sigma$. En effet, on a que Ω est non-vide par définition et $\emptyset \in \sigma$. Donc, $\overline{\Omega} = \Omega \setminus \emptyset = \Omega \in \sigma$.
- σ est fermé par des intersections dénombrables, i.e., $\forall E \in \sigma$ et $\forall n$ tel que $E_n \subseteq E$,

$$\bigcap_{n \geq 0} E_n = \overline{\bigcup_{n \geq 0} E_n}$$

En effet, $\bigcup_{n \geq 0} \overline{E_n}$ avec $\overline{E_n} = \Omega \setminus E_n \in \sigma$ est l'*union des ensembles formés de tous les éléments de Ω non-contenus dans chaque ensemble E_n* .

$\overline{\bigcup_{n \geq 0} E_n} = \Omega \setminus (\bigcup_{n \geq 0} \overline{E_n}) \in \sigma$ est donc l'*ensemble des éléments contenus dans tous les ensembles E_n* , i.e., $\bigcap_{n \geq 0} E_n$.

Définition 1.2 (Mesure de probabilité). Soit (Ω, σ) , une σ -algèbre. Une mesure de probabilité sur (Ω, σ) est une fonction $\mathbb{P} : \sigma \rightarrow [0, 1]$ telle que

- $\mathbb{P}(\Omega) = 1$
- Si $(E_n)_{n \geq 1}$ est une suite d'événements disjoints $E_n \in \sigma$, alors :

$$\mathbb{P}\left(\bigcup_{n \geq 1} E_n\right) = \sum_{n \geq 1} \mathbb{P}(E_n)$$

On dit alors que $(\Omega, \sigma, \mathbb{P})$ est un *espace probabiliste*. On appelle $\mathbb{P}(E)$ la mesure de probabilité de l'évènement E ou encore plus simplement la probabilité de E .

Définition 1.3 (Distribution de probabilité). Soit (Ω, σ) , une σ -algèbre. On suppose que Ω est un ensemble dénombrable. Alors, il existe une fonction $\mu : \Omega \rightarrow [0, 1]$, une mesure de probabilité telle que

$$\sum_{\omega \in \Omega} \mu(\omega) = 1$$

μ est appelée *distribution de probabilité sur Ω* . Pour avoir une mesure de probabilité sur la σ -algèbre dont $\sigma = \mathcal{P}(\Omega)$, il suffit d'avoir μ sur Ω et d'étendre μ à \mathbb{P} de la façon suivante :

$$\forall E \in \sigma, \mathbb{P}_\mu(E) = \sum_{\omega \in E} \mu(\omega)$$

On dit que \mathbb{P}_μ est la mesure de probabilité induite par la distribution de probabilité μ . On dénote par $\mathcal{D}(\Omega)$ l'ensemble des distributions de probabilité sur Ω .

Propriétés

Nous allons maintenant introduire quelques propriétés que nous allons utiliser au fil du document. Soit $(\Omega, \sigma, \mathbb{P})$, un espace probabiliste.

Propriété 1.1. $\forall E \in \sigma, \mathbb{P}(E) = 1 - \mathbb{P}(\bar{E})$. En particulier, $\mathbb{P}(\emptyset) = 0$ (car $\mathbb{P}(\Omega) = 1$).

Propriété 1.2 (Les mesures de probabilité sont monotones).

$$\forall E, E' \in \sigma \text{ tel que } E \subseteq E', \mathbb{P}(E') = \mathbb{P}(E) + \mathbb{P}(E' \setminus E) \geq \mathbb{P}(E)$$

De ce fait, soit $(E_n)_{n \geq 1}$, une suite d'évènements (pas forcément disjoints).

$$\bigcup_{n \geq 1} E_n = \bigcup_{n \geq 1} E'_n \text{ où } E_1 = E'_1 \text{ et } E'_n = E_n \setminus (E_1 \cup E_2 \cup \dots \cup E_{n-1}) \quad \forall n \geq 2$$

Par définition de $(E'_n)_{n \geq 1}$, on a toujours que $E'_n \cap E'_m = \emptyset$ quand $n \neq m$ (tous les éléments de la suite sont des ensembles disjoints), et donc

$$\mathbb{P}\left(\bigcup_{n \geq 1} E_n\right) = \mathbb{P}\left(\bigcup_{n \geq 1} E'_n\right) = \sum_{n \geq 1} \mathbb{P}(E'_n) \quad (\text{par la définition 1.2})$$

Supposons à présent que $E_1 \subseteq E_2 \subseteq E_3 \subseteq \dots$, alors, par le fait que $\forall E, E' \in \sigma$ tels que

$E \subseteq E'$, $\mathbb{P}(E') = \mathbb{P}(E) + \mathbb{P}(E' \setminus E)$, on a :

$$\begin{aligned}\mathbb{P}(\bigcup_{n \geq 1} E_n) &= \mathbb{P}(E_1) + \sum_{n=2}^{\infty} (\mathbb{P}(E_n) - \mathbb{P}(E_{n-1})) \\ &= \mathbb{P}(E_1) + \sum_{n=2}^{\infty} \mathbb{P}(E_n \setminus E_{n-1}) \\ &= \mathbb{P}(E'_1) + \sum_{n=2}^{\infty} \mathbb{P}(E'_n) \\ &= \lim_{n \rightarrow \infty} \mathbb{P}(E_n)\end{aligned}$$

et $\mathbb{P}(E_1) \leq \mathbb{P}(E_2) \leq \dots \leq \mathbb{P}(E_n) \leq 1$.

Propriété 1.3. Soit $(E_n)_{n \geq 1}$, une suite d'évènements. Supposons que $E_1 \supseteq E_2 \supseteq E_3 \supseteq \dots$, alors :

$$\mathbb{P}(\bigcap_{n \geq 1} E_n) = \lim_{n \rightarrow \infty} \mathbb{P}(E_n)$$

En effet, par le fait que $\forall n$, $E_n \supseteq E_{n+1} \iff \overline{E_n} \subseteq \overline{E_{n+1}}$ (car $\overline{E_n} = \Omega \setminus E_n$),

$$\begin{aligned}\mathbb{P}\left(\bigcap_{n \geq 1} E_n\right) &= \mathbb{P}\left(\overline{\bigcup_{n \geq 1} \overline{E_n}}\right) && \text{(par la remarque 1.1)} \\ &= 1 - \mathbb{P}\left(\bigcup_{n \geq 1} \overline{E_n}\right) && \text{(par la propriété 1.1)} \\ &= 1 - \lim_{n \rightarrow \infty} \mathbb{P}(\overline{E_n}) && \text{(par la monotonie)} \\ &= 1 - \lim_{n \rightarrow \infty} (1 - \mathbb{P}(E_n)) \\ &= \lim_{n \rightarrow \infty} \mathbb{P}(E_n)\end{aligned}$$

Exemple 1.1 (Lancer d'un dé). On lance un dé. Chaque face a exactement une chance sur six d'apparaître suite à ce lancer de dé. On définit formellement le σ -algèbre correspondant à ce lancer de dé : Les résultats sont $\Omega = \{1, 2, 3, 4, 5, 6\}$ et les évènements sont $\sigma = \mathcal{P}(\Omega)$. On sait qu'il existe une fonction $\mu : \Omega \rightarrow [0, 1]$ telle que μ est une distribution de probabilité. Prenons $\mu(\omega) = \frac{1}{6}$ pour tout $\omega \in \Omega$.

On est à présent intéressé par la probabilité des évènements suivants à l'aide de la mesure de probabilité \mathbb{P}_μ induite par μ :

- "Le résultat du lancer de dé est 1 ou 6" = $\{1, 6\} \in \sigma$

$$\mathbb{P}_\mu(\{1, 6\}) = \mu(1) + \mu(6) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$

- "Le résultat du lancer de dé n'est pas 1 ou 6" = $\overline{\{1, 6\}} = \{2, 3, 4, 5\}$

$$\mathbb{P}_\mu(\{2, 3, 4, 5\}) = 1 - \mathbb{P}_\mu(\{1, 6\}) = 1 - \frac{1}{3} = \frac{2}{3}$$

Espérance mathématique

Nous serons par la suite, à de nombreuses reprises, confrontés à la notion d'*espérance*. Il est donc intéressant d'introduire les concepts liés à cette notion.

Définition 1.4 (Variable aléatoire). [3]

Soient $I \subseteq \mathbb{N}$, $\mathcal{X} = (x_i)_{i \in I}$, un ensemble dénombrable et $(\Omega, \sigma, \mathbb{P})$, un espace probabiliste. Soit X , une application $X : (\Omega, \sigma, \mathbb{P}) \rightarrow \mathcal{X} = (x_i)_{i \in I}$, $\omega \mapsto X(\omega)$. X est une *variable aléatoire* si

$$\forall i \in I, \quad \{X = x_i\} = \{\omega \in \Omega \mid X(\omega) = x_i\} \in \sigma$$

i.e., si on peut faire correspondre chaque élément de \mathcal{X} à un évènement de l'espace probabiliste. Cette condition assure que tout ensemble $\{X = x_i\}$ possède une probabilité mesurable par \mathbb{P} .

Exemple 1.2 (Parité lors d'un lancer de dé). On lance un dé (cf. exemple 1.1) et on est intéressé de savoir si la face du dé qui apparaît est paire ou impaire. On définit la variable aléatoire discrète $X : (\Omega, \sigma, \mathbb{P}_\mu) \rightarrow \mathcal{X} = \{\text{pair}, \text{impair}\}$. On a $\{X = \text{pair}\} = \{2, 4, 6\}$ et $\{X = \text{impair}\} = \{1, 3, 5\}$. Dès lors $\mathbb{P}_\mu(X = \text{pair}) = \mathbb{P}_\mu(\{2, 4, 6\}) = \frac{1}{2} = \mathbb{P}_\mu(X = \text{impair}) = \mathbb{P}_\mu(\{1, 3, 5\})$.

Définition 1.5 (Espérance). [3]

Soit X , une variable aléatoire réelle. On définit $\mathbb{E}[X]$ comme étant *l'espérance* de X où

$$\mathbb{E}[X] = \sum_{i \in I} x_i \cdot \mathbb{P}(X = x_i)$$

i.e., l'espérance de X est la **moyenne pondérée** des valeurs x_i par la probabilité que $X = x_i$ ou encore la moyenne de X .

Exemple 1.3 (Espérance d'un lancer de dé). On suppose qu'on lance un dé (cf. exemple 1.1). Soit X , une variable aléatoire représentant les faces du dé. On a

$$\mathbb{E}[X] = \sum_{x=1}^6 x \cdot \mathbb{P}(X = x) = \frac{1}{6} \sum_{x=1}^6 x = \frac{1}{6} \cdot \frac{6 \cdot 7}{2} = \frac{7}{2}$$

i.e., l'espérance d'un lancer de dé est 3.5.

Chapitre 2

Chaînes de Markov

On va à présent étudier des systèmes sujets à des phénomènes aléatoires, i.e., dont le comportement est incertain et imprévisible (e.g., un protocole de communication dont les messages peuvent être perdus, une installation de panneaux solaires dont la production dépend des conditions météorologiques, un algorithme dont la sortie dépend de valeurs générées pseudo-aléatoirement, etc.). Afin d'étudier de tels phénomènes, on modélise ces derniers en définissant un système de transitions où chaque transition d'un état vers ses successeurs suit une distribution de probabilité dépendante de cet état. De ce fait, la probabilité que le système évolue d'un état courant à un autre dépend entièrement de cet état courant. Chaque état du système est donc le résultat d'un phénomène aléatoire. On appelle ce type de modèle une *chaîne de Markov*.

Ce chapitre est essentiellement inspiré du chapitre *Probabilistic Systems* du livre *Principles of model checking* [1] ainsi que du chapitre intitulé *Model Checking Probabilistic Systems* du cours de Mickael Randour (*Formal verification of computer systems*) [5].

2.1 Définitions et propriétés

Définition 2.1 (Chaîne de Markov à temps discret). Une *chaîne de Markov à temps discret*, notée **CM**, est un automate probabiliste défini par un tuple $\mathcal{M} = (S, \Delta)$ où :

- S est un ensemble dénombrable d'états.
- $\Delta : S \times S \rightarrow [0, 1] \cap \mathbb{Q}$ est la *fonction de transition* telle que

$$\forall s \in S, \sum_{s' \in S} \Delta(s, s') = 1$$

Δ spécifie, pour tout état $s \in S$, la probabilité de passer de l'état s à l'état s' . On dit que \mathcal{M} est *finie* si S est un ensemble fini.

Propriété 2.1. Soient $\mathcal{M} = (S, \Delta)$ et $s \in S$. Les contraintes imposées sur Δ assurent

que Δ_s est une distribution de probabilité sur S (par la définition 1.3) avec

$$\Delta_s : S \rightarrow [0, 1] \cap \mathbb{Q}, s' \mapsto \Delta(s, s')$$

On dénote par \mathbb{P}_s la mesure de probabilité induite par la distribution de probabilité Δ_s .

Définition 2.2 (Graphe sous-jacent d'une chaîne de Markov). Une CM $\mathcal{M} = (S, \Delta)$ induit un *graphe sous-jacent* (orienté) $G^{\mathcal{M}} = (V, E)$ où :

- V est l'ensemble de sommets du graphe tel que $|V| = |S|$, i.e., il existe une bijection de S vers V . Chaque sommet $s' \in V$ est donc associé à un unique état $s \in S$. Par abus de langage, on dit que $V = S$.
- E est l'ensemble des arcs du graphe tel que

$$\forall s, s' \in S, (s, s') \in E \text{ssi } \Delta(s, s') > 0$$

On dit que s' est un successeur de s dans $G^{\mathcal{M}}$ ssi $(s, s') \in E$.

Afin d'illustrer une chaîne de Markov, on utilise la représentation de son graphe sous-jacent où chaque arc $(s, s') \in E$ est étiqueté par la probabilité de passer de l'état s à l'état s' en une étape : $\Delta(s, s')$.

Exemple 2.1 (Simuler un lancer de dé avec une pièce de monnaie). On génère le comportement d'un dé via une pièce de monnaie selon l'algorithme probabiliste de Knuth et Yao [4]. Cet algorithme est simulé à l'aide de la chaîne de Markov \mathcal{M}_{Kd} illustrée à la figure 2.1.

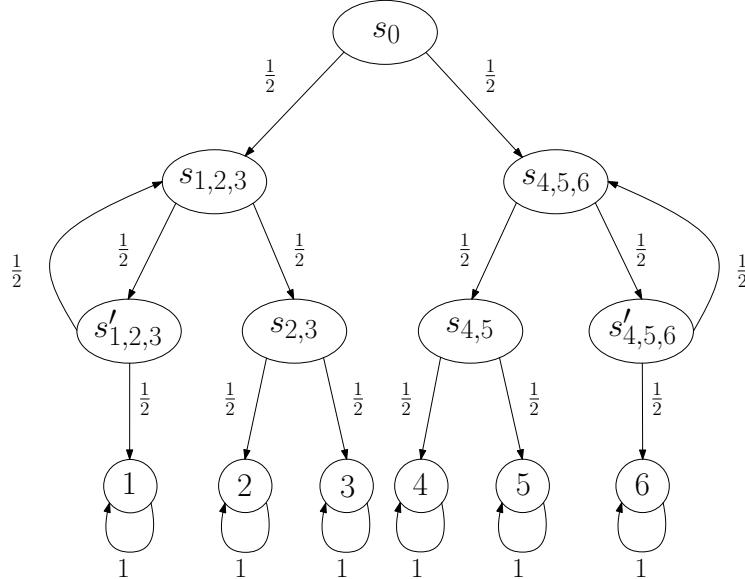


FIGURE 2.1 – Simulation d'un lancer de dé avec une pièce par une chaîne de Markov

On commence en l'état s_0 , qu'on appellera ici l'état initial. Les états $1, 2, 3, 4, 5, 6$ sont appelés états finaux et représentent les différentes faces du dé (i.e., les résultats possibles du lancer de dé), tandis que les états internes ($\notin \{s_0, 1, 2, 3, 4, 5, 6\}$) représentent les états du système après un lancer de pièce. Pour tout état interne, un lancer

de pièce, dont le résultat est face, emprunte l'arc de gauche pour déterminer son état suivant. Un lancer de pièce, dont le résultat est pile, emprunte l'arc de droite pour déterminer son état suivant. Lorsqu'on lance un dé, la probabilité de tomber sur n'importe quelle face du dé est exactement de $\frac{1}{6}$. Le comportement du modèle doit donc simuler ce phénomène lorsqu'un état final est atteint.

Simulation : On suppose que le système démarre en l'état s_0 . On lance une pièce. Si le résultat est face, le système évolue en l'état $s_{1,2,3}$. La probabilité que le résultat du lancer de pièce soit pile est égale à la probabilité que le résultat du lancer de pièce soit face. Par conséquent, en relançant la pièce, la probabilité que le système évolue en $s_{2,3}$ est égale à la probabilité que le système évolue en $s'_{1,2,3}$. Si le système évolue en $s'_{1,2,3}$, un lancer de pièce mène à la face 1 du dé avec une probabilité $\frac{1}{2}$, égale à la probabilité de retourner en l'état $s_{1,2,3}$. Sinon, le système évolue en $s_{2,3}$ et un lancer de pièce mènera obligatoirement au résultat d'un lancer de dé (avec une probabilité $\frac{1}{2}$ ou avec une probabilité $\frac{1}{2}$, et donc avec une probabilité 1), à savoir la face 2 ou 3. Le comportement du système se trouvant dans l'état s_0 , dans le cas où le résultat du lancer de pièce est pile, est symétrique.

On verra plus tard dans ce document qu'en simulant un lancer de dé par une pièce, en suivant le système décrit par la CM \mathcal{M}_{Kd} et en démarrant en l'état s_0 , chaque face du dé est atteinte avec une probabilité $\frac{1}{6}$.

Définition 2.3 (Matrice de transition). Soient $\mathcal{M} = (S, \Delta)$, une CM et $n = |S|$. On suppose que S est dénombrable. On peut dès lors énumérer les états de S . Soient $i, j \in \{1, \dots, n\}$ ($s_i \in S$, est le $i^{\text{ème}}$ sommet de S et $s_j \in S$ est le $j^{\text{ème}}$ sommet de S). Soit $\mathbf{P} \in \mathbb{Q}^{n \times n}$. On dit que \mathbf{P} est la matrice de transition de \mathcal{M} ssi $\mathbf{P}_{i,j} = \Delta(s_i, s_j)$. La ligne $\mathbf{P}_{i,\cdot}$ contient la probabilité des transitions de l'état s_i vers ses successeurs, tandis que la colonne $\mathbf{P}_{\cdot,j}$ spécifie la probabilité, pour tout état s , d'atteindre l'état s_j en une étape.

Exemple 2.2 (Modèle d'Ehrenfest pour la diffusion des gaz [2]). Le modèle est proposé par Ehrenfest pour décrire les échanges de chaleur entre deux systèmes portés initialement à une température différente. On modélise la répartition de N molécules de gaz à l'intérieur d'un récipient divisé en deux compartiments (urnes) séparés par une membrane poreuse.

Pour cet exemple simplifié, on prend $N = 4$. Les 2 urnes contiennent 4 molécules à tout moment et, à chaque étape, une des 4 molécules est choisie au hasard et change d'urne (cf. figure 2.2).

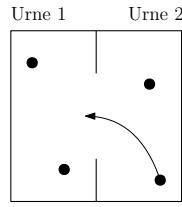


FIGURE 2.2 – Schéma simplifié du principe d’Ehrenfest pour $N = 4$ molécules.

On modélise ce phénomène par une chaîne de Markov (cf. figure 2.3). Ici, $S = \{(2|2), (1|3), (0|4), (3|1), (4|0)\}$. Chaque état représente le nombre de molécules présentes dans chacune des urnes. À chaque étape, chacune des molécules a exactement 1 chance sur 4 de changer d’urne.

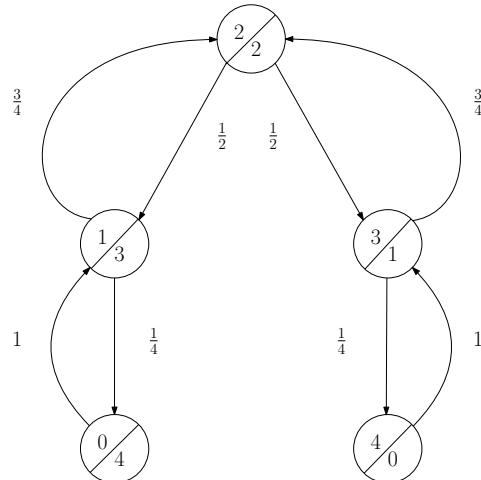


FIGURE 2.3 – Chaîne de Markov associée pour $N = 4$ molécules

Chaque état de S correspond à la répartition des molécules dans les 2 urnes. Supposons que le système se situe en l’état $(2|2)$. On a donc 2 molécules dans la première ainsi que dans la seconde urne. Alors, par le fait que chaque molécule change d’urne avec une probabilité $\frac{1}{4}$, le système a une probabilité $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$ d’évoluer en l’état $(1|3)$ ou en l’état $(3|1)$. On peut appliquer ce principe pour chaque état. En effet, supposons que le système évolue en $(1|3)$. Cela signifie qu’il n’y a qu’une seule molécule dans la première urne, tandis qu’il y a 3 molécules dans la seconde. Par le même principe qu’à l’étape précédente, le système a donc une probabilité $\frac{1}{4}$ d’évoluer en l’état $(0|4)$ et une probabilité $\frac{3}{4}$ d’évoluer en l’état $(2|2)$.

En énumérant les états de S comme suit : $(0|4), (1|3), (2|2), (3|1), (4|0)$, on a la matrice 5×5 de transition \mathbf{P} :

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{3}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

2.2 Chemins dans une chaîne de Markov

On va maintenant s'intéresser aux chemins dans une chaîne de Markov ainsi qu'aux propriétés qui s'y rapportent afin de pouvoir par la suite définir formellement des événements et calculer leur probabilité. Le but est donc d'être capable de résoudre différents problèmes comme par exemple celui de *l'accessibilité*.

Définition 2.4 (Chemin). Soit $\mathcal{M} = (S, \Delta)$, une CM. On définit $Paths(\mathcal{M})$ comme étant l'ensemble des *chemins infinis* de \mathcal{M} , i.e., des séquences $\pi = s_0s_1s_2\cdots \in S^\omega$ tel que $\Delta(s_i, s_{i+1}) > 0$ pour tout $i \geq 0$ (en d'autres termes, tel que l'arc $(s_i, s_{i+1}) \in E$ dans le graphe sous-jacent $G^{\mathcal{M}} = (S, E)$).

De la même façon, on définit $Paths_{fin}(\mathcal{M})$, comme étant l'ensemble des chemins finis de \mathcal{M} , i.e., des séquences $\hat{\pi} = s_0s_1s_2\cdots s_n$ tel que $\forall i \in \{0, \dots, n-1\}, s_i, s_{i+1} \in S$ et $\Delta(s_i, s_{i+1}) > 0$.

On dénote par $Paths(s)$ l'ensemble des chemins infinis qui commencent en l'état $s \in S$ et $Paths_{fin}(s)$, l'ensemble des chemins finis qui commencent en l'état $s \in S$.

Afin d'analyser le comportement d'une CM, il faut à présent définir formellement un espace probabiliste sur de tels chemins. Soit $\mathcal{M} = (S, \Delta)$, une CM et $s \in S$, un état de \mathcal{M} . On suppose que le système est en s . Les résultats possibles du système sont tous les chemins infinis de la CM commençant en l'état s et les événements sont tous les sous-ensembles formés par les chemins infinis de \mathcal{M} commençant en s . Plus formellement, on définit l'espace probabiliste $(\Omega, \sigma, \mathbb{P}_s)$ tel que $\Omega = Paths(s)$ et $\sigma = \mathcal{P}(Paths(s))$. On introduit la notion de *cylindre* afin de définir une mesure de probabilité \mathbb{P}_s pour cet espace probabiliste.

Définition 2.5 (Préfixe d'un chemin). Soient $\mathcal{M} = (S, \Delta)$, une CM et $\pi = s_0\cdots \in Paths(\mathcal{M})$, un chemin de \mathcal{M} . On définit $pref(\pi)$ comme étant *l'ensemble des préfixes de π* , i.e.,

$$pref(\pi) = \{\hat{\pi} = s'_0\cdots s'_n \in Paths_{fin}(\mathcal{M}) \mid \forall i \in \{0, \dots, n\}, s_i = s'_i\}$$

Définition 2.6 (Cylindre d'un chemin fini). Soient $\mathcal{M} = (S, \Delta)$, une CM et $\hat{\pi} \in Paths_{fin}(\mathcal{M})$, un chemin fini de \mathcal{M} . *L'ensemble cylindre* de $\hat{\pi}$ est défini comme suit :

$$Cyl(\hat{\pi}) = \{\pi \in Paths(\mathcal{M}) \mid \hat{\pi} \in pref(\pi)\}$$

Exemple 2.3 (Cylindre dans le système de dés de Knuth). Reprenons la CM \mathcal{M}_{Kd} définie dans l'exemple 2.1. Soit $\hat{\pi} = s_0s_{1,2,3}s_{2,3} \in Paths_{fin}(\mathcal{M}_{Kd})$. Alors,

$$Cyl(\hat{\pi}) = \{\pi \in Paths(\mathcal{M}) \mid \pi = s_0s_{1,2,3}s_{2,3}2^\omega \text{ ou } \pi = s_0s_{1,2,3}s_{2,3}3^\omega\}$$

L'ensemble des événements σ de l'espace probabiliste contient tous les cylindres des chemins finis $\hat{\pi}$ commençant en l'état s , i.e., $\{Cyl(\hat{\pi}) \mid \hat{\pi} \in Paths_{fin}(s)\} \subseteq \sigma$. On va voir que les cylindres engendrent les événements de l'ensemble σ et que, de ce fait, la mesure de probabilité de l'espace probabiliste sur les chemins de \mathcal{M} commençant en l'état s peut être définie à l'aide des cylindres.

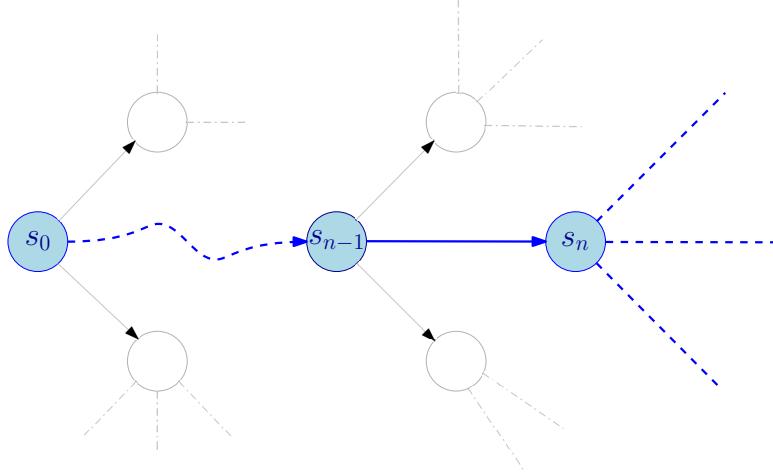


FIGURE 2.4 – L’ensemble cylindre du chemin fini $\hat{\pi} = s_0 \dots s_n$ est l’ensemble des chemins dont $\hat{\pi}$ est préfixe.

Propriété 2.2 (*Mesure de probabilité du cylindre d’un chemin fini*). Soient $\mathcal{M} = (S, \Delta)$, une CM, $s \in S$ et $\hat{\pi} = s_0 \dots s_n \in Paths_{fin}(s)$, un chemin fini de \mathcal{M} . Supposons que le système est actuellement en l’état s . Il existe une unique mesure de probabilité \mathbb{P}_s du cylindre de $\hat{\pi}$ sur $Paths(s)$ et celle-ci est définie par :

$$\mathbb{P}_s(Cyl(\hat{\pi})) = \Delta(\hat{\pi}) = \Delta(s_0 \dots s_n) = \prod_{i=0}^{n-1} \Delta(s_i, s_{i+1})$$

On peut dès lors mesurer la probabilité d’un chemin fini $\hat{\pi} \in Paths_{fin}(s)$ avec

$$\mathbb{P}_s(\{\hat{\pi}\}) = \mathbb{P}_s(Cyl(\hat{\pi}))$$

Remarque 2.1. Soit $\mathcal{M} = (S, \Delta)$, une CM et $s \in S$, un état de \mathcal{M} . Supposons que le système est en s . La probabilité du chemin fini $\hat{\pi} = s \in Paths_{fin}(s)$ est égale à la probabilité de $Cyl(\hat{\pi})$, i.e., $\mathbb{P}_s(Cyl(\hat{\pi}))$.

$$\mathbb{P}_s(\{s\}) = 1 = \mathbb{P}_s(Cyl(s)) = \mathbb{P}_s(Paths(s)) = \mathbb{P}_s(\Omega)$$

Corollaire 2.1 (*Probabilité d’un chemin*). Soient $\mathcal{M} = (S, \Delta)$, une CM, $s \in S$, un état et $\pi = s_0 s_1 \dots \in Paths(s)$, un chemin de \mathcal{M} . On suppose que l’état actuel du système est en s .

$$\mathbb{P}_s(\{\pi\}) = \Delta(\pi) = \Delta(s_0 s_1 \dots) = \prod_{i \in \mathbb{N}} \Delta(s_i, s_{i+1})$$

Démonstration. En effet, la probabilité d’un chemin infini $\pi \in Paths(s)$ correspond à une intersection infinie de cylindres :

$$\{\pi\} = \bigcap_{\hat{\pi} \in pref(\pi)} Cyl(\hat{\pi})$$

En définissant $\{\pi\}$ de cette façon, on peut mesurer sa probabilité avec la propriété

1.3 :

$$\begin{aligned}
\mathbb{P}_s(\{\pi\}) &= \mathbb{P}_s\left(\bigcap_{\hat{\pi} \in pref(\pi)} Cyl(\hat{\pi})\right) \\
&= \mathbb{P}_s\left(\bigcap_{i \in \mathbb{N}} C_i\right) && (\text{avec } C_i = Cyl(s_0 \dots s_i)) \\
&= \lim_{n \rightarrow \infty} \mathbb{P}_s(C_i) && (\text{car } \forall i, C_i \supseteq C_{i+1}) \\
&= \Delta(s_0 s_1 \dots) \\
&= \prod_{i \in \mathbb{N}} \Delta(s_i, s_{i+1})
\end{aligned}$$

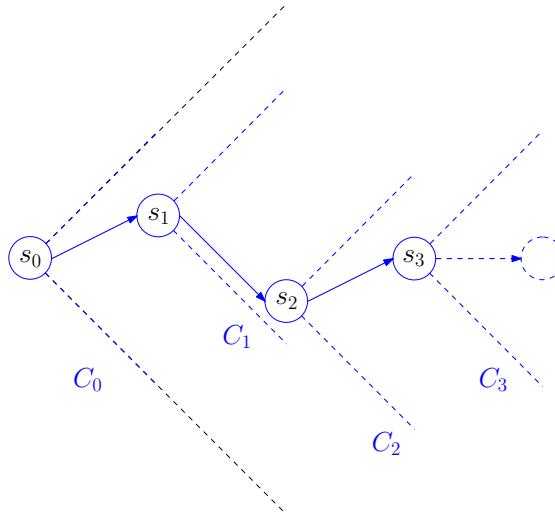


FIGURE 2.5 – Cylindres des préfixes de $\pi = s_0 s_1 \dots \in Paths(s)$ tels que
 $C_i = Cyl(s_0 \dots s_i)$

□

Exemple 2.4 (Chemins dans le système du dé de Knuth). Pour cet exemple, on reprend la CM de l'exemple 2.1. Soit le chemin $\pi = s_0 s_{1,2,3} s'_{1,2,3} s_{1,2,3} s_{2,3} 2^\omega \in Paths(\mathcal{M}_{Kd})$. On suppose que l'état actuel du système est s_0 . Alors, la probabilité que le système emprunte le chemin π est $\Delta(\pi) = \Delta(s_0 s_{1,2,3} s'_{1,2,3} s_{1,2,3} s_{2,3} 2^\omega) = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot 1 = \frac{1}{2^5} = \frac{1}{32}$

2.3 Problème d'accessibilité

L'un des problèmes les plus élémentaires de l'étude des chaînes de Markov est de déterminer la probabilité d'atteindre un sous-ensemble T d'états cibles du système. La résolution de ce problème est fortement liée à l'étude des problèmes que nous allons rencontrer par la suite dans ce document.

2.3.1 Énoncé du problème

Soient $\mathcal{M} = (S, \Delta)$ une CM et $T \subseteq S$, un ensemble d'états cibles. On dénote par $\Diamond T$ l'évènement "atteindre au moins un état de T via un chemin dans \mathcal{M} ".

Tout d'abord, il faut s'assurer que la probabilité de $\Diamond T$ soit mesurable.

Notation. $Paths_{fin}^T(\mathcal{M})$ désigne l'ensemble des chemins finis dans \mathcal{M} de la forme $\hat{\pi} = s_0 \dots s_n$ où $s_i \notin T$ pour tout $i \in \{0, \dots, n-1\}$ et $s_n \in T$ (et $Paths_{fin}^T(s)$ ceux qui commencent en l'état $s \in S$).

On peut exprimer $\Diamond T$ comme étant l'union dénombrable de tous les cylindres de $\hat{\pi} \in Paths_{fin}^T(\mathcal{M})$. Formellement, $\Diamond T$ est défini de la façon suivante :

$$\Diamond T = \bigcup_{s_0 \dots s_n \in Paths_{fin}^T(\mathcal{M})} Cyl(s_0 \dots s_n)$$

Propriété 2.3. Soit $s \in S$,

$$\forall \hat{\pi}_1, \hat{\pi}_2 \in Paths_{fin}^T(s) \text{ tels que } \hat{\pi}_1 \neq \hat{\pi}_2, \quad Cyl(\hat{\pi}_1) \cap Cyl(\hat{\pi}_2) = \emptyset$$

i.e., les cylindres des chemins finis de l'ensemble $Paths_{fin}^T(s)$ sont disjoints. En effet, supposons au contraire que $Cyl(\hat{\pi}_1) \cap Cyl(\hat{\pi}_2) \neq \emptyset$. Alors, cela signifie que $\exists \pi$ tel que $\pi \in Cyl(\hat{\pi}_1)$ et $\pi \in Cyl(\hat{\pi}_2)$, i.e., $\hat{\pi}_1 \in pref(\pi)$ et $\hat{\pi}_2 \in pref(\pi)$, ce qui est vrai uniquement si $\hat{\pi}_1$ est préfixe de $\hat{\pi}_2$ (sans perdre de généralité). C'est impossible par définition de $Paths_{fin}^T(s)$.

Notation. On dit que π satisfait $\Diamond T$, i.e., $\pi \models \Diamond T$ ssi il existe un chemin fini $\hat{\pi} \in Path_{fin}^T(\mathcal{M})$ tel que $\pi \in Cyl(\hat{\pi})$.

On sait que si le système est actuellement en l'état $s \in S$, une unique mesure de probabilité \mathbb{P}_s existe sur $Paths(s)$ pour l'ensemble de ces cylindres. Dès lors, soit $s \in S$,

$$\begin{aligned} \mathbb{P}_s(\Diamond T) &= \mathbb{P}_s(\{\pi \in Paths(s) \mid \pi \models \Diamond T\}) \\ &= \mathbb{P}_s\left(\bigcup_{s_0 \dots s_n \in Paths_{fin}^T(s)} Cyl(s_0 \dots s_n)\right) \end{aligned}$$

Par la propriété 2.3, les cylindres des chemins finis $\hat{\pi} \in Paths_{fin}^T(s)$ sont disjoints. Alors, par définition de \mathbb{P}_s ,

$$\begin{aligned} \mathbb{P}_s(\Diamond T) &= \sum_{s_0 \dots s_n \in Paths_{fin}^T(s)} \mathbb{P}_s(Cyl(s_0 \dots s_n)) \\ &= \sum_{s_0 \dots s_n \in Paths_{fin}^T(s)} \Delta(s_0 \dots s_n) \end{aligned}$$

est la probabilité qu'un chemin commençant en l'état s satisfasse l'évènement $\Diamond T$, ou encore la probabilité d'atteindre un état de T depuis l'état s via un chemin dans \mathcal{M} .

Le problème d'accessibilité de la CM \mathcal{M} consiste à calculer la valeur de $\mathbb{P}_s(\Diamond T)$ pour tout état $s \in S$.

2.3.2 Résolution du problème

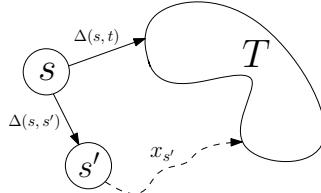
Afin de résoudre ce problème, il est nécessaire d'introduire la notion de *connexité à un sous-ensemble de sommets dans un graphe*.

Définition 2.7 (Connexité à un sous ensemble de sommets dans un graphe). Soient $G = (V, E)$, un graphe, $s \in V$, un sommet de G et $T \subseteq V$, un sous-ensemble de sommets de G . On dit que s est *connexe à T* ssi il existe un chemin $\pi = s_0s_1s_2s_3\dots$ dans ce graphe, i.e., une séquence de sommets telle que $(s_i, s_{i+1}) \in E$ pour tout $i \in \mathbb{N}$, où $s_0 = s$ et où il existe un $k \in \mathbb{N}$ tel que $s_k \in T$.

À présent, on définit $x_s = \mathbb{P}_s(\Diamond T)$ pour tout $s \in S$ de la façon suivante :

1. Si s est non-connexe à T dans le graphe sous-jacent $G^{\mathcal{M}}$, alors $x_s = 0$.
2. Si $s \in T$, alors $x_s = 1$.
3. Si $s \in S \setminus T$ et que la condition 1. n'est pas vérifiée, alors

$$x_s = \sum_{s' \in S \setminus T} \Delta(s, s') \cdot x_{s'} + \sum_{t \in T} \Delta(s, t)$$



- $\sum_{s' \in S \setminus T} \Delta(s, s') \cdot x_{s'}$ correspond à la probabilité que s atteigne le sous-ensemble d'états T en passant par un état intermédiaire $s' \in S \setminus T$.
- $\sum_{t \in T} \Delta(s, t)$ correspond à la probabilité que s atteigne le sous-ensemble d'états T en une seule étape.

Soit $n = |S|$. On obtient alors un système de n équations à n inconnues. De ce fait, résoudre x_s pour tout $s \in S$ revient à résoudre le *problème d'accessibilité* de la CM \mathcal{M} pour T .

Exemple 2.5 (Retour sur le dé de Knuth). Reprenons la CM $\mathcal{M}_{Kd} = (S, \Delta)$ de l'exemple 2.1. Lorsqu'on lance un dé à 6 faces, la probabilité d'obtenir n'importe quelle face de ce dé est de $\frac{1}{6}$. Dans \mathcal{M}_{Kd} , s_0 doit atteindre un des états finaux avec une probabilité $\frac{1}{6}$. On est donc intéressé de résoudre

$$\mathbb{P}_{s_0}(\Diamond T) \text{ pour tout } T \in \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$$

À l'aide du système défini dans cette sous-section, on calcule :

1. $\mathbb{P}_{s_0}(\Diamond \{1\})$

- $x_1 = 1$ car 1 est l'état cible.
- $x_{s_{2,3}} = x_{s_{4,5,6}} = x_{s_{4,5}} = x_{s'_{4,5,6}} = x_2 = x_3 = x_4 = x_5 = x_6 = 0$ car ces sommets n'atteignent pas le sommet 1 dans le graphe sous-jacent $G^{\mathcal{M}_{Kd}}$.
- $x_{s'_{1,2,3}} = \frac{1}{2}x_{s_{1,2,3}} + \frac{1}{2}$
- $x_{s_{1,2,3}} = \frac{1}{2}x_{s'_{1,2,3}} + \frac{1}{2}x_{s_{2,3}} = \frac{1}{2}x_{s'_{1,2,3}} = \frac{1}{4}(x_{s_{1,2,3}} + 1) \Leftrightarrow 4x_{s_{1,2,3}} = x_{s_{1,2,3}} + 1 \Leftrightarrow x_{s_{1,2,3}} = \frac{1}{3}$
- $x_{s_0} = \frac{1}{2}x_{s_{1,2,3}} + \frac{1}{2}x_{s_{4,5,6}} = \frac{1}{2}x_{s_{1,2,3}} = \frac{1}{6}$

2. $\mathbb{P}_{s_0}(\Diamond\{2\})$

- $x_2 = 1$ car 2 est l'état cible.
- $x_{s_{4,5,6}} = x_{s_{4,5}} = x_{s'_{4,5,6}} = x_1 = x_3 = x_4 = x_5 = x_6 = 0$ car ces sommets n'atteignent pas le sommet 2 dans le graphe sous-jacent $G^{\mathcal{M}_{Kd}}$.
- $x_{s_{2,3}} = \frac{1}{2}x_{s_3} + \frac{1}{2} = \frac{1}{2}x_{s_2}$
- $x_{s'_{1,2,3}} = \frac{1}{2}x_{s_{1,2,3}} + \frac{1}{2}x_{s_1} = \frac{1}{2}x_{s_{1,2,3}}$
- $x_{s_{1,2,3}} = \frac{1}{2}x_{s'_{1,2,3}} + \frac{1}{2} = \frac{1}{2}(\frac{1}{2}x_{s_{1,2,3}}) + \frac{1}{2}(\frac{1}{2}) = \frac{1}{4}x_{s_{1,2,3}} + \frac{1}{4} \Leftrightarrow \frac{3}{4}x_{s_{1,2,3}} = \frac{1}{4} \Leftrightarrow x_{s_{1,2,3}} = \frac{1}{3}$
- $x_{s_0} = \frac{1}{2}x_{s_{1,2,3}} + \frac{1}{2}x_{s_{4,5,6}} = \frac{1}{2}x_{s_{1,2,3}} = \frac{1}{6}$

3. $\mathbb{P}_{s_0}(\Diamond\{3\}) = \frac{1}{6}$ (idem que 2.).

Le comportement du système dans le cas où l'arc de droite est emprunté (i.e., le résultat du premier lancer de pièce est pile) en s_0 est symétrique au cas où l'arc de gauche est emprunté (cf. exemple 2.1). Dès lors, $\mathbb{P}_{s_0}(\Diamond\{4\}) = \mathbb{P}_{s_0}(\Diamond\{3\}) = \frac{1}{6}$ et $\mathbb{P}_{s_0}(\Diamond\{5\}) = \mathbb{P}_{s_0}(\Diamond\{2\}) = \frac{1}{6}$ et $\mathbb{P}_{s_0}(\Diamond\{6\}) = \mathbb{P}_{s_0}(\Diamond\{1\}) = \frac{1}{6}$. Le modèle simule donc bien un lancer de dé.

2.3.3 Généralisation matricielle

Le problème d'accessibilité pour la chaîne de Markov $\mathcal{M} = (S, \Delta)$ et le sous-ensemble d'états cibles $T \subseteq S$ peut se résoudre par un système d'équations formé par les valeurs de x_s , comme décrit ci-dessus. On veut maintenant définir un système matriciel équivalent possédant une solution unique.

Théorème 2.1. Soit $\mathcal{M} = (S, \Delta)$, une CM finie et $T \subseteq S$, un ensemble d'états cibles. On suppose que

- $S_{=0}$ est l'ensemble des états de S non-connexes à T .
- $S_{=1} = T$.
- $S_? = S \setminus (S_{=1} \cup S_{=0})$

Alors, le vecteur $(x_s)_{s \in S_?}$ est **l'unique solution** du système d'équations

$$x = Ax + b$$

où

- $A \in \mathbb{Q}^{|S_?| \times |S_?|}$ est la matrice de probabilité de transitions telle que $\forall i, j \in \{1, \dots, |S_?\}|$, $A_{i,j} = \Delta(s_i, s_j)$.
 $(Ax)_i$ correspond donc à la probabilité que s_i atteigne T via un état intermédiaire.
- $b \in \mathbb{Q}^{|S_?|}$ tel que $\forall i \in \{1, \dots, |S_?\}\}, b_i = \sum_{t \in S_?=1} \Delta(s_i, t)$.
 b_i correspond donc à la probabilité que s_i atteigne T en une étape.

Cette équation peut être réécrite sous forme d'un système d'équations linéaires

$$(\mathbb{1} - A)x = b$$

avec $\mathbb{1}$, la matrice unité de cardinalité $|S_?| \times |S_?|$, dans le but de résoudre le système avec des algorithmes de résolution de systèmes d'équations linéaires (e.g., avec le pivot de Gauss).

Exemple 2.6 (Problème d'accessibilité). On considère la chaîne de Markov $\mathcal{M}_{re} = (S, \Delta)$ de la figure 2.6 tel que $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$ et $T = \{s_5, s_6\}$. On est intéressé par $\mathbb{P}_s(\Diamond T)$ pour tout $s \in S$.

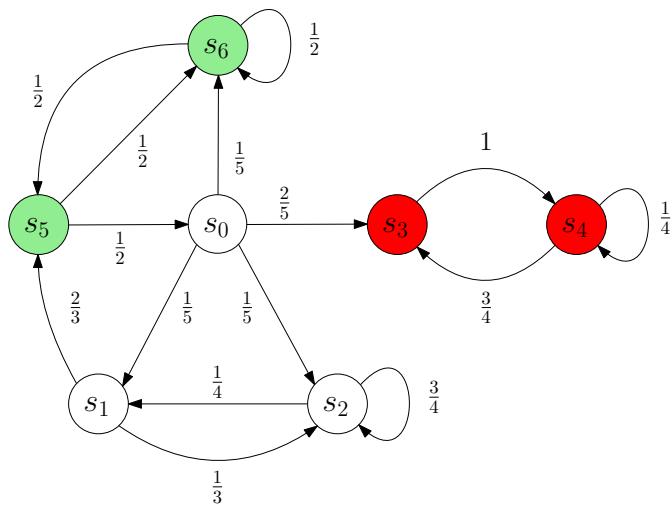


FIGURE 2.6 – Chaîne de Markov sur laquelle on va résoudre le problème d'accessibilité.

Par le fait que $T = \{s_5, s_6\}$, on a que $x_{s_5} = x_{s_6} = 1$. Le graphe sous-jacent $G^{\mathcal{M}_{re}}$ permet de détecter que les états s_3 et s_4 ne sont pas connexes à T . Dès lors, on a que

$$x_{s_3} = x_{s_4} = 0.$$

$$\begin{cases} x_{s_0} = \frac{1}{5}x_{s_1} + \frac{1}{5}x_{s_2} + \frac{2}{5}x_{s_3} + \frac{1}{5} \\ x_{s_1} = \frac{1}{3}x_{s_2} + \frac{2}{3} \\ x_{s_2} = \frac{1}{4}x_{s_1} + \frac{3}{4}x_{s_2} \\ x_{s_3} = 0 \\ x_{s_4} = 0 \\ x_{s_5} = 1 \\ x_{s_6} = 1 \end{cases} \iff \begin{cases} x_{s_0} - \frac{1}{5}x_{s_1} - \frac{1}{5}x_{s_2} - \frac{2}{5}x_{s_3} = \frac{1}{5} \\ x_{s_1} - \frac{1}{3}x_{s_2} = \frac{2}{3} \\ \frac{-1}{4}x_{s_1} + \frac{1}{4}x_{s_2} = 0 \\ x_{s_3} = 0 \\ x_{s_4} = 0 \\ x_{s_5} = 1 \\ x_{s_6} = 1 \end{cases}$$

Afin de résoudre ce système, il est utile de le passer sous forme matricielle :

$$\begin{pmatrix} 1 & \frac{-1}{5} & \frac{-1}{5} \\ 0 & 1 & \frac{-1}{3} \\ 0 & \frac{-1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x_{s_0} \\ x_{s_1} \\ x_{s_2} \end{pmatrix} = \begin{pmatrix} \frac{1}{5} \\ \frac{2}{3} \\ 0 \end{pmatrix}$$

Ce système d'équations linéaires peut se résoudre par la méthode du *pivot de Gauss*. Dès lors, la solution de ce système est :

$$x_{s_0} = \frac{3}{5}, \quad x_{s_1} = 1, \quad x_{s_2} = 1$$

2.4 Chaînes de Markov pondérées

Il arrive qu'une chaîne de Markov classique ne soit pas suffisante pour modéliser un phénomène, et plus particulièrement lorsque chaque transition a une répercussion différente sur un problème donné lié à ce système, e.g., la quantité d'énergie utilisée pour passer d'un état à un autre dans un système embarqué, la quantité d'argent dépensée lors d'une soirée au casino ou encore le temps écoulé pour atteindre une destination lors d'un voyage, etc.

Les chaînes de Markov sont alors enrichies par une fonction de coût. Quitter un état pour en rejoindre un autre sera considéré comme une action pondérée, i.e., chaque transition aura un coût en plus d'une probabilité. Dès lors, lorsqu'on s'intéresse aux chemins présents dans un tel modèle et plus particulièrement à leur coût, un problème classique fait son apparition : quel sera le coût pour atteindre un ensemble d'états cibles ? Les probabilités étiquetées sur les transitions de l'automate compléxifient le problème. Dans cette section, deux approches seront étudiées. *L'espérance du coût vers une cible ainsi que l'accessibilité limitée par un coût.*

Définition 2.8 (Chaîne de Markov pondérée). Une *chaîne de Markov pondérée*, notée **CMP**, est un tuple $\mathcal{M} = (S, \Delta, w)$ où

- S et Δ sont définis comme pour une CM à temps discret.
- $w : S \times S \rightarrow \mathbb{N}$ est la fonction de coût qui associe un poids entier positif à chaque transition, i.e., chaque transition (s, s') telle que $s, s' \in S$ et $\Delta(s, s') > 0$.

Remarque 2.2. La représentation d'une CMP est la représentation de la CM correspondante où les poids sont ajoutés à côté des probabilités sur les étiquettes des transitions.

Exemple 2.7 (Production énergétique d'un système embarqué équipé de panneaux solaires). Afin d'illustrer ces concepts, on se base sur un exemple personnel. Un système embarqué est alimenté par des panneaux solaires. Ceux-ci produisent chaque jour une certaine quantité d'énergie en fonction du climat : 5 kJ les jours ensoleillés, 3 kJ les jours légèrement nuageux, 2 kJ les jours moyennement nuageux et 1 kJ les jours fortement nuageux. Afin de modéliser ce système, on modélise d'abord la CM représentant les différents états du climat possibles chaque jour et on fixe ensuite la production énergétique sur les transitions en tant que poids. La CMP correspondante est illustrée à la figure 2.7

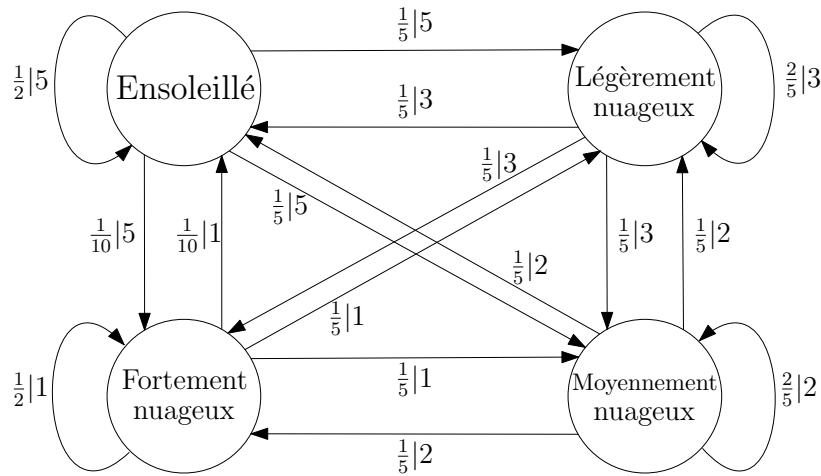


FIGURE 2.7 – Chaîne de Markov pondérée de la production énergétique du système équipé de panneaux solaires en fonction du climat

Définition 2.9 (Somme tronquée). Soient $\mathcal{M} = (S, \Delta, w)$, une CMP, $T \subseteq S$, un sous-ensemble d'états cibles et $\pi = s_0s_1s_2\cdots \in \text{Paths}(\mathcal{M})$, un chemin dans \mathcal{M} . La *somme tronquée* du chemin π dans \mathcal{M} pour T est le coût total des transitions entre les états du chemin π jusqu'à atteindre **pour la première fois** un des états cibles de T (si le chemin n'atteint jamais un sommet de T , on dira que la somme tronquée est infinie). Plus précisément, soit $TS^T : \text{Paths}(\mathcal{M}) \rightarrow \mathbb{Z} \cup \{\infty\}$, la fonction qui calcule la somme tronquée de tout chemin vers l'ensemble cible T . La somme tronquée du chemin π pour T est définie par

$$TS^T(\pi) = \begin{cases} \sum_{i=0}^{n-1} w(s_i, s_{i+1}) & \text{si } \forall i \in \{0, \dots, n-1\}, s_i \notin T \text{ et } s_n \in T \\ \infty & \text{si } \pi \not\models \Diamond T, \text{ i.e., } \forall i \ s_i \notin T \end{cases}$$

Exemple 2.8 (Somme tronquée dans le système équipé de panneaux solaires). Soit $\mathcal{M}_{sp} = (S, \Delta, w)$ la CMP de l'exemple 2.7. On veut calculer l'énergie produite par les panneaux solaires cette semaine jusqu'à ce que le climat ait été fortement nuageux. On suppose que le temps était ensoleillé lundi, légèrement nuageux mardi et mercredi, ensoleillé jeudi, fortement nuageux vendredi et samedi ainsi que moyennement nuageux dimanche.

Cette séquence forme un chemin $\pi = \text{ensoleillé}, \text{légèrement nuageux}, \text{légèrement nuageux}, \text{ensoleillé}, \text{fortement nuageux}, \text{fortement nuageux}, \text{moyennement nuageux}, \dots$ de \mathcal{M}_{sp} . Soit $T = \{\text{fortement nuageux}\} \subseteq S$, l'ensemble cible. $TS^T(\pi) = 5 + 3 + 3 + 5 + 1 = 17$. Le système a donc produit 17 kJ avant que le temps soit fortement nuageux.

2.4.1 Problème de l'espérance du coût de l'accessibilité

Dans cette section, on s'intéresse à l'espérance (cf. définition 1.5) du coût des chemins qui atteignent un sous-ensemble d'états cibles d'une chaîne de Markov pondérée et plus précisément du coût total attendu pour qu'un état fixé atteigne un sous-ensemble d'états cibles.

Définition 2.10 (Espérance du coût de l'accessibilité). Soient $\mathcal{M} = (S, \Delta, w)$, une CMP, $s \in S$, un état de s et $T \subseteq S$, un ensemble d'états cibles. On définit l'espérance $\mathbb{E}_s(TS^T)$, correspondant au *coût total attendu pour atteindre T depuis s* comme suit :

- Si $\mathbb{P}_s(\Diamond T) < 1$, alors $\mathbb{E}_s(TS^T) = \infty$.

En effet, soit $A = \{\pi \in \text{Paths}(s) \mid \pi \models \Diamond T\}$ et $B = \{\pi \in \text{Paths}(s) \mid \pi \not\models \Diamond T\}$.

On sait que $\mathbb{P}_s(\Diamond T) < 1 \implies \mathbb{P}_s(B) > 0$, et donc, on a forcément que

$$\mathbb{E}_s(TS^T) = \mathbb{P}_s(A) \cdot \mathbb{E}_s(TS^T \mid A) + \mathbb{P}_s(B) \cdot \underbrace{\mathbb{E}_s(TS^T \mid B)}_{=\infty} = \infty$$

- Sinon, i.e., si $\mathbb{P}_s(\Diamond T) = 1$, alors :

$$\mathbb{E}_s(TS^T) = \sum_{c=0}^{\infty} c \cdot \mathbb{P}_s(\{\pi \in \text{Paths}(s) \mid TS^T(\pi) = c\})$$

Proposition 2.1. Une définition équivalente de $\mathbb{E}_s(TS^T)$ dans le cas où $\mathbb{P}_s(\Diamond T) = 1$ peut être donné par la moyenne du coût des chemins $\hat{\pi}$ pondérée par la probabilité de ces chemins $\hat{\pi}$, i.e.,

$$\mathbb{E}_s(TS^T) = \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s)} \Delta(\hat{\pi}) \cdot TS^T(\hat{\pi})$$

Démonstration. Soit $\pi = s_0 \dots s_n \dots \in \text{Paths}(s)$ tel que $\pi \models \Diamond T$. On suppose que $s_0 = s$, $s_i \notin T$ pour tout $i \in \{0, \dots, n-1\}$ et $s_n \in T$. Soit $\hat{\pi} = s_0 \dots s_n \in \text{Paths}_{fin}^T(s)$. Alors, on a toujours que $TS^T(\pi) = TS^T(\hat{\pi})$.

(on peut ramener tout chemin infini π qui atteint T à un chemin fini $\hat{\pi}$ se terminant en la première occurrence d'un état de T dans π afin de calculer sa somme tronquée).

Dès lors, soit $c \in \mathbb{N} \cup \{\infty\}$,

$$\begin{aligned}
& c \cdot \mathbb{P}_s(\{\pi \in \text{Paths}(s) \mid TS^T(\pi) = c\}) \\
&= c \cdot \mathbb{P}_s(\{\hat{\pi} \in \text{Paths}_{fin}^T(s) \mid TS^T(\hat{\pi}) = c\}) \\
&= c \cdot \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s) \mid TS^T(\hat{\pi})=c} \Delta(\hat{\pi}) \\
&= \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s) \mid TS^T(\hat{\pi})=c} \Delta(\hat{\pi}) \cdot c \\
&= \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s) \mid TS^T(\hat{\pi})=c} \Delta(\hat{\pi}) \cdot TS^T(\hat{\pi}) \quad (\text{car } c = TS^T(\hat{\pi}))
\end{aligned}$$

Cela nous permet d'en déduire l'espérance de la façon suivante :

$$\begin{aligned}
\mathbb{E}_s(TS^T) &= \sum_{c=0}^{\infty} c \cdot \mathbb{P}_s(\{\pi \in \text{Paths}(s) \mid TS^T(\pi) = c\}) \\
&= \sum_{c=0}^{\infty} \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s) \mid TS^T(\hat{\pi})=c} \Delta(\hat{\pi}) \cdot TS^T(\hat{\pi}) \\
&= \sum_{\hat{\pi} \in \text{Paths}_{fin}^T(s)} \Delta(\hat{\pi}) \cdot TS^T(\hat{\pi})
\end{aligned}$$

□

Système d'équations linéaires

Soient $\mathcal{M} = (S, \Delta, w)$, une CMP **finie**, $s \in S$, un état de S et $T \subseteq S$, un ensemble d'états cibles. $\mathbb{E}_s(TS^T)$ peut se calculer via un système d'équations linéaires comme suit :

Soit $\text{succ}(s)$, l'ensemble des successeurs de s dans $G^{\mathcal{M}}$,

$$x_s = \begin{cases} \infty & \text{si } \mathbb{P}_s(\Diamond T) < 1 \\ 0 & \text{si } s \in T \\ \sum_{s' \in \text{succ}(s)} \Delta(s, s') \cdot (w(s, s') + x_{s'}) & \text{sinon} \end{cases}$$

La probabilité d'un chemin fini $\hat{\pi} = s_0s_1\dots t \in \text{Paths}_{fin}^T(s)$, à savoir $\Delta(s_0s_1\dots t)$, peut se décomposer en $\Delta(s_0, s_1) \cdot \Delta(s_1 \dots t)$ et la somme tronquée de ce chemin, à savoir $TS^T(s_0s_1\dots t)$, en $w(s_0, s_1) + TS^T(s_1 \dots t)$. Intuitivement, cela mène au fait que le coût moyen attendu pour atteindre l'ensemble d'états cibles depuis s en passant par un de ses successeurs s' peut se décomposer par le coût de la transition de s vers le successeur s' (à savoir $w(s, s')$) additionné à l'espérance du coût de l'accessibilité à l'ensemble d'états cibles depuis s' (à savoir $\mathbb{E}_{s'}(TS^T)$). Afin de calculer l'espérance du coût de l'accessibilité à l'ensemble d'états cibles depuis s , le tout est ensuite pondéré par la distribution de probabilité de s vers ses successeurs, et cela pour chacun de ses

successeurs (cf. figure 2.8).

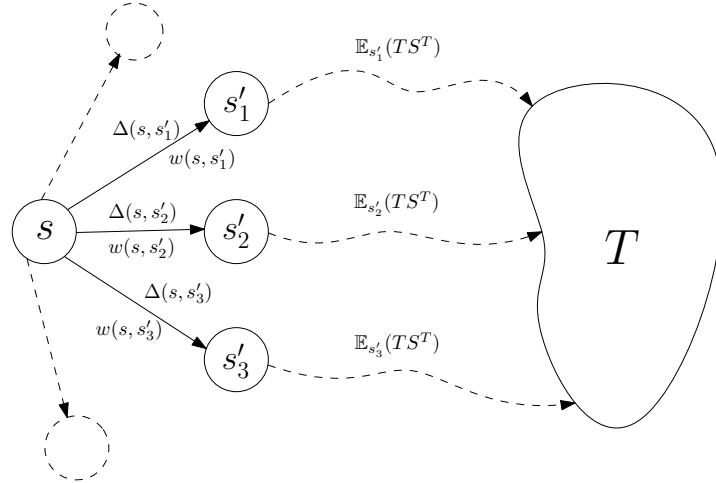


FIGURE 2.8 – Intuition de l’espérance du coût de l’accessibilité à l’ensemble d’états cibles T depuis l’état s , exprimé sous la forme
 $\mathbb{E}_s(TS^T) = \sum_{s' \in \text{succ}(s)} \Delta(s, s') \cdot (w(s, s') + \mathbb{E}_{s'}(TS^T))$

Généralisation matricielle

Afin de résoudre ce système, on souhaite à présent définir un système matriciel équivalent possédant une unique solution.

Théorème 2.2. Soit $S_{=1} = \{s \in S \mid \mathbb{P}_s(\Diamond T) = 1\}$. Le vecteur $(x_s)_{s \in S_{=1}}$ est l’unique solution du système d’équations

$$x = Ax + b$$

où

- $A \in \mathbb{Q}^{|S_{=1} \setminus T| \times |S_{=1} \setminus T|}$ tel que $\forall i, j \in \{1, \dots, |S_{=1} \setminus T|\}$, $A_{i,j} = \Delta(s_i, s_j)$.
 $(Ax)_i$ correspond donc à l’espérance moyenne attendue pour qu’un successeur ($\in S_{=1} \setminus T$) de $s_i \in S_{=1} \setminus T$ atteigne T .
- $b \in \mathbb{Q}^{|S_{=1} \setminus T|}$ tel que $\forall i \in \{1, \dots, |S_{=1} \setminus T|\}$, $b = \sum_{s' \in \text{succ}(s_i)} \Delta(s_i, s') \cdot w(s_i, s')$.
 b_i correspond donc à l’espérance du coût engendré par l’action de quitter l’état $s_i \in S_{=1} \setminus T$ pour rejoindre un de ses successeurs $\in S_{=1}$.

Intuition : Soient $s \in S_{=1} \setminus T$ et $i \in \{1, \dots, |S_{=1} \setminus T|\}$ tel que $s = s_i$,

$$\begin{aligned} x_s &= \sum_{s' \in \text{succ}(s)} \Delta(s, s') \cdot (w(s, s') + x_{s'}) \\ &= \sum_{s' \in \text{succ}(s) \cap S_{=1}} \Delta(s, s') \cdot x_{s'} + \sum_{s' \in \text{succ}(s)} \Delta(s, s') \cdot w(s, s') \\ &= \underbrace{\sum_{s' \in S_{=1} \setminus T} \Delta(s, s') \cdot x_{s'}}_{(Ax)_i} + \underbrace{\sum_{s' \in \text{succ}(s)} \Delta(s, s') \cdot w(s, s')}_{b_i} \quad (\text{car } x_s = 0 \text{ quand } s \in T) \end{aligned}$$

Exemple 2.9 (Espérance de la production énergétique du système de panneaux solaires). Cet exemple se base sur la CMP $\mathcal{M}_{sp} = (S, \Delta, w)$ de l'exemple 2.7. On souhaite connaitre l'espérance de la production énergétique des panneaux solaires lorsque le climat est ensoleillé jusqu'à ce que le climat devienne fortement nuageux et cela dans le but de connaitre la production énergétique attendue que peut avoir un tel système au moment où sa production est maximale jusqu'à son niveau de production minimale due au climat.

Soit $s \in S$ et $T = \text{fortement nuageux}$. On a $x_s = \mathbb{E}_s(TS^T)$. Le graphe $G^{\mathcal{M}_{sp}}$ est fortement connexe, on a donc $x_s \neq \infty$. Le système d'équations linéaires correspondant s'écrit :

$$\begin{aligned} & \begin{cases} x_e = \frac{1}{2}(5 + x_e) + \frac{1}{5}(5 + x_{ln}) + \frac{1}{5}(5 + x_{mn}) + \frac{1}{10}(5 + x_{fn}) \\ x_{ln} = \frac{2}{5}(3 + x_{ln}) + \frac{1}{5}(3 + x_e) + \frac{1}{5}(3 + x_{mn}) + \frac{1}{5}(3 + x_{fn}) \\ x_{mn} = \frac{2}{5}(2 + x_{mn}) + \frac{1}{5}(2 + x_e) + \frac{1}{5}(2 + x_{ln}) + \frac{1}{5}(2 + x_{fn}) \\ x_{fn} = 0 \end{cases} \\ \iff & \begin{cases} \frac{1}{2}x_e - \frac{1}{5}x_{ln} - \frac{1}{5}x_{mn} - \frac{1}{10}x_{fn} = 5 \\ -\frac{1}{5}x_e + \frac{3}{5}x_{ln} + \frac{1}{5}x_{mn} - \frac{1}{5}x_{fn} = 3 \\ -\frac{1}{5}x_e - \frac{1}{5}x_{ln} + \frac{3}{5}x_{mn} - \frac{1}{5}x_{fn} = 2 \\ x_{fn} = 0 \end{cases} \end{aligned}$$

Note : Par souci de visibilité, $e = \text{ensoleillé}$, $ln = \text{légèrement nuageux}$, $mn = \text{moyennement nuageux}$ et $fn = \text{fortement nuageux}$.

Afin de résoudre ce système, il est utile de le passer sous forme matricielle par le théorème 2.2 :

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{3}{5} & -\frac{1}{5} \\ -\frac{1}{5} & -\frac{1}{5} & \frac{3}{5} \end{pmatrix} \begin{pmatrix} x_e \\ x_{ln} \\ x_{mn} \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 2 \end{pmatrix}$$

On résout encore une fois ce système d'équations linéaires par le pivot de Gauss. La solution du système est :

$$x_e = 25, x_{ln} = 19.375, x_{mn} = 18.125, x_{fn} = 0$$

De ce fait, $\mathbb{E}_{\text{ensoleillé}}(TS^{\{\text{fortement nuageux}\}}) = 25 \text{ kJ}$.

2.4.2 Problème d'accessibilité limitée par un coût

Dans le problème précédent, on était intéressé par le coût moyen attendu pour qu'un état du système atteigne un sous-ensemble d'états cibles. Avec le *problème d'accessibilité limitée par un coût*, on s'intéresse plutôt à la probabilité que cet état atteigne le sous-ensemble d'états cibles avec un coût inférieur à un seuil fixé au préalable.

Définition 2.11 (Accessibilité limitée par un coût). Soient $\mathcal{M} = (S, \Delta, w)$, une

CMP, $s \in S$, un état, $T \subseteq S$, un sous-ensemble d'états cibles et $b \in \mathbb{Z}$, un seuil. La probabilité d'atteindre T depuis s limitée par un coût de seuil b est définie comme suit :

$$\mathbb{P}_s(TS^T \leq b) = \mathbb{P}_s(\{\pi \in Paths(s) \mid TS^T(\pi) \leq b\})$$

Notation. Soit $\mathcal{M} = (S, \Delta, w)$, une CMP. On désigne par $\mathbb{P}_s^{\mathcal{M}}$ la mesure de probabilité \mathbb{P}_s telle que $s \in S$ est un état de la CMP \mathcal{M} .

Réduction au problème d'accessibilité

Soient $\mathcal{M} = (S, \Delta, w)$, une CMP, $s \in S$, un état du système, $T \subseteq S$, un ensemble d'états cibles et $b \in \mathbb{Z}$, un seuil. Afin de résoudre $\mathbb{P}_s(TS^T \leq b)$, on réduit le problème à un simple problème d'accessibilité sur la CM $\mathcal{M}' = (S', \Delta')$ pour le sous-ensemble d'états cibles $T' \subseteq S'$, que l'on construit comme suit :

- S' est un ensemble de tuples (s, v) tel que $s \in S$ et $v \in \mathbb{N} \cup \{\perp\}$. On considère que $\perp > b$, avec $\perp + v = \perp \quad \forall v \in \mathbb{N}$. Intuitivement, on enregistre dans v le coût total du chemin en parcourant \mathcal{M} . Les états cibles sont donc les états de $T' = \{(s, v) \in S' \mid s \in T \wedge v \leq b\}$.
- $\Delta' : S' \times S' \rightarrow [0, 1]$ est la fonction de probabilité de transition définie comme suit :

$$\forall (s, v), (s', v') \in S',$$

$$\Delta'((s, v), (s', v')) = \begin{cases} \Delta(s, s') & \text{si } v' = v + w(s, s') \text{ et } v' \leq b \text{ ou} \\ & \text{si } v' = \perp \text{ et } v + w(s, s') > b \\ 0 & \text{sinon} \end{cases}$$

Dès lors, on a

$$\mathbb{P}_s^{\mathcal{M}}(\{\pi \in Paths(s) \mid TS^T(\pi) \leq b\}) = \mathbb{P}_{(s, 0)}^{\mathcal{M}'}(\Diamond T')$$

i.e., résoudre le problème d'accessibilité de s à T dans \mathcal{M} limitée par le coût de seuil b revient à résoudre le problème d'accessibilité de $(s, 0)$ à T' dans \mathcal{M}' .

Exemple 2.10 (Accessibilité limitée par un coût dans le système équipé de panneaux solaires). On reprend la CMP $\mathcal{M}_{sp} = (S, \Delta, w)$ de l'exemple 2.7. On souhaite connaître la probabilité que le système équipé de panneaux solaires produise **au moins** 8 kJ avant que le climat soit fortement nuageux, en supposant que l'installation commence à produire l'énergie un jour ensoleillé.

Pour ce faire, on va d'abord déterminer la probabilité que le système produise moins de 7 kJ , i.e., $\mathbb{P}_{\text{ensoleillé}}(TS^{\{\text{fortement nuageux}\}} \leq 7)$. On réduit le problème à un problème d'accessibilité en construisant la CM $\mathcal{M}'_{sp} = (S', \Delta')$ comme décrit ci-dessus (cf. figure 2.9).

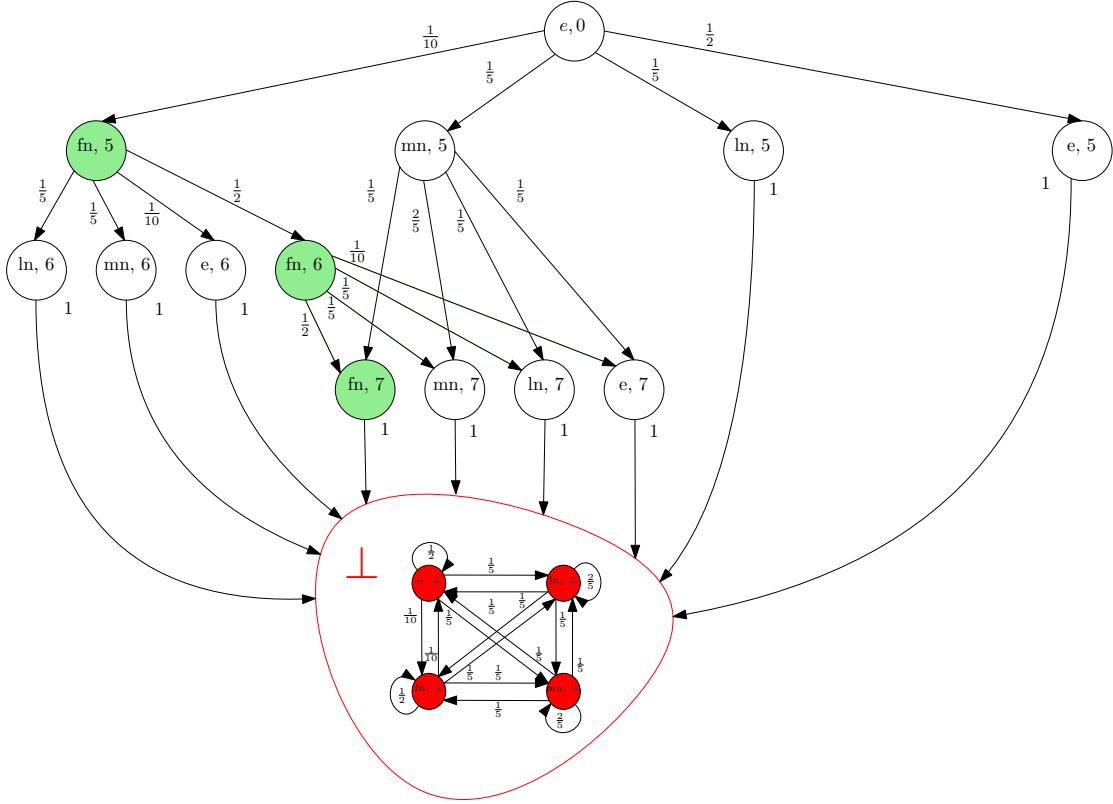


FIGURE 2.9 – CM \mathcal{M}'_{sp} construite à partir de \mathcal{M}_{sp} afin de résoudre $\mathbb{P}_{ensoleillé}(TS^{\{fortement nuageux\}} \leq 7)$

Par le théorème 2.1, on peut résoudre ce problème d’accessibilité de la façon suivante :

- $S_{=1} = T' = \{(fn, 5), (fn, 6), (fn, 7)\}$. Donc, $\forall s \in S_{=1}$, $\mathbb{P}_s^{\mathcal{M}'}(\Diamond T') = 1$.
- $S_{=0}$ est l’ensemble des états non-connexes à T' . On a donc que $S_{=0} = \{(ln, 5), (e, 5), (ln, 6), (mn, 6), (mn, 7), (ln, 7), (e, 7), (e, \perp), (ln, \perp), (mn, \perp), (fn, \perp)\}$ et que $\forall s \in S_{=0}$, $\mathbb{P}_s^{\mathcal{M}'}(\Diamond T') = 0$
- Il reste à déterminer $\mathbb{P}_s^{\mathcal{M}'}(\Diamond T') \forall s \in S_?$ tel que $S_? = S \setminus (S_{=1} \cup S_{=0})$. Pour ce faire, on résout le système matriciel suivant :

$$\begin{aligned} \begin{pmatrix} 0 & \frac{1}{5} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{(e,0)} \\ x_{(mn,5)} \end{pmatrix} + \begin{pmatrix} \frac{1}{10} \\ \frac{1}{5} \end{pmatrix} &= \begin{pmatrix} x_{(e,0)} \\ x_{(mn,5)} \end{pmatrix} \\ \iff \begin{pmatrix} 1 & -\frac{1}{5} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{(e,0)} \\ x_{(mn,5)} \end{pmatrix} &= \begin{pmatrix} \frac{1}{10} \\ \frac{1}{5} \end{pmatrix} \\ \iff \begin{pmatrix} x_{(e,0)} \\ x_{(mn,5)} \end{pmatrix} &= \begin{pmatrix} 0.14 \\ \frac{1}{5} \end{pmatrix} \end{aligned}$$

On a $\mathbb{P}_{(e,0)}^{\mathcal{M}'}(\Diamond T') = \mathbb{P}_{ensoleillé}^{\mathcal{M}}(TS^{\{fortement nuageux\}} \leq 7) = 0.14$. On revient au problème initial. On veut connaître $\mathbb{P}_{ensoleillé}(TS^{\{fortement nuageux\}} > 7)$. Comme $\mathbb{P}_{ensoleillé}$ est une mesure de probabilité, on a par la propriété 1.1

$$1 - \mathbb{P}_{ensoleillé}(TS^{\{fortement nuageux\}} \leq 7) = \mathbb{P}_{ensoleillé}(TS^{\{fortement nuageux\}} > 7)$$

On a $\mathbb{P}_{\text{ensoleillé}}(TS^{\text{fortement nuageux}} > 7) = \mathbb{P}_{\text{ensoleillé}}(TS^{\text{fortement nuageux}} \geq 8)$ car la production est entière, et donc, on a $\mathbb{P}_{\text{ensoleillé}}(TS^{\{\text{fortement nuageux}\}} \geq 8) = 0.86$.

Chapitre 3

Processus Décisionnels de Markov

Les chaînes de Markov ne permettent pas de modéliser des situations probabilistes impliquant des prises de décisions. Avec les *processus décisionnels de Markov*, de telles situations peuvent être modélisées. À chaque étape, le système se trouve en un état de prise de décision et une action doit être choisie. Une fois que l'action est choisie, le comportement du système devient probabiliste, comme lorsqu'on passe d'un état à un autre dans une chaîne de Markov. Dans un tel système, la transition d'un état à un autre dépend donc d'abord de l'action choisie, et ensuite de la distribution de probabilité définie par l'état actuel du système ainsi que par l'action choisie.

Ce chapitre est essentiellement inspiré du chapitre *Probabilistic Systems* du livre *Principles of model checking* [1] ainsi que de l'article *Variation on the Stochastic Path Problem* [6].

3.1 Définitions et propriétés

Définition 3.1 (Processus décisionnel de Markov). Un *processus décisionnel de Markov*, noté **PDM** est un tuple $\mathcal{M} = (S, A, \Delta)$ où

- S est un ensemble dénombrable d'états.
- A est un ensemble dénombrable d'actions. On dénote par $A(s) \in \mathcal{P}(A)$ l'ensemble des actions possibles lorsque le système se trouve dans l'état s . Pour tout état $s \in S$, on a toujours que $A(s) \neq \emptyset$.
- $\Delta : S \times A \times S \rightarrow [0, 1] \cap \mathbb{Q}$ est la fonction de transition telle que

$$\forall s \in S, \forall \alpha \in A(s), \sum_{s' \in S} \Delta(s, \alpha, s') = 1$$

et $\forall s, s' \in S, \forall \alpha \in A \setminus A(s), \Delta(s, \alpha, s') = 0$

Δ spécifie, pour tout état $s \in S$, la probabilité que le système passe de l'état s à s' lorsque l'action $\alpha \in A(s)$ est choisie.

Propriété 3.1. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM, $s \in S$ un état de \mathcal{M} et $\alpha \in A(s)$, une action possible lorsque le PDM \mathcal{M} est en s . Les contraintes imposées sur Δ assurent

que $\Delta_{s,\alpha}$ est une distribution de probabilité sur S (cf. définition 1.3) où

$$\Delta_{s,\alpha} : S \rightarrow [0, 1] \cap \mathbb{Q}, s' \mapsto \Delta(s, \alpha, s')$$

Soient $\mathcal{M} = (S, A, \Delta)$, un PDM et $s \in S$ un état de \mathcal{M} . Au moment où le système entre dans l'état s , un choix doit être effectué afin de passer à l'étape suivante. En effet, supposons que $A(s) = \{\alpha, \beta\}$. Une action possible de $\{\alpha, \beta\}$ doit être choisie, ce qui implique qu'on ne peut pas savoir quels seront les successeurs possibles de s tant qu'une action n'aura pas été choisie. Cela rend ce choix purement **non-déterministe**.

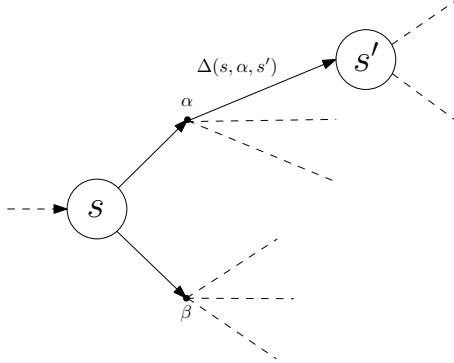


FIGURE 3.1 – Extrait du Processus décisionnel de Markov \mathcal{M}

On suppose alors que l'action α est sélectionnée. Lorsque α est choisie, le *successeur- α* de s est choisi aléatoirement selon la distribution de probabilité $\Delta_{s,\alpha}$. Donc, $s' \in S$ est successeur de s avec une probabilité de $\Delta(s, \alpha, s')$.

Notation. Soit $\mathcal{M} = (S, A, \Delta)$, un PDM. $Succ(s, \alpha)$ dénote l'ensemble des *successeurs- α* de l'état $s \in S$, i.e.,

$$Succ(s, \alpha) = \{s' \in S \mid \Delta(s, \alpha, s') > 0\}$$

Remarque 3.1. Si s' est l'unique successeur- α de s , alors, on a que $\Delta(s, \alpha, s') = 1$ et que $\Delta(s, \alpha, s^*) = 0$ pour tout $s^* \neq s'$. Dans ce cas, il n'est pas nécessaire de représenter l'actions α dans la représentation du PDM.

Exemple 3.1 (Agent dans un labyrinthe stochastique). Afin d'illustrer ces concepts, on se base sur un exemple personnel. Un agent ère dans un labyrinthe (cf. figure 3.2). Son but est d'atteindre les cases cibles t_1 et t_2 . Lorsqu'il se déplace dans le labyrinthe, l'agent doit décider, à chaque case, la direction dans laquelle se diriger à la prochaine étape. Cependant, l'agent ne peut pas prendre la décision de revenir sur ses pas, i.e., de se diriger dans la case dans laquelle il se trouvait à l'étape précédente, à moins d'y être contraint. On suppose que l'environnement du labyrinthe est stochastique. En effet, certaines cases du labyrinthe regorgent de pièges, ce qui peut contraindre l'agent à changer de direction. Dans le cas des cases dans lesquelles il y a la présence de pièges, l'agent a le droit de décider de faire demi-tour. On considère ici que les pièges sont simples et consistent à forcer l'agent à prendre une direction différente de celle issue de la prise de décision. La liste des pièges est la suivante :

- case (1, 1) : le piège a 20% de chance de s'activer.
- case (1, 3) : lorsque l'agent essaie de rejoindre la case (2, 3), i.e., lorsque l'agent décide de se diriger vers le sud, le piège a une chance sur trois de s'activer. Si l'agent choisit une autre direction, le piège a 10% de chance d'activation, mais ne constraint pas l'agent à se diriger dans la direction opposée à celle choisie.
- case (4, 3) : lorsque l'agent essaie de rejoindre la case t_1 , i.e., lorsque l'agent décide de se diriger vers le nord, le piège a une chance sur cinq de s'activer. Si l'agent choisit une autre direction, alors le piège a 10% de chance d'activation. Cependant, le piège ne constraint pas l'agent à revenir sur ses pas.

	1	2	3	4	5	6
1	\downarrow, \rightarrow	\leftarrow, \rightarrow	$\leftarrow, \downarrow, \rightarrow$	\leftarrow, \rightarrow	\leftarrow, \rightarrow	\leftarrow, \downarrow
2	\uparrow, \downarrow		\downarrow			\uparrow, \downarrow
3	\uparrow, \downarrow		t_1	t_2		\uparrow, \downarrow
4	\uparrow, \rightarrow	\leftarrow, \rightarrow	$\leftarrow, \uparrow, \downarrow$			\uparrow, \downarrow
5			\uparrow, \rightarrow	\leftarrow, \rightarrow	\leftarrow, \rightarrow	\leftarrow, \rightarrow
6						

FIGURE 3.2 – Labyrinthe dans lequel un agent tente de rejoindre les cases t_1 et t_2 . Les directions indiquées dans les cases sont les choix possibles de l'agent lorsqu'il se situe dans ces cases. Les cases en orange représentent les cases dans lesquelles des pièges sont présents, pouvant forcer l'agent à prendre une direction différente.

On peut modéliser cette situation sous la forme d'un PDM $\mathcal{M}_{\text{maze}} = (S, A, \Delta)$ (cf. figure 3.3). Comme l'agent ne peut pas prendre la décision de revenir sur ses pas, il n'est pas intéressant de considérer toutes les cases comme états du système. On considère donc uniquement les cases qui requièrent une prise de décision ainsi que les états cibles. On a donc $S = \{(1, 1), (1, 3), (4, 3), t_1, t_2\}$. On a ensuite 4 prises de décisions possibles au total. Il s'agit des actions de $A = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$.

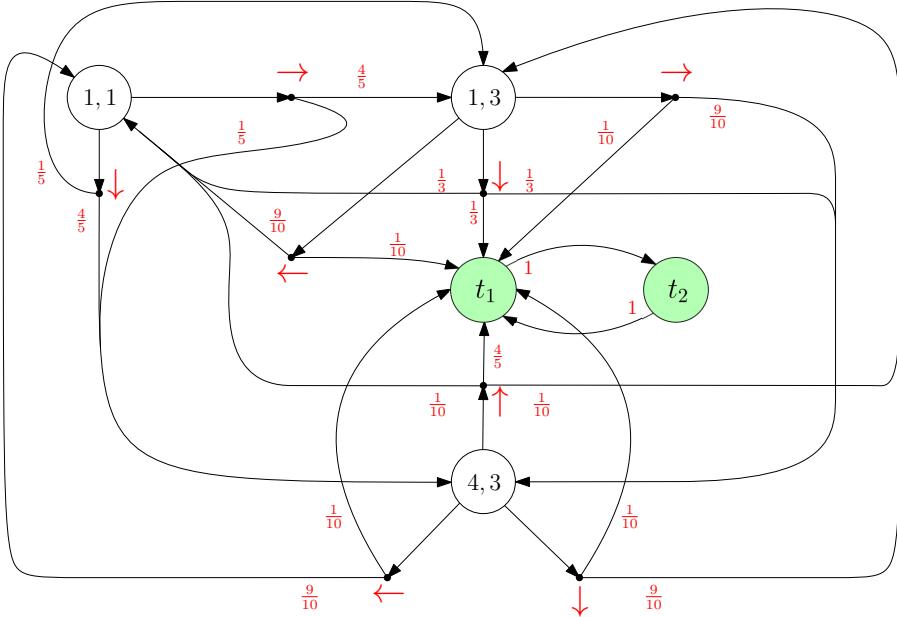


FIGURE 3.3 – PDM $\mathcal{M}_{\text{maze}}$.

En pratique, et à plus grande échelle, on peut considérer un véhicule autonome comme étant un agent, qui a connaissance de son environnement, correspondant au labyrinthe, via les cartes fournies par un GPS. Il est évident que l'environnement dans lequel ce véhicule se déplacerait est stochastique (e.g. voies bloquées, bouchons non renseignés par les données provenant du GPS et autres évènements dont l'agent ne peut pas avoir connaissance). Ces évènements sont considérés comme étant les pièges présents dans certaines cases du labyrinthe.

Propriété 3.2. Toute CM est un PDM tel que pour tout état s , un seul choix d'action est possible à chaque étape, i.e., $|A(s)| = 1$. La réciproque est également vraie, tout PDM possédant cette propriété (i.e., pour tout état s , $|A(s)| = 1$) est une CM. Dans ce cas, représenter les actions de $A(s)$ dans la représentation du PDM n'est plus pertinent et peut être omis.

Remarque 3.2. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM et $\alpha_1, \alpha_2 \in A$, deux actions. Si pour tout état $s \in S$, les successeurs- α_1 et successeurs- α_2 de s sont égaux et que pour chacun de ces successeurs s' , $\Delta(s, \alpha_1, s') = \Delta(s, \alpha_2, s')$, alors $\alpha_1 = \alpha_2$.

Définition 3.2 (Graphe sous-jacent d'un processus décisionnel de Markov). Un PDM $\mathcal{M} = (S, A, \Delta)$ induit un *graphe sous-jacent* (orienté) $G^{\mathcal{M}} = (V, E)$ où :

- Comme dans une chaîne de Markov, l'ensemble des sommets V du graphe correspond à l'ensemble des états du système. Par abus de langage, on dit que $V = S$ (cf. définition 2.2).
- E est l'ensemble des arcs du graphe. On a que l'arc $(s, s') \in E$ si il existe une action $\alpha \in A(s)$ telle que $\Delta(s, \alpha, s') > 0$.

Propriété 3.3. Tout PDM $\mathcal{M} = (S, A, \Delta)$ est fini si S et A sont finis. La taille de \mathcal{M} correspond au nombre de triplets (s, α, s') tels que $\Delta(s, \alpha, s') > 0$. Soit $G^{\mathcal{M}} = (S, E)$,

le graphe sous-jacent de \mathcal{M} . Par définition de E , on a

$$|\mathcal{M}| = \mathcal{O}(|E| |A|)$$

En effet,

$$\begin{aligned} |\mathcal{M}| &= |\{(s, \alpha, s') \in S \times A \times S \mid \Delta(s, \alpha, s') > 0\}| \\ &= |\{(s, \alpha, s') \in S \times A \times S \mid s' \in \text{Succ}(s, \alpha)\}| \\ &= \sum_{s \in S} \sum_{\alpha \in A(s)} |\text{Succ}(s, \alpha)| \\ &= \mathcal{O}\left(\sum_{s \in S} |A| |\text{succ}(s)|\right) \\ &= \mathcal{O}(|E| |A|) \end{aligned}$$

Exemple 3.2 (Graphe sous-jacent de l'agent évoluant dans un labyrinthe). Reprenons le PDM $\mathcal{M}_{\text{maze}}$ de l'exemple 3.1. Le graphe sous-jacent $G^{\mathcal{M}_{\text{maze}}}$ est donné à la figure 3.4 et $|\mathcal{M}_{\text{maze}}| = 20$.

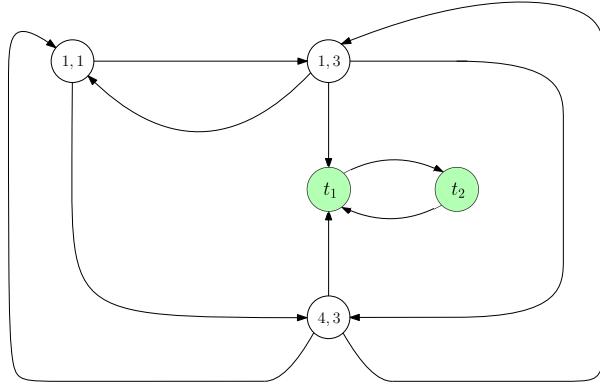


FIGURE 3.4 – Graphe sous jacent du PDM $\mathcal{M}_{\text{maze}}$.

3.2 Chemins et Stratégies de PDM

Définition 3.3 (Chemins dans un PDM). Soit $\mathcal{M} = (S, A, \Delta)$, un PDM. Un chemin de \mathcal{M} est une séquence infinie $s_0 \alpha_1 s_1 \alpha_2 s_2 \alpha_3 s_3 \dots \in (S \times A)^\omega$ où $\forall i \in \mathbb{N}$, $\Delta(s_i, \alpha_{i+1}, s_{i+1}) > 0$. Dès lors, soit la relation de transition

$$\rightarrow = \{(s, \alpha, s') \in S \times A \times S \mid \Delta(s, \alpha, s') > 0\},$$

on définit un chemin π de \mathcal{M} comme suit :

$$\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots$$

Soit $s \in S$, un état de \mathcal{M} . $\text{Paths}(s)$ dénote l'ensemble des chemins infinis de \mathcal{M} qui commencent en l'état s , et $\text{Paths}_{fin}(s)$ dénote l'ensemble des chemins finis de \mathcal{M}

commençant en l'état s . De la même façon, $Paths(\mathcal{M})$ dénote l'ensemble des chemins infinis de \mathcal{M} et $Paths_{fin}(\mathcal{M})$ dénote l'ensemble des chemins finis de \mathcal{M} , avec $Paths(\mathcal{M}) = \bigcup_{s \in S} Paths(s)$ et $Paths_{fin}(\mathcal{M}) = \bigcup_{s \in S} Paths_{fin}(s)$.

Exemple 3.3 (Chemin d'un agent dans un labyrinthe). Soit le PDM \mathcal{M}_{maze} de l'exemple 3.1. Dans cette situation, un chemin de \mathcal{M}_{maze} est en réalité un chemin infini possible pour l'agent lorsqu'il se déplace dans le labyrinthe. Le chemin

$$\pi = (1, 1) \xrightarrow{\downarrow} (4, 3) \xrightarrow{\uparrow} (t_1 \xrightarrow{\rightarrow} t_2 \xleftarrow{\leftarrow})^\omega \in Paths((1, 1))$$

est donc un chemin de \mathcal{M}_{maze} .

Ici, contrairement aux CM, les PDM ne sont pas équipés de σ -algèbre dû au choix d'action non-déterministe auquel le système est confronté à chaque étape (les choix ne suivent donc pas une distribution de probabilité). Étudier les probabilités des chemins d'un PDM est lié à l'étude de la résolution du non-déterminisme de ce PDM. Le non-déterminisme peut être résolu grâce à une *stratégie*.

Définition 3.4 (Histoire). Soit $\mathcal{M} = (S, A, \Delta)$, un PDM. Une *histoire* de \mathcal{M} est une séquence finie d'états $(s_0 \dots s_n) \in S^+$ telle que $\forall i \in \{1, \dots, n\}, \exists \alpha \in A(s_{i-1})$ telle que $\Delta(s_{i-1}, \alpha, s_i) > 0$. Une histoire d'un PDM est donc la succession d'états qui a amené l'état s_0 à l'état s_n dans un chemin de \mathcal{M} .

Exemple 3.4 (Histoire d'un agent dans un labyrinthe). Soit le PDM $\mathcal{M}_{maze} = (S, A, \Delta)$ de l'exemple 3.1. Une histoire de \mathcal{M}_{maze} correspond, dans cette situation, à la succession de cases qui forme un chemin fini de l'agent lorsqu'il se déplace dans le labyrinthe. Ainsi, $h = ((1, 1)(4, 3)(1, 3)(1, 1)) \in S^+$ est une histoire de \mathcal{M}_{maze} .

Définition 3.5 (Stratégie). Soit $\mathcal{M} = (S, A, \Delta)$, un PDM. Une *stratégie* pour \mathcal{M} est une fonction $\sigma : S^+ \rightarrow A$ qui sélectionne pour toute histoire $h = (s_0 \dots s_n)$ de \mathcal{M} , une action réalisable, i.e., $\sigma(s_0 \dots s_n) = \alpha \in A(s_n)$.

Le chemin $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots$ est appelé σ -chemin ssi $\alpha_i = \sigma(s_0 \dots s_{i-1})$ pour tout $i \in \mathbb{N}_0$.

Notons que les stratégies que l'on va utiliser dans ce document sont dites *pures*, i.e., non-randomisée. Il existe également un autre type de stratégies que nous ne traiterons pas dans ce document dont l'ensemble des images est l'ensemble des distributions de probabilités sur A , i.e., les stratégies σ telles que $\sigma : S^+ \rightarrow \mathcal{D}(A)$.

Supposons que σ est une stratégie pour le PDM \mathcal{M} . Alors, σ résout le non-déterminisme lié aux choix des actions dans \mathcal{M} . En effet, le comportement de \mathcal{M} sous les décisions de σ peut être formalisé sous la forme d'une CM \mathcal{M}^σ .

Définition 3.6 (CM d'un PDM induite par stratégie). Soit $\mathcal{M} = (S, A, \Delta)$, un PDM et σ , une stratégie pour \mathcal{M} . La CM \mathcal{M}^σ est donnée par $\mathcal{M}^\sigma = (S^+, \Delta_\sigma)$, où pour toute histoire $h = s_0 s_1 \dots s_n$ de \mathcal{M} ,

$$\Delta_\sigma(h, h.s_{n+1}) = \Delta(s_n, \sigma(h), s_{n+1})$$

Par construction, la CM \mathcal{M}^σ va prendre la forme d'une forêt (1 arbre par état de \mathcal{M}), où chaque chemin de \mathcal{M}^σ est en fait un σ -chemin de \mathcal{M} .

Propriété 3.4. Soit \mathcal{M} , un PDM et σ , une stratégie pour \mathcal{M} . On peut faire correspondre, pour tout σ -chemin de \mathcal{M} , un chemin de \mathcal{M}^σ , la CM induite par σ . En effet, soit π , un σ -chemin de \mathcal{M} tel que

$$\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots$$

Le chemin π^σ correspondant à π dans \mathcal{M}^σ est donné par

$$\pi^\sigma = \hat{\pi}_0 \hat{\pi}_1 \hat{\pi}_2 \dots$$

où $\hat{\pi}_n = s_0 \dots s_n$ est une histoire de \mathcal{M} telle que cette histoire est préfixe de π et se termine en l'état s_n après avoir parcouru $n + 1$ états. Dès lors, on a que

$$\pi = s_0 \xrightarrow{\sigma(\hat{\pi}_0)} s_1 \xrightarrow{\sigma(\hat{\pi}_1)} s_2 \xrightarrow{\sigma(\hat{\pi}_2)} \dots$$

Exemple 3.5 (Stratégie naïve d'un agent pour résoudre un labyrinthe dans un milieu stochastique). Soit le PDM $\mathcal{M}_{\text{maze}} = (S, A, \Delta)$ de l'exemple 3.1. On va définir une stratégie naïve qui va permettre à l'agent d'atteindre les cases cibles. Pour ce faire, on suppose qu'il existe une fonction $\text{étapes} : S \times \mathcal{P}(S) \rightarrow \mathbb{N} \cup \{\infty\}$ qui calcule le nombre minimum d'étapes nécessaires afin de passer d'un état à un sous-ensemble d'états cibles, i.e., le nombre d'arcs empreintés dans le graphe sous-jacent $G^{\mathcal{M}_{\text{maze}}}$ pour atteindre le sous-ensemble (e.g., $\text{étapes}((1, 1), \{t_1, t_2\}) = 2$). On définit également les fonctions $dernier : S^+ \rightarrow S$ et $avant-dernier : S^+ \rightarrow S$ qui calculent respectivement le dernier et avant-dernier état d'une histoire (par convention, soit $s \in S$, si $h = (s)$, alors $avant-dernier(h) = s$). E.g., soit $h = ((1, 1)(1, 3)(4, 3))$, une histoire de $\mathcal{M}_{\text{maze}}$, $dernier(h) = (4, 3)$ et $avant-dernier(h) = (1, 3)$.

On définit σ comme étant une stratégie qui va tenter de minimiser le nombre d'étapes pour atteindre les cases cibles. Cette stratégie va se servir de l'histoire afin d'éviter de choisir une action qui ferait repasser l'agent sur la case sur laquelle il était à l'étape précédente et ainsi d'éviter que l'agent "tourne en rond". En effet, si une telle action est choisie, alors la stratégie va de nouveau choisir l'action qui a amené l'agent sur la case actuelle, et cela risque donc de provoquer une "boucle" dans l'histoire.

Pour ce faire, soient h , une histoire de $\mathcal{M}_{\text{maze}}$, $T = \{t_1, t_2\}$, les cases cibles, $s = dernier(h)$, $s^* = avant-dernier(h)$ et

$$A^{\text{excl}} = \begin{cases} \{\alpha^*\}, \text{ avec } \alpha^* = \arg \max_{\alpha \in A(s)} \Delta(s, \alpha, s^*) & \text{si } |A(s)| \neq 1 \text{ et } \Delta(s, \alpha^*, s') \neq 0 \\ \emptyset & \text{sinon} \end{cases},$$

le singleton contenant l'action qui a le plus de chance d'amener l'agent vers l'avant-dernière case s^* depuis la case où il se situe actuellement (à savoir s). On définit σ

comme suit :

$$\sigma(h) = \arg \min_{\alpha \in A(s) \setminus A^{\text{excl}}} \text{étapes} \left(\max_{s' \in \text{Succ}(s, \alpha)} \Delta(s, \alpha, s'), T \right)$$

(Note : $\arg \max_{s' \in \text{Succ}(s, \alpha)} \Delta(s, \alpha, s')$ correspond à l'état en lequel le système a le plus de chance d'évoluer quand l'action α est choisie, i.e., à la case vers laquelle l'agent a le plus de chance de se diriger en choisissant la direction α).

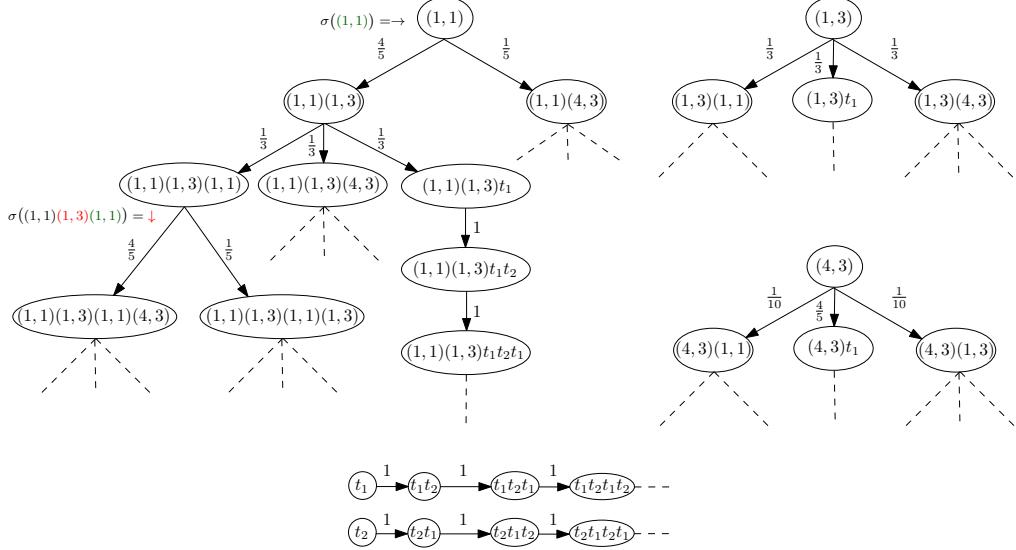


FIGURE 3.5 – CM $\mathcal{M}_{\text{maze}}^\sigma$ induite par la stratégie σ .

De cette façon,

$$\pi = (1, 1) \xrightarrow{\rightarrow} (1, 3) \xrightarrow{\downarrow} (1, 1) \xrightarrow{\downarrow} (4, 3) \xrightarrow{\uparrow} (t_1 \xrightarrow{\rightarrow} t_2 \xleftarrow{\leftarrow})^\omega$$

est un σ -chemin de $\mathcal{M}_{\text{maze}}$. En effet, d'abord, le système est en l'état $(1, 1)$. La stratégie choisit l'action \rightarrow car l'état $(1, 3)$ atteint t_1 en 1 étape dans le graphe sous-jacent. Ensuite, elle choisit l'action \downarrow pour accéder à la case t_1 , mais cela échoue et renvoie l'agent dans la case $(1, 1)$. La stratégie choisit alors l'action \downarrow (et plus \rightarrow) car l'action effectuée lorsque le système se trouvait en $(1, 3)$ à l'étape précédente a renvoyé l'agent dans la case $(1, 1)$. Cette stratégie induit une CM $\mathcal{M}_{\text{maze}}^\sigma$ (cf. figure 3.5). On peut faire correspondre le chemin π de $\mathcal{M}_{\text{maze}}$ au chemin π^σ de $\mathcal{M}_{\text{maze}}^\sigma$, où

$$\pi^\sigma = (1, 1) \ (1, 1)(1, 3) \ (1, 1)(1, 3)(1, 1) \ (1, 1)(1, 3)(1, 1)(4, 3) \ (1, 1)(1, 3)(1, 1)(4, 3)t_1 \ \dots$$

Comme la probabilité des chemins d'une CM est mesurable, toute stratégie σ d'un PDM \mathcal{M} permet de mesurer les σ -chemins de ce PDM en induisant une CM \mathcal{M}^σ sur laquelle un espace probabiliste est défini.

Notation. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM, $s \in S$, un état du système et σ , une stratégie définie pour \mathcal{M} . La mesure de probabilité induite par la stratégie σ sur les σ -chemins de $Paths(s)$ est dénotée par \mathbb{P}_s^σ .

Propriété 3.5. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM, σ , une stratégie définie pour \mathcal{M} , $s \in S$, un état de \mathcal{M} et $\pi = s_0\alpha_1s_1\alpha_2s_2\alpha_3 \dots \in Paths(s)$ tel que π est un σ -chemin. La mesure de probabilité $\mathbb{P}_s^\sigma(\{\pi\})$ est donc donnée par

$$\mathbb{P}_s^\sigma(\{\pi\}) = \mathbb{P}_{\hat{\pi}_0}^{\mathcal{M}^\sigma}(\{\pi^\sigma\})$$

où $\hat{\pi}_0 = s_0 = s$

Les stratégies que l'on a étudiées jusqu'ici sont des stratégies à mémoire infinie, car les éléments du domaine de la stratégie (i.e., les histoires) sont infinis. De ce fait, la CM \mathcal{M}^σ est infinie, et cela même si \mathcal{M} est finie. Intuitivement, l'état $s_0s_1\dots s_n$ de \mathcal{M}^σ représente la configuration du PDM \mathcal{M} lorsque le système est en s_n et que son histoire est $s_0\dots s_{n-1}$. Comme σ est dépendante de l'histoire du système, même si le système était en s_{n-1} dans une autre configuration, i.e., si l'histoire du système était différente, σ pourrait sélectionner une action différente (cf. exemple 3.5).

Les *stratégies à mémoire finies* apportent une indépendance au niveau des histoires du PDM pour lequel la stratégie est définie, ce qui permet d'induire des CM finies.

Définition 3.7 (Stratégie à mémoire finie). Soit $\mathcal{M} = (S, A, \Delta)$, un PDM. Une stratégie à mémoire finie est un tuple $\sigma = (Q, \sigma_\alpha, \delta, \delta_0)$ formant un automate fini (plus précisément, une machine de Moore) où

- Q est un ensemble fini de modes,
- $\sigma_\alpha : Q \times S \rightarrow A$ est une fonction qui sélectionne pour tout état $s \in S$ une action $\alpha \in A(s)$ en fonction du mode $q \in Q$ dans lequel est l'automate,
- $\delta : Q \times S \rightarrow Q$ est la fonction de transition,
- $\delta_0 : S \rightarrow Q$ est la fonction qui choisit le mode initial de l'automate en fonction d'un état $s \in S$ avec lequel il va être initialisé.

Définition 3.8 (Produit d'un PDM par une stratégie à mémoire finie). Soient $\mathcal{M} = (S, A, \Delta)$, un PDM et $\sigma = (Q, \sigma_\alpha, \delta, \delta_0)$, une stratégie à mémoire finie pour \mathcal{M} . Le produit du PDM \mathcal{M} par la stratégie σ est donné par

$$\mathcal{M} \times \sigma = \mathcal{M}^\sigma = (S \times Q, \Delta_\sigma)$$

où \mathcal{M}^σ est la CM induite par la stratégie σ et où pour tout états $s, s' \in S$ et pour tout modes de la stratégie $q, q' \in Q$,

$$\Delta_\sigma((s, q), (s', q')) = \begin{cases} \Delta(s, \sigma_\alpha(q, s), s') & \text{si } \delta(q, s) = q' \\ 0 & \text{sinon} \end{cases}$$

Exemple 3.6 (Stratégie à mémoire finie sur un PDM simple). Soit $\mathcal{M}_{\text{simple}} = (S, A, \Delta)$, un PDM (cf. figure 3.6). On suppose que le système est actuellement en l'état s . Soit

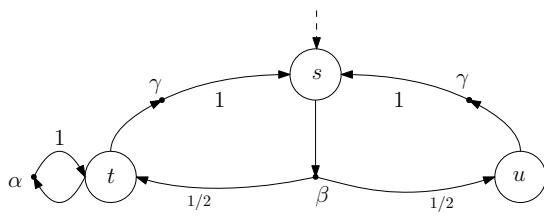


FIGURE 3.6 – PDM $\mathcal{M}_{\text{simple}}$.

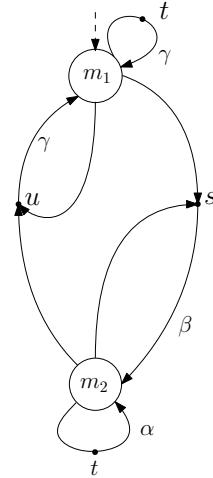


FIGURE 3.7 – Stratégie à mémoire finie σ .

$\sigma = (Q, \sigma_\alpha, \delta, \delta_0)$, une stratégie à mémoire finie (cf. figure 3.7). On suppose que la stratégie est toujours initialisée dans le mode m_1 , i.e., $\forall s \in S, \delta_0(s) = m_1$. La stratégie σ est une stratégie qui consiste à atteindre l'état t du système en passant au moins une fois par l'état s . Lorsqu'on applique la stratégie sur le PDM $\mathcal{M}_{\text{simple}}$, son comportement est le suivant :

1. Le PDM $\mathcal{M}_{\text{simple}}$ est actuellement en l'état s . On initialise la stratégie avec le mode m_1 . Comme on se situe en l'état s , $\sigma_\alpha(m_1, s) = \beta$ est l'action retournée par la stratégie. Le mode de la stratégie est mis à jour par la fonction de transition : $\delta(m_1, s) = m_2$.
2. Comme $\Delta(s, \beta, t) = \frac{1}{2}$ et $\Delta(s, \beta, u) = \frac{1}{2}$, le système a une chance sur deux d'évoluer en l'état t et une chance sur deux d'évoluer en l'état u . On suppose ici que le système évolue en l'état t .
3. Le PDM $\mathcal{M}_{\text{simple}}$ est actuellement en l'état t . Comme la stratégie est dans le mode m_2 , $\sigma_\alpha(m_2, t) = \alpha$ est l'action retournée par la stratégie et comme $\delta(m_2, t) = m_2$, la stratégie reste dans le mode m_2 .
4. La 3^{ème} étape est répétée indéfiniment.

Le σ -chemin de \mathcal{M} formé par le comportement de la stratégie ci-dessus est le suivant :

$$\pi = s \xrightarrow{\beta} (t \xrightarrow{\alpha})^\omega \in \text{Paths}(s)$$

La CM induite $\mathcal{M}_{\text{simple}}^\sigma$ est le produit du PDM avec la stratégie σ (cf. figure 3.8).

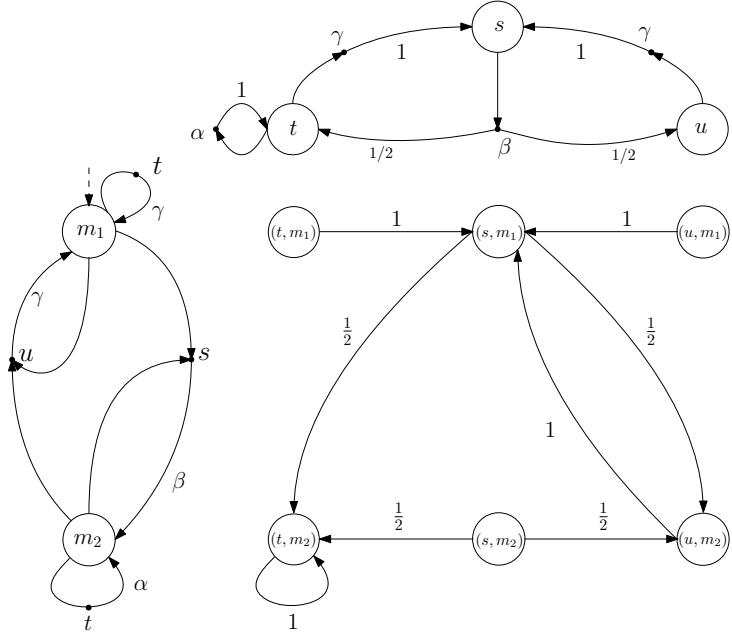


FIGURE 3.8 – La chaîne de Markov $\mathcal{M}_{\text{simple}}^\sigma$ induite par la stratégie σ est en réalité le produit de σ et $\mathcal{M}_{\text{simple}}$

Définition 3.9 (Stratégie sans mémoire). Soient $\mathcal{M} = (S, A, \Delta)$, un PDM et σ , une stratégie pour \mathcal{M} . La stratégie σ est dite *sans mémoire*, ou encore *simple*, si pour toutes histoires de \mathcal{M} $(s_0 \dots s_n)$ et $(t_0 \dots t_m)$ telles que $s_n = t_m$,

$$\sigma(s_0 \dots s_n) = \sigma(t_0 \dots t_m)$$

Alors, σ ne dépend pas de l'histoire complète du système, mais uniquement du dernier état dans lequel se trouve ce dernier. En effet, σ va toujours sélectionner la même action pour différentes histoires données se terminant en un même état. Dès lors, σ peut être vue comme étant une fonction

$$\sigma : S \rightarrow A$$

et on a donc $\sigma(s_n) = \sigma(s_0 \dots s_n) = \sigma(t_0 \dots t_m) = \sigma(t_m)$.

Propriété 3.6. On peut associer toute stratégie sans mémoire σ à une stratégie à mémoire finie $\sigma^* = (Q, \sigma_\alpha^*, \delta, \delta_0)$ ne possédant qu'un seul mode, i.e., telle que $|Q| = 1$. De ce fait, soit $q \in Q$, on a que $\forall s \in S$, $\sigma_\alpha^*(q, s) = \sigma(s)$, $\delta(q, s) = q$ et $\delta_0(s) = q$

Exemple 3.7 (Stratégie sans mémoire pour résoudre un labyrinthe dans un milieu stochastique). Soit le PDM $\mathcal{M}_{\text{maze}} = (S, A, \Delta)$ de l'exemple 3.1. Comme pour l'exemple 3.5, on va définir une stratégie σ pour atteindre les cases cibles du labyrinthe, mais on ne va plus considérer les histoires du système. De ce fait, on va définir une stratégie plus simple qu'à l'exemple 3.5, qui va naïvement tenter de minimiser le nombre d'étapes pour atteindre les cases cibles du labyrinthe. Soient $s \in S$, et $T = \{t_1, t_2\}$, on définit σ comme suit :

$$\sigma(s) = \arg \min_{\alpha \in A(s)} \text{étapes}\left(\arg \max_{s' \in \text{Succ}(s, \alpha)} \Delta(s, \alpha, s'), T \right)$$

Note : on suppose qu'arbitrairement, $\sigma((1, 1)) = (1, 3)$ vu que $\text{étapes}((1, 3), T) = \text{étapes}((4, 3), T)$.

Dès lors, le chemin

$$\pi = (1, 1) \xrightarrow{\cdot} (1, 3) \xrightarrow{\downarrow} (1, 1) \xrightarrow{\cdot} (1, 3) \xrightarrow{\downarrow} (t_1 \xrightarrow{\cdot} t_2 \xrightarrow{\cdot})^\omega$$

est un σ -chemin de $\mathcal{M}_{\text{maze}}$. La stratégie induit une CM $\mathcal{M}_{\text{maze}}^\sigma$ (cf. figure 3.9). Dès lors, on peut faire correpondre le chemin π de $\mathcal{M}_{\text{maze}}$ avec le chemin π^σ de $\mathcal{M}_{\text{maze}}^\sigma$, avec

$$\pi^\sigma = (1, 1) (1, 3) (1, 1) (1, 3) (t_1 t_2)^\omega$$

On peut donc mesurer la probabilité du chemin π :

$$\mathbb{P}_{(1,1)}^\sigma(\pi) = \mathbb{P}_{(1,1)}^{M_{\text{maze}}^\sigma}(\pi^\sigma) = \Delta(\pi^\sigma) = \frac{4}{5} \cdot \frac{1}{3} \cdot \frac{4}{5} \cdot \frac{1}{3} \cdot 1^\omega = \frac{16}{225}$$

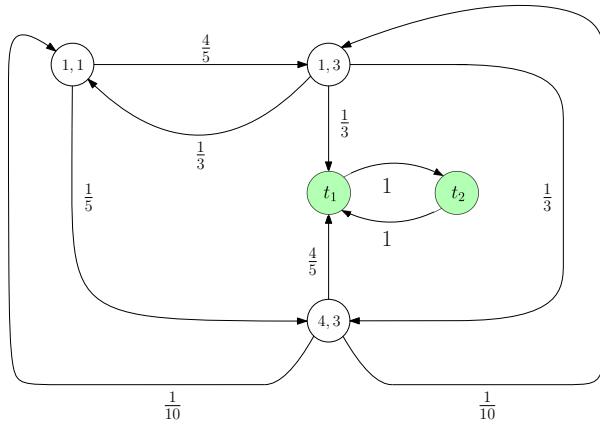


FIGURE 3.9 – CM $\mathcal{M}_{\text{maze}}^\sigma$ induite par la stratégie σ .

3.3 Problème d'accessibilité

Comme pour les CM, nous allons commencer par résoudre le problème d'accessibilité dans un PDM, i.e., étudier la probabilité d'atteindre un sous-ensemble d'états cibles du système, et cela pour chaque état. En effet, la résolution de ce problème est fondamentale pour résoudre les problèmes que nous allons rencontrer par la suite.

3.3.1 Énoncé du problème

Soient $\mathcal{M} = (S, A, \Delta)$, un PDM et $T \subseteq S$, un sous-ensemble d'états cibles. La mesure qui nous intéresse ici est la probabilité maximale (ou minimale pour le problème dual) d'atteindre un état du sous-ensemble cible T depuis un état du système $s \in S$.

Alors que dans les CMs on cherchait à connaître la probabilité d'atteindre un sous-ensemble d'états cibles depuis un état du système, les choix non-déterministes des PDMs

entraînent plusieurs espaces probabilistes et on cherche donc ici à connaître celui qui va maximiser la probabilité d'atteindre le sous-ensemble cible depuis un état du système. Plus strictement, soit $s \in S$, l'état depuis lequel on veut atteindre le sous-ensemble cible T ,

$$\mathbb{P}_s^{\max}(\diamond T) = \sup_{\sigma} \mathbb{P}_s^{\sigma}(\diamond T)$$

Ce supremum couvre toutes les stratégies définies pour \mathcal{M} , et leur nombre est possiblement infini.

Le *problème d'accessibilité* du PDM \mathcal{M} consiste à calculer la valeur de $\mathbb{P}_s^{\max}(\diamond T)$ pour tout état $s \in S$.

3.3.2 Résolution du problème

Ces probabilités maximales peuvent être calculées grâce à la résolution d'un programme linéaire. On va également voir qu'il est inutile de considérer toutes les stratégies possibles définies pour un PDM et qu'il suffit de ne considérer que le sous-ensemble de stratégies sans mémoire. En effet, il existe une stratégie sans mémoire qui maximise la probabilité d'atteindre T , et cela pour tout état du PDM considéré.

Théorème 3.1. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM, $s \in S$, un état de \mathcal{M} et $T \subseteq S$, un sous-ensemble d'états cibles. Le vecteur $(x_s)_{s \in S}$, avec $x_s = \mathbb{P}_s^{\max}(\diamond T)$ est l'unique solution du système d'équations suivant :

1. Si s est non-connexe à T dans le graphe sous-jacent $G^{\mathcal{M}}$, alors $x_s = 0$.
2. Si $s \in T$, alors $x_s = 1$.
3. Si $s \notin T$ et que la condition 1. n'est pas vérifiée, alors

$$x_s = \max_{\alpha \in A(s)} \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'}$$

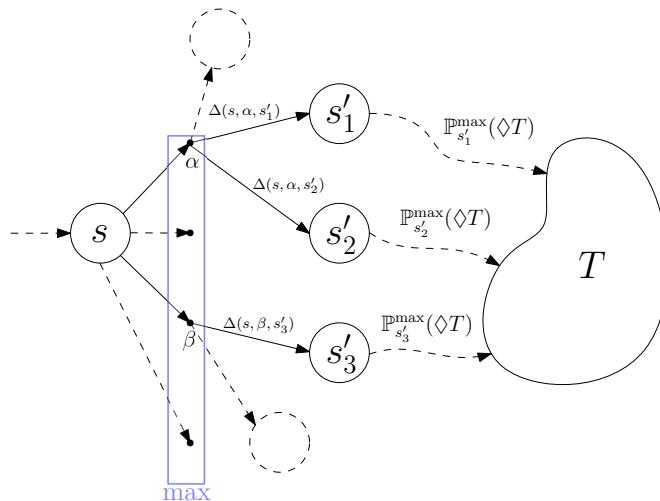


FIGURE 3.10 – Situation dans laquelle un état $s \notin T$ et s est connexe à T dans le graphe sous-jacent d'un PDM. .

On suppose que le système est en l'état s et que cet état s satisfait la condition 3. du théorème 3.1 (cf. figure 3.10). On suppose également que $x_{s'} = \mathbb{P}_{s'}^{\max}(\diamond T)$ pour tout $s' \neq s$. Afin de déterminer la valeur de x_s , on choisit l'action α qui maximise la probabilité d'atteindre T depuis un état intermédiaire s' . Comme on connaît déjà la probabilité maximale d'atteindre T depuis ces états intermédiaires, il reste à connaître la probabilité d'atteindre ces états intermédiaires depuis s . Celle-ci est donnée, pour tout successeur- α s' , par $\Delta(s, \alpha, s')$. La probabilité maximale que s atteigne T en passant par l'état intermédiaire s' est donc donnée par :

$$\underbrace{\Delta(s, \alpha, s')}_{\text{probabilité de passer de } s \text{ à } s'} \cdot \underbrace{x_{s'}}_{\text{probabilité maximale pour que } s' \text{ atteigne } T}$$

Il reste ensuite à faire la somme de ces probabilités pour tous les successeurs- α de s pour obtenir x_s .

Lemme 3.1. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM fini et $T \subseteq S$, un sous-ensemble d'états cibles. Il existe une stratégie sans mémoire σ telle que, pour tout $s \in S$,

$$\mathbb{P}_s^\sigma(\diamond T) = \mathbb{P}_s^{\max}(\diamond T)$$

Démonstration. En effet, soit $x_s = \mathbb{P}_s^{\max}(\diamond T)$. On va construire une stratégie sans mémoire σ telle que $\mathbb{P}_s^\sigma(\diamond T) = \mathbb{P}_s^{\max}(\diamond T)$. Pour ce faire, pour tout état s , on dénote par $A^{\max}(s)$ l'ensemble des actions $\alpha \in A(s)$ telles que $x_s = \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'}$. Donc, vu que $x_s = \mathbb{P}_s^{\max}(\diamond T)$, les actions de A^{\max} maximisent la probabilité d'atteindre un état de T depuis l'état s .

Construire une stratégie qui choisit arbitrairement un état de l'ensemble $A^{\max}(s)$ n'est pas suffisant. En effet, prenons par exemple un état s tel que $A^{\max}(s) = \{\alpha, \beta\}$ où $\Delta(s, \beta, t) = 1$ pour un certain $t \in T$ et où choisir l'action α ne permet jamais au système d'atteindre T , i.e., $\Delta(s, \alpha, s) = 1$ (cf. figure 3.11).

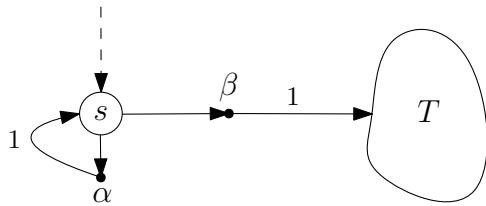


FIGURE 3.11 – La stratégie σ qui choisit toujours β permet d'assurer au système de toujours atteindre un état de T dans la CM \mathcal{M}^σ . Tandis que si la stratégie choisit toujours α , alors s n'atteindra jamais T dans \mathcal{M}^σ .

Une sélection d'action est donc requise et cette dernière doit assurer l'accessibilité de T dans la CM induite par la stratégie σ . On considère le PDM \mathcal{M}^{\max} qui correspond au PDM \mathcal{M} où on a supprimé les actions $\beta \in A(s) \setminus A^{\max}(s)$ de $A(s)$ pour tout état s connexe à T dans $G^{\mathcal{M}}$. Par définition, \mathbb{P}_s^{\max} n'est pas affectée par cette simplification de \mathcal{M} .

Pour tout s tel que s est connexe à T dans le graphe sous-jacent $G^{\mathcal{M}^{\max}}$, on dénote par $\|s\|$ la longueur du *plus court chemin* de s à n'importe quel état de T dans $G^{\mathcal{M}^{\max}}$

(i.e., le nombre minimal d'arêtes empreintées par un chemin de $G^{\mathcal{M}^{\max}}$ pour atteindre T). Intuitivement, calculer la valeur de $\|s\|$ permet d'éviter que σ choisisse des actions qui empêcheront d'atteindre T (cf. α dans la figure 3.11).

- $\|s\| = 0$ ssi $s \in T$.
- Soit $n \in \mathbb{N}_0$. Par induction sur n , on définit $\sigma(s)$ pour tout état s connexe à T dans $G^{\mathcal{M}^{\max}}$ et tel que $\|s\| = n$. La stratégie choisit une action $\sigma(s) \in A^{\max}(s)$ telle qu'il existe un successeur- $\sigma(s)$ s' où $\Delta(s, \sigma(s), s') > 0$, avec s' connexe à T dans $G^{\mathcal{M}^{\max}}$ et $\|s'\| = n - 1$. Une action $\sigma \in A(s)$ est choisie arbitrairement pour les états s qui ne sont pas connexes à T dans $G^{\mathcal{M}^{\max}}$.

On construit de cette façon la stratégie sans mémoire σ . Comme σ est sans mémoire, la CM induite par stratégie \mathcal{M}^σ est finie. On sait qu'il existe un système d'équations linéaires qui possède une solution unique $(y_s)_{s \in S}$, telle que $y_s = \mathbb{P}_s^{\mathcal{M}^\sigma}(\Diamond T)$ pour tout état s (cf. section 2.3) :

1. Si s est non-connexe à T dans $G^{\mathcal{M}^\sigma}$, alors $y_s = 0$.
2. Si $s \in T$, alors $y_s = 1$.
3. Si $s \notin T$ est que la condition 1. n'est pas vérifiée, alors

$$y_s = \sum_{s' \in S} \Delta(s, \sigma(s), s') \cdot y_{s'}$$

Comme x_s résout également ce système, on a

$$\mathbb{P}_s^\sigma(\Diamond T) = y_s = x_s = \mathbb{P}_s^{\max}(\Diamond T)$$

□

Il est possible de réécrire le système d'équations linéaires du théorème 3.1 sous forme d'un programme linéaire afin de calculer $\mathbb{P}_s^{\max}(\Diamond T)$ pour tout état s .

Théorème 3.2. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM fini et $T \subseteq S$, un sous-ensemble d'états cibles. Le vecteur $(x_s)_{s \in S}$ avec $x_s = \mathbb{P}_s^{\max}(\Diamond T)$ est l'unique solution optimale du programme linéaire suivant :

$$\min \sum_{s \in S} x_s$$

sous les contraintes suivantes :

$$\begin{aligned} x_s &= 1 && \forall s \in T, \\ x_s &= 0 && \forall s \notin T \text{ tels que } s \text{ n'est pas connexe à } T \text{ dans } G^{\mathcal{M}}, \\ x_s &\geq \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'} && \forall \alpha \in A(s) \text{ et } \forall s \notin T \text{ tels que } s \text{ est connexe à } T \text{ dans } G^{\mathcal{M}}. \\ 0 \leq x_s &\leq 1 && \forall s \in S \end{aligned}$$

Soient $(x_s)_{s \in S}$, la solution du système d'équations linéaires du théorème 3.1 et $(y_s)_{s \in S}$, une solution optimale du programme linéaire (dit PL) du théorème 3.2.

Tout d'abord, on remarque que les deux premières conditions sont équivalentes, dans le système d'équations linéaires ainsi que dans le PL. On a donc que x_s satisfait toutes les contraintes du PL. En effet, comme pour tout état $s \notin T$ connexe à T dans $G^{\mathcal{M}}$, $x_s = \max_{\alpha \in A(s)} \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'}$, on a forcément que $x_s \geq \sum_{s' \in S} \Delta(s, \alpha, s') \cdot x_{s'}$ pour tout $\alpha \in A(s)$. Comme $(y_s)_{s \in S}$ est une solution optimale du PL, $\sum_{s \in S} x_s \geq \sum_{s \in S} y_s$.

Ensuite, comme pour tout $s \notin T$ connexe à T dans $G^{\mathcal{M}}$, $y_s \geq \sum_{s' \in S} \Delta(s, \alpha, s') \cdot y_{s'}$ pour tout $\alpha \in A(s)$ et comme $\sum_{s \in S} y_s$ est minimale, on a intuitivement que la contrainte est serrée avec $y_s = \max_{\alpha \in A(s)} \sum_{s' \in S} \Delta(s, \alpha, s') \cdot y_{s'}$. Ceci est équivalent à la condition 3. du système d'équations linéaires. Donc, $(y_s)_{s \in S}$ est solution du système d'équations linéaires. Comme cette solution est unique, $(x_s)_{s \in S} = (y_s)_{s \in S}$ et $(y_s)_{s \in S}$ est unique.

Corollaire 3.1. Soient $\mathcal{M} = (S, A, \Delta)$, un PDM fini, $T \subseteq S$, un sous-ensemble d'états cibles et $s \in S$, un état de \mathcal{M} , $\mathbb{P}_s^{\max}(\Diamond T)$ peut être calculée en temps polynomial en la taille de \mathcal{M} .

Exemple 3.8 (Résolution du problème d'accessibilité dans un labyrinthe stochastique). Soit $\mathcal{M}_{\text{maze}} = (S, A, \Delta)$, le PDM de l'exemple 3.1. On suppose que l'agent commence à se déplacer en haut à gauche du labyrinthe, i.e., depuis la case (1, 1). On souhaite connaître la probabilité maximale que l'agent atteigne les cases cibles t_1 et t_2 depuis cette case (1, 1), i.e., $\mathbb{P}_{(1,1)}^{\max}(\Diamond \{t_1, t_2\})$. Pour ce faire, il faut résoudre le problème d'accessibilité pour le PDM $\mathcal{M}_{\text{maze}}$. On définit le programme linéaire suivant :

$$\min x_{(1,1)} + x_{(1,3)} + x_{(4,3)} + x_{t_1} + x_{t_2}$$

sous les contraintes

$$\begin{aligned} x_{(1,1)} &\geq \frac{4}{5}x_{(1,3)} + \frac{1}{5}x_{(4,3)} & (\alpha = \rightarrow) \\ x_{(1,1)} &\geq \frac{1}{5}x_{(1,3)} + \frac{4}{5}x_{(4,3)} & (\alpha = \downarrow) \\ x_{(1,3)} &\geq \frac{1}{3}x_{(1,1)} + \frac{1}{3}x_{(4,3)} + \frac{1}{3}x_{t_1} & (\alpha = \downarrow) \\ x_{(1,3)} &\geq \frac{9}{10}x_{(1,1)} + \frac{1}{10}x_{t_1} & (\alpha = \leftarrow) \\ x_{(1,3)} &\geq \frac{9}{10}x_{(4,3)} + \frac{1}{10}x_{t_1} & (\alpha = \rightarrow) \\ x_{(4,3)} &\geq \frac{1}{10}x_{(1,1)} + \frac{1}{10}x_{(1,3)} + \frac{4}{5}x_{t_1} & (\alpha = \uparrow) \\ x_{(4,3)} &\geq \frac{9}{10}x_{(1,1)} + \frac{1}{10}x_{t_1} & (\alpha = \leftarrow) \\ x_{(4,3)} &\geq \frac{9}{10}x_{(1,3)} & (\alpha = \downarrow) \\ 1 &\geq x_{(1,1)}, x_{(1,3)}, x_{(4,3)} \geq 0 \\ x_{t_1} &= x_{t_2} = 1 \end{aligned}$$

Selon le théorème 3.2, il existe une solution unique $(x_s)_{s \in S}$ à ce PL tel que

$$(x_s)_{s \in S} = \mathbb{P}_s^{\max}(\Diamond \{t_1, t_2\})$$

On réécrit ce PL sous sa forme canonique :

$$\min_{(x_s)_{s \in S \setminus \{t_1, t_2\}}} (1 \ 1 \ 1) \begin{pmatrix} x_{(1,1)} \\ x_{(1,3)} \\ x_{(4,3)} \end{pmatrix}$$

sous les contraintes

$$\begin{pmatrix} 1 & -\frac{4}{5} & -\frac{1}{5} \\ 1 & \frac{-1}{5} & \frac{-4}{5} \\ \frac{-1}{3} & 1 & \frac{-1}{3} \\ \frac{-9}{10} & 1 & 0 \\ 0 & 1 & \frac{-9}{10} \\ \frac{-1}{10} & \frac{-1}{10} & 1 \\ \frac{-9}{10} & 0 & 1 \\ 0 & \frac{-9}{10} & 1 \end{pmatrix} \begin{pmatrix} x_{(1,1)} \\ x_{(1,3)} \\ x_{(4,3)} \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ \frac{1}{3} \\ \frac{1}{10} \\ \frac{1}{10} \\ \frac{4}{5} \\ \frac{1}{10} \\ \frac{1}{10} \end{pmatrix}$$

et tel que $1 \geq x_{(1,1)}, x_{(1,3)}, x_{(4,3)} \geq 0$

Cela permet de résoudre le problème grâce à des algorithmes de résolution de PL (e.g., par la méthode du simplexe). Dans notre cas, on résout le problème à l'aide du module `linprog` de la bibliothèque `scipy`, en `python` avec la méthode du simplexe. Dès lors, la solution de ce système est donc $x_{(1,1)} = x_{(1,3)} = x_{(4,3)} = x_{t_1} = x_{t_2} = 1$ (ce qui est logique, car tout sommet du graphe sous-jacent $G^{\mathcal{M}_{\text{maze}}}$ est connexe à $\{t_1, t_2\}$).

Certains problèmes nécessitent de savoir si un état $s \in S$ peut toujours atteindre T , i.e., si $\mathbb{P}_s^{\max}(\Diamond T) = 1$. Bien que résoudre un PL où toutes les variables sont continues peut se faire en temps polynomial, il n'est pas nécessaire de résoudre un PL pour déterminer les états s qui vérifient $\mathbb{P}_s^{\max}(\Diamond T) = 1$. En effet, cela peut être calculé à l'aide du graphe sous-jacent $G^{\mathcal{M}}$ par un algorithme de parcours de graphe, polynomial en $|\mathcal{M}|$. Dès lors, on peut diminuer le nombre de variables du PL, en prétraitant le PDM pour déterminer les états $s \in S$ qui vérifient $x_s = \mathbb{P}_s^{\max}(\Diamond T) = 1$.

3.4 Le problème du plus court chemin stochastique

On va maintenant étudier des PDM enrichis de poids sur chacune de leurs actions. En effet, chaque action aura maintenant un coût. Cela permet d'apporter un contenu encore plus riche aux systèmes qui modélisent les situations probabilistes. Grâce à de tels PDM, il est désormais possible de modéliser des problèmes comme par exemple les jeux de hasard, où chaque action aura un coût (e.g., parier peut être vu comme une action) ou encore modéliser des situations dans le domaine de la finance où tout investissement a un coût et où les bénéfices (ou pertes) engendrés par cet investissement sont incertains, etc.

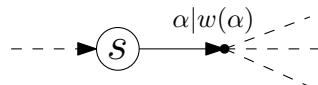
Lorsqu'on parle de coûts, une question naturelle apparaît : **Comment minimiser ces coûts ? Quelle stratégie employer ?** C'est le sujet dont on va traiter dans cette

section. Le but de cette section est de définir des stratégies qui vont minimiser les coûts des chemins de PDM pour atteindre des états cibles. Le problème du plus court chemin stochastique est naturellement en relation avec le problème des plus courts chemins dans un graphe, car là aussi on cherche à atteindre des noeuds avec un coût minimum et à déterminer quel chemin a le coût minimal. Le problème est néanmoins très différent dû aux probabilités pour passer d'un état à un autre dans un PDM. Dès lors, on ne peut pas définir une stratégie qui assure un coût minimal fixe, mais on peut aborder deux problèmes : *le problème de l'espérance du plus court chemin stochastique* et *le problème des plus courts chemins stochastiques de taille limitée*. Ces problèmes sont évidemment en relation avec les problèmes abordés au chapitre précédent pour les CM, à savoir le problème de l'espérance du coût de l'accessibilité ainsi que celui de l'accessibilité limitée par un coût.

Définition 3.10 (PDM pondéré). Un PDM *pondéré*, noté **PDMP**, est un tuple $\mathcal{M} = (S, A, \Delta, w)$, où

- S, A, Δ sont définis comme pour un PDM classique.
- $w : A \rightarrow \mathbb{N}_0$ est la fonction de coût qui associe un poids entier strictement positif à **chaque action**, i.e., $\forall \alpha \in A, \exists n \in \mathbb{N}_0, w(\alpha) = n$.

Remarque 3.3. On représente un PDMP de la même façon qu'un PDM, mais chaque action α est étiquetée par son coût.



Définition 3.11 (CMP induite par stratégie). Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP et σ , une stratégie pour \mathcal{M} . La stratégie σ induit une CMP $\mathcal{M}^\sigma = (S_\sigma, \Delta_\sigma, w_\sigma)$ telle que

- S_σ et Δ_σ sont induits de la même façon que pour les CM classiques induites par stratégie,
- $\forall s, s' \in S_\sigma$, si $\Delta_\sigma(s, s') > 0$, alors $w_\sigma(s, s') = w(\sigma(s))$.

Définition 3.12 (Somme Tronquée d'un PDMP). Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $T \subseteq S$, un sous-ensemble d'états cibles et $\pi = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \dots \in \text{Paths}(\mathcal{M})$, un chemin dans \mathcal{M} . La somme tronquée du chemin π dans \mathcal{M} pour T est le coût total des actions nécessaires pour enchaîner les états du chemin π jusqu'à atteindre **pour la première fois** un des états cibles de T (si le chemin n'atteint jamais un sommet de T , on dira que la somme tronquée est infinie). Plus strictement, soit $TS^T : \text{Paths}(\mathcal{M}) \rightarrow \mathbb{N} \cup \infty$, la somme tronquée de π pour atteindre T dans \mathcal{M} . Celle-ci est définie par

$$TS^T(\pi) = \begin{cases} \sum_{i=1}^n w(\alpha_i) & \text{si } \forall i \in \{0, \dots, n-1\}, s_i \notin T \text{ et } s_n \in T \\ \infty & \text{si } \nexists i \in \mathbb{N} \text{ tels que } s_i \in T \end{cases}$$

Exemple 3.9 (PDMP de l'agent se dirigeant dans un labyrinthe stochastique). Soit $\mathcal{M}_{\text{maze}} = (S, A, \Delta)$, le PDM de l'exemple 3.1. On souhaite étudier la longueur des chemins de ce PDM, i.e., le nombre de cases que doit traverser l'agent pour atteindre les cases cibles t_1 et t_2 . Pour ce faire, on va enrichir le système d'une fonction de coût $w : A \rightarrow \mathbb{N}_0$ qui va représenter le nombre de cases que l'agent traverse en effectuant une action. Afin de respecter le labyrinthe de l'exemple (cf. figure 3.2), il est utile d'introduire de nouveaux états et de nouvelles actions. Dès lors, soit $\mathcal{M}_{\text{maze}} = (S', A', \Delta, w)$, le PDMP représentant l'agent qui cherche à atteindre les cases t_1 et t_2 dans le labyrinthe stochastique (cf. figure 3.12). On a $S' = S \cup \{(1, 2), (1, 2)', (2, 1), (2, 3), (1, 4), (4, 2), (5, 3)\}$ et $A' = A \cup \{\downarrow, \uparrow, \leftarrow, \rightarrow\}$. On définit la fonction de coût comme suit : Soit $\alpha \in A$,

$$w(\alpha) = \begin{cases} 1 & \text{si } \alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\} \\ 4 & \text{si } \alpha \in \{\downarrow, \uparrow\} \\ 10 & \text{si } \alpha \in \{\leftarrow, \rightarrow\} \end{cases}$$

ce qui va nous permettre de calculer le nombre de cases parcourues par l'agent.

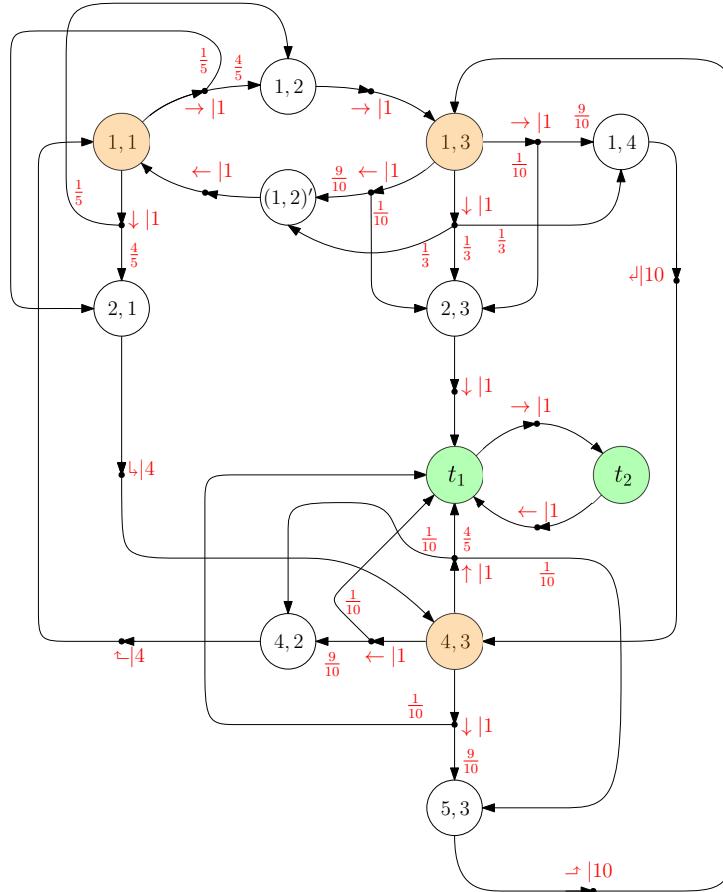


FIGURE 3.12 – PDMP représentant l'agent parcourant le labyrinthe stochastique de la figure 3.2. Les probabilités étiquetées sur les transitions pour lesquelles $\Delta(s, \alpha, s') = 1$ pour un certain $\alpha \in A(s)$ sont omises sur la représentation du PDMP par souci de lisibilité.

En effet, soit $\pi \in Paths((1, 1))$ tel que

$$\pi = (1, 1) \xrightarrow{\downarrow} (2, 1) \xrightarrow{\downarrow} (4, 3) \xrightarrow{\uparrow} (t_1 \xrightarrow{\rightarrow} t_2 \xleftarrow{\leftarrow})^\omega$$

Le nombre de cases traversées par l'agent pour atteindre les cases cibles lorsqu'il emprunte ce chemin est donné par la somme tronquée de ce chemin :

$$\begin{aligned} TS^T(\pi) &= w(\downarrow) + w(\downarrow) + w(\uparrow) \\ &= 1 + 4 + 1 \\ &= 6 \end{aligned}$$

Ainsi, l'agent traverse 6 cases en empreignant ce chemin avant d'atteindre les cases cibles t_1 ou t_2 .

3.4.1 Minimiser l'espérance de la longueur des chemins

Le problème que l'on va traiter dans cette section est celui de *l'espérance du plus court chemin stochastique* d'un PDMP. On s'intéresse donc au coût des chemins qui atteignent un sous-ensemble d'états cibles. Plus particulièrement, selon un état du système donné, le problème consiste à définir une stratégie qui va minimiser le coût espéré pour atteindre le sous-ensemble d'états cibles à partir de cet état.

Définition 3.13 (L'espérance du plus court chemin stochastique). Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $T \subseteq S$, un sous-ensemble d'états cibles, $s \in S$, un état de \mathcal{M} et $l \in \mathbb{Q}$, un seuil (longueur maximale). Le problème consiste à décider si il existe une stratégie σ pour laquelle

$$\mathbb{E}_s^\sigma(TS^T) \leq l$$

où $\mathbb{E}_s^\sigma(TS^T) = \mathbb{E}_s^{\mathcal{M}^\sigma}(TS^T)$ est l'espérance du coût de l'accessibilité de l'état s vers le sous ensemble T dans la CMP \mathcal{M}^σ induite par la stratégie σ (cf. section 2.10).

Théorème 3.3. Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP fini et $T \subseteq S$, un sous-ensemble d'états cibles. Soient $S_{=1} = \{s \in S \mid \mathbb{P}_s^{\max}(\Diamond T) = 1\}$ et le vecteur $(x_s)_{s \in S}$. On définit le PL suivant :

$$\max \sum_{s \in S_{=1}} x_s$$

sous les contraintes

$$\begin{aligned} x_s &= 0 & \forall s \in T \\ x_s &= \infty & \forall s \in S \text{ tel que } \mathbb{P}_s^{\max}(\Diamond T) < 1 \\ x_s &\leq w(\alpha) + \sum_{s' \in S \setminus T} \Delta(s, \alpha, s') \cdot x_{s'} & \forall \alpha \in A(s) \text{ et } \forall s \in S \setminus T \text{ tel que } \mathbb{P}_s^{\max}(\Diamond T) = 1 \end{aligned}$$

Ce PL a une solution optimale unique $(v_s)_{s \in S}$. On peut ensuite construire une stratégie

sans mémoire optimale

$$\sigma(s) = \arg \min_{\alpha \in A(s)} \left(w(\alpha) + \sum_{s' \in S \setminus T} \Delta(s, \alpha, s') \cdot v_{s'} \right)$$

telle que

$$\mathbb{E}_s^\sigma(TS^T) = \min_\sigma \mathbb{E}_s^\sigma(TS^T)$$

i.e., σ minimise $\mathbb{E}_s^\sigma(TS^T)$. Dès lors, $v_s = \mathbb{E}_s^\sigma(TS^T)$ pour tout état $s \in S$.

Remarque 3.4. Soit $A^{\min}(s) = \arg \min_{\alpha \in A(s)} w(\alpha) + \sum_{s' \in S \setminus T} \Delta(s, \alpha, s') \cdot v_{s'}$. Le scénario dans lequel $|A^{\min}(s)| > 1$ ne pose pas de problème (cf. figure 3.11) car $w : A \rightarrow \mathbb{N}_0$, i.e., la fonction de poids va toujours associer un coût > 0 , ce qui permet d'éviter de choisir des actions qui bloqueraient le système. Cependant, dans le cas où la fonction de poids permet d'associer un coût nul à une action, il est nécessaire de définir une stratégie d'une façon similaire à la démonstration du lemme 3.1.

Intuitivement, le fait que $\sum_{s \in S} v_s$ est maximal mène au fait que la 3^e contrainte du PL est serrée avec

$$v_s = \min_{\alpha \in A(s)} \left(w(\alpha) + \sum_{s' \in S \setminus T} \Delta(s, \alpha, s') \cdot v_{s'} \right)$$

Par la stratégie sans mémoire optimale σ définie ci-dessus, on va montrer que la solution du PL est également la solution du système d'équations linéaires qui résout le problème d'espérance du coût de l'accessibilité de \mathcal{M}^σ (cf. le système d'équations linéaires défini à la section 2.4.1).

1. Vu qu'on fixe x_s à 0 pour tout $s \in T$, on a bien que $(\mathbb{E}_t^\sigma)_{t \in T}(TS^T) = 0$.
2. Fixer x_s à ∞ pour tout $s \in S$ tel que $\mathbb{P}_s^{\max}(\Diamond T) < 1$ permet d'assurer que $\mathbb{E}_s^\sigma(TS^T) = \infty$ pour tout s tel que $\mathbb{P}_s^\sigma(\Diamond T) < 1$. En effet,

$$\mathbb{P}_s^\sigma(\Diamond T) < 1 \implies \mathbb{P}_s^{\max}(\Diamond T) < 1$$

Afin de s'en convaincre, on prend la contraposée de cette implication, i.e., $\mathbb{P}_s^{\max}(\Diamond T) = 1 \implies \mathbb{P}_s^\sigma(\Diamond T) = 1$. Cela est vrai. On suppose que $\mathbb{P}_s^{\max}(\Diamond T) = 1$. Cela signifie par le lemme 3.1 qu'il existe une stratégie sans mémoire σ^* telle que $\mathbb{P}_s^{\sigma^*}(\Diamond T) = \mathbb{P}_s^{\max}(\Diamond T) = 1$. Cela implique forcément que $\mathbb{P}_s^\sigma(\Diamond T) = 1$, car sinon, on aurait que $\mathbb{E}_s^{\sigma^*}(TS^T) < \mathbb{E}_s^\sigma(TS^T) = \infty$, ce qui est faux par définition de σ (σ minimise $\mathbb{E}_s^\sigma(TS^T)$). Vu qu'on a fixé x_s à ∞ pour tout $s \in S$ tel que $\mathbb{P}_s^{\max}(\Diamond T) < 1$, et vu qu'on vient de prouver que $\mathbb{P}_s^\sigma(\Diamond T) < 1 \implies \mathbb{P}_s^{\max}(\Diamond T) < 1$, on a bien que $v_s = \mathbb{E}_s^\sigma(TS^T) = \infty$ quand $\mathbb{P}_s^\sigma(\Diamond T) < 1$.

3. Pour tous les autres états $s \in S \setminus T$:

$$\begin{aligned}
\mathbb{E}_s^\sigma(TS^T) &= \min_{\alpha \in A(s)} \left(w(\alpha) + \sum_{s' \in S \setminus T} \Delta(s, \alpha, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \right) \\
&= \min_{\alpha \in A(s)} \left(w(\alpha) + \sum_{s' \in S} \Delta(s, \alpha, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \right) \quad (\text{avec } (\mathbb{E}_t^\sigma)_{t \in T}(TS^T) = 0) \\
&= \min_{\alpha \in A(s)} \left(w(\alpha) + \sum_{s' \in \text{Succ}(s, \alpha)} \Delta(s, \alpha, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \right) \\
&\qquad\qquad\qquad (\text{car } \Delta(s, \alpha, s') = 0 \text{ pour tout } s' \notin \text{Succ}(s, \alpha)) \\
&= w(\alpha^*) + \sum_{s' \in \text{Succ}(s, \alpha^*)} \Delta(s, \alpha^*, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \\
&\qquad\qquad\qquad (\text{par définition de } \sigma, \text{ avec } \alpha^* = \sigma(s)) \\
&= w(\alpha^*) \cdot \sum_{s' \in S} \Delta(s, \alpha^*, s') + \sum_{s' \in \text{Succ}(s, \alpha^*)} \Delta(s, \alpha^*, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \\
&\qquad\qquad\qquad (\text{car } \Delta_{s, \alpha^*} \text{ est une distribution de probabilité sur } S) \\
&= \sum_{s' \in \text{Succ}(s, \alpha^*)} \Delta(s, \alpha^*, s') \cdot w(\alpha^*) + \sum_{s' \in \text{Succ}(s, \alpha^*)} \Delta(s, \alpha^*, s') \cdot \mathbb{E}_{s'}^\sigma(TS^T) \\
&= \sum_{s' \in \text{Succ}(s, \alpha^*)} \Delta(s, \alpha^*, s') \cdot (w(\alpha^*) + \mathbb{E}_{s'}^\sigma(TS^T)) \\
&= \sum_{s' \in \text{succ}(s)} \Delta_\sigma(s, s') \cdot (w_\sigma(s, s') + \mathbb{E}_{s'}^\sigma(TS^T))
\end{aligned}$$

Par 1., 2. et 3., la solution du PL respecte bien la définition de l'espérance du coût de l'accessibilité de la CMP induite par la stratégie σ .

Corollaire 3.2. La stratégie sans mémoire optimale qui résout le problème de l'espérance du plus court chemin stochastique pour un PDM \mathcal{M} peut être construite en temps polynomial en la taille de \mathcal{M} .

Exemple 3.10 (Espérance du plus court chemin de l'agent dans le labyrinthe stochastique). Soient $\mathcal{M}_{\text{maze}} = (S, A, \Delta, w)$, le PDMP de l'exemple 3.9 représentant l'agent qui se déplace de la labyrinthe de la figure 3.2 et $T = \{t_1, t_2\}$, les cases cibles de ce labyrinthe. On suppose que l'agent se situe dans la case $(1, 1)$ et se déplace vers les cases t_1 ou t_2 . Comme le labyrinthe est stochastique, on souhaite connaître l'espérance du plus court chemin stochastique pour atteindre les cases t_1 ou t_2 depuis la case $(1, 1)$ et déterminer si il existe une stratégie qui résout ce problème et qui assure une espérance de la longueur du chemin inférieure à 10. Pour ce faire, on définit le PL suivant (par le théorème 3.3) :

$$\max_{s \in S \setminus T} x_{(1,1)} + x_{(1,2)} + x_{(1,2')} + x_{(1,3)} + x_{(1,4)} + x_{(2,1)} + x_{(2,3)} + x_{(4,2)} + x_{(4,3)} + x_{(5,3)}$$

sous les contraintes

$$\begin{array}{lll}
x_{(1,1)} & \leq w(\rightarrow) + \frac{4}{5}x_{(1,2)} + \frac{1}{5}x_{(2,1)} & x_{(1,1)} - \frac{4}{5}x_{(1,2)} - \frac{1}{5}x_{(2,1)} \leq 1 \\
x_{(1,1)} & \leq w(\downarrow) + \frac{1}{5}x_{(1,2)} + \frac{4}{5}x_{(2,1)} & x_{(1,1)} - \frac{1}{5}x_{(1,2)} - \frac{4}{5}x_{(2,1)} \leq 1 \\
x_{(1,3)} & \leq w(\downarrow) + \frac{1}{3}x_{(1,2)'} + \frac{1}{3}x_{(1,4)} + \frac{1}{3}x_{(2,3)} & -\frac{1}{3}x_{(1,2)'} + x_{(1,3)} - \frac{1}{3}x_{(1,4)} - \frac{1}{3}x_{(2,3)} \leq 1 \\
x_{(1,3)} & \leq w(\leftarrow) + \frac{9}{10}x_{(1,2)'} + \frac{1}{10}x_{2,3} & -\frac{9}{10}x_{(1,2)'} + x_{(1,3)} - \frac{1}{10}x_{(2,3)} \leq 1 \\
x_{(1,3)} & \leq w(\rightarrow) + \frac{9}{10}x_{(1,4)} + \frac{1}{10}x_{2,3} & x_{(1,3)} - \frac{9}{10}x_{(1,4)} - \frac{1}{10}x_{(2,3)} \leq 1 \\
x_{(4,3)} & \leq w(\uparrow) + \frac{1}{10}x_{(4,2)} + \frac{1}{10}x_{(5,3)} & -\frac{1}{10}x_{(4,2)} + x_{(4,3)} - \frac{1}{10}x_{(5,3)} \leq 1 \\
x_{(4,3)} & \leq w(\leftarrow) + \frac{9}{10}x_{(4,2)} & -\frac{9}{10}x_{(4,2)} + x_{(4,3)} \leq 1 \\
x_{(4,3)} & \leq w(\downarrow) + \frac{9}{10}x_{(5,3)} & \iff x_{(4,3)} - \frac{9}{10}x_{(5,3)} \leq 1 \\
x_{(1,2)} & \leq w(\rightarrow) + x_{(1,3)} & x_{(1,2)} - x_{(1,3)} \leq 1 \\
x_{(1,2)'} & \leq w(\leftarrow) + x_{(1,1)} & -x_{(1,1)} + x_{(1,2)'} \leq 1 \\
x_{(1,4)} & \leq w(\downarrow) + x_{(4,3)} & x_{(1,4)} - x_{(4,3)} \leq 10 \\
x_{(2,1)} & \leq w(\downarrow) + x_{(4,3)} & x_{(2,1)} - x_{(4,3)} \leq 4 \\
x_{(2,3)} & \leq w(\downarrow) & x_{(2,3)} \leq 1 \\
x_{(4,2)} & \leq w(\leftarrow) + x_{(1,1)} & -x_{(1,1)} + x_{(4,2)} \leq 4 \\
x_{(5,3)} & \leq w(\rightarrow) + x_{(1,3)} & -x_{(1,3)} + x_{(5,3)} \leq 10
\end{array}$$

Note : Le graphe sous-jacent $G^{\mathcal{M}_{maze}}$ permet d'affirmer que $\mathbb{P}_s^{\max}(\diamond T) = 1 \quad \forall s \in S$ car tous les sommets de ce graphe sont connexes à T . Dès lors, on peut résoudre ce problème avec la méthode du simplexe. La solution optimale $(v_s)_{s \in S \setminus T}$ de ce PL est

$$\begin{array}{lll}
v_{(1,1)} = 9.83050847 & v_{(1,2)} = 10.72881356 & v_{(1,2)'} = 10.83050847 \\
v_{(1,3)} = 9.72881356 & v_{(1,4)} = 14.3559322 & v_{(2,1)} = 8.3559322 \\
v_{(2,3)} = 1 & v_{(4,2)} = 13.83050847 & v_{(4,3)} = 4.3559322 \\
v_{(5,3)} = 19.72881356
\end{array}$$

L'espérance du plus court chemin de l'agent dans le labyrinthe stochastique en commençant à la case $(1, 1)$ est donc de $v_{(1,1)}$. Cela signifie que le nombre de cases moyen espéré, que l'agent va traverser pour atteindre les cases t_1 et t_2 en utilisant la stratégie optimale σ , est d'environ 10 cases. Cela signifie également que l'action que va effectuer l'agent lorsqu'il se situe dans la case $(1, 1)$ est

$$\begin{aligned}
\sigma((1, 1)) &= \arg \min_{\alpha \in A((1, 1))} w(\alpha) + \sum_{s' \in S \setminus T} \Delta((1, 1), \alpha, s') \cdot v_{s'} \\
&= \arg \min_{\downarrow, \rightarrow} \left(w(\rightarrow) + \frac{4}{5} \cdot 10.73 + \frac{1}{5} \cdot 8.356, w(\downarrow) + \frac{1}{5} \cdot 10.73 + \frac{4}{5} \cdot 8.356 \right) \\
&= \arg \min_{\downarrow, \rightarrow} (w(\rightarrow) + 10.2552, w(\downarrow) + 8.8305) \quad \text{avec } w(\rightarrow) = w(\downarrow) = 1 \\
&= \downarrow
\end{aligned}$$

3.4.2 Forcer des chemins de taille faible sous une haute probabilité

Dans le problème précédent, on a défini une stratégie qui minimise le coût moyen attendu pour atteindre un sous-ensemble d'états cibles. Dans cette section on va traiter le problème *des plus courts chemins stochastiques de taille limitée*. On est donc intéressé par définir une stratégie qui va maximiser la probabilité d'atteindre les états cibles avec un faible coût.

Définition 3.14 (Les plus courts chemins stochastiques de taille limitée). Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $l \in \mathbb{N}$, la longueur maximale des chemins que l'on va traiter, $s \in S$, un état de \mathcal{M} et $b \in [0, 1] \cap \mathbb{Q}$, le seuil de probabilité. Le problème consiste à décider si il existe une stratégie σ pour \mathcal{M} telle que

$$\mathbb{P}_s^\sigma(\{\pi \in Paths(s) \mid TS^T(\pi) \leq l\}) \geq b$$

i.e., une stratégie pour laquelle la probabilité de l'accessibilité limitée par le coût l dans la CMP induite par cette stratégie (cf. section 2.11) est supérieure à b .

Réduction au problème d'accessibilité

Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $T \subseteq S$, un sous-ensemble d'états cibles, $l \in \mathbb{N}$, la longueur maximale des chemins et $b \in [0, 1] \cap \mathbb{Q}$, le seuil de probabilité. De façon similaire au problème d'accessibilité limité par un coût dans une CMP, on résout le problème des plus courts chemins stochastiques de taille limitée par le coût l , en utilisant la stratégie qui résout le problème d'accessibilité sur la PDM $\mathcal{M}' = (S', A', \Delta')$ pour le sous-ensemble $T' \subseteq S'$, que l'on construit comme suit :

- S' est un ensemble de tuples (s, v) où $s \in S$ et $v \in \{0, 1, \dots, l\} \cup \{\perp\}$. On considère que $\perp > l$, avec $\perp + v = \perp \quad \forall v \in \{0, 1, \dots, l\}$. Intuitivement, v enregistre le coût du chemin en parcourant \mathcal{M} . Les états cibles sont donc les états de $T' = \{(s, v) \mid s \in T \wedge v \leq l\}$.
- Pour toutes actions $\alpha \in A$, $\alpha \in A'$ et $\forall (s, v) \in S'$, $A'((s, v)) = A(s)$.
- La fonction de transition Δ' est définie comme suit :

Pour toutes paires $(s, v), (s', v') \in S'$ et $\forall \alpha \in A$,

$$\Delta'((s, v), \alpha, (s', v')) = \begin{cases} \Delta(s, \alpha, s') & \text{si } v' = v + w(\alpha) \leq l \text{ ou} \\ & \text{si } v' = \perp \text{ et } v + w(\alpha) > l \\ 0 & \text{sinon} \end{cases}$$

Il est évident que résoudre le problème d'accessibilité dans \mathcal{M}' pour T' et ainsi définir une stratégie pour laquelle la probabilité d'atteindre T' est maximale revient à maximiser la probabilité d'atteindre les états de T avec un coût $\leq l$. Dès lors, soit σ , la stratégie sans mémoire qui résout le problème d'accessibilité du PDM \mathcal{M}' (cf. lemme 3.1), on a

$$\mathbb{P}_s^\sigma(\{\pi \in Paths(s) \mid TS^T(\pi) \leq l\}) = \mathbb{P}_{(s,0)}^\sigma(\Diamond T')$$

Remarque 3.5. La stratégie sans mémoire σ résout le problème d'accessibilité du PDM \mathcal{M}' et résout le problème des plus courts chemins stochastiques de taille limitée par l sur \mathcal{M}' . On peut définir une stratégie à mémoire finie $\sigma^* = (Q, \sigma_\alpha, \delta, \delta_0)$ équivalente pour le PDM \mathcal{M} , où

- il existe une bijection entre l'ensemble des modes Q de σ^* et l'ensemble des éléments de $V = \{0, 1, \dots, l\} \cup \{\perp\}$. Par abus de langage, on dit que $Q = V$. Dès lors, $\forall v \in V$ et $\forall s \in S$,
- $\sigma_\alpha(v, s) = \sigma((s, v))$,
- $\delta(v, s) = \begin{cases} v + w(\sigma(s, v)) & \text{si } v + w(\sigma(s, v)) \leq l \\ \perp & \text{sinon} \end{cases}$ et
- $\delta_0(s) = 0$.

La CM induite par la stratégie σ^* sur \mathcal{M} est égale à celle induite par la stratégie σ sur \mathcal{M}' .

Théorème 3.4. Le problème des plus courts chemins stochastiques de taille limitée peut être résolu en temps pseudo-polynomial, en fonction de la taille de l'encodage de l .

Exemple 3.11 (Résolution du problème des plus courts chemins stochastiques de taille limitée sur un PDMP simple). Soit $\mathcal{M}_{\text{simple}} = (S, A, \Delta, w)$, un PDMP (cf. figure 3.13). On souhaite savoir si il existe une stratégie σ qui permet d'obtenir une probabilité d'atteindre t avec un σ -chemin de taille ≤ 8 supérieure au seuil de probabilité $\frac{3}{4}$. Pour ce faire, on construit d'abord $\mathcal{M}'_{\text{simple}}$, pour ensuite réduire le problème au problème d'accessibilité du PDMP $\mathcal{M}'_{\text{simple}}$, et cela pour finalement créer la stratégie σ , qui résout ce problème d'accessibilité.

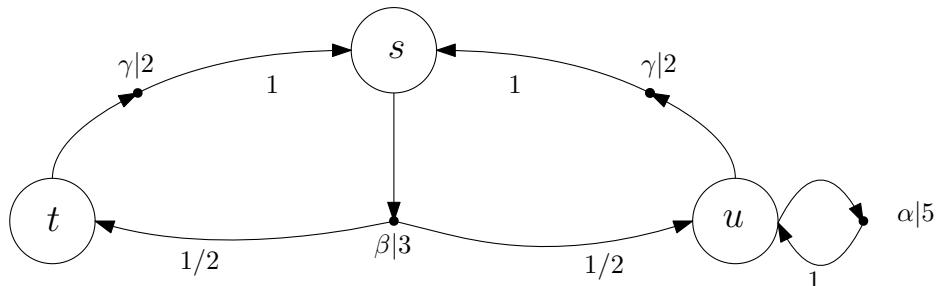


FIGURE 3.13 – Simple PDMP $\mathcal{M}_{\text{simple}}$

Soit σ , la stratégie qui résout le problème d'accessibilité de $\mathcal{M}'_{\text{simple}}$. On souhaite connaitre si la probabilité d'atteindre t depuis s avec un chemin de longueur inférieure à 8 est supérieure à $\frac{3}{4}$, i.e.,

$$\mathbb{P}_s^\sigma(\{\pi \in \text{Paths}(s) \mid TS^T(\pi) \leq 8\}) \geq 0.75$$

Cela revient à déterminer si $\mathbb{P}_{(s,0)}^\sigma(\Diamond T') \geq 0.75$, i.e., si $\mathbb{P}_{(s,0)}^{\max}(\Diamond T') \geq 0.75$.

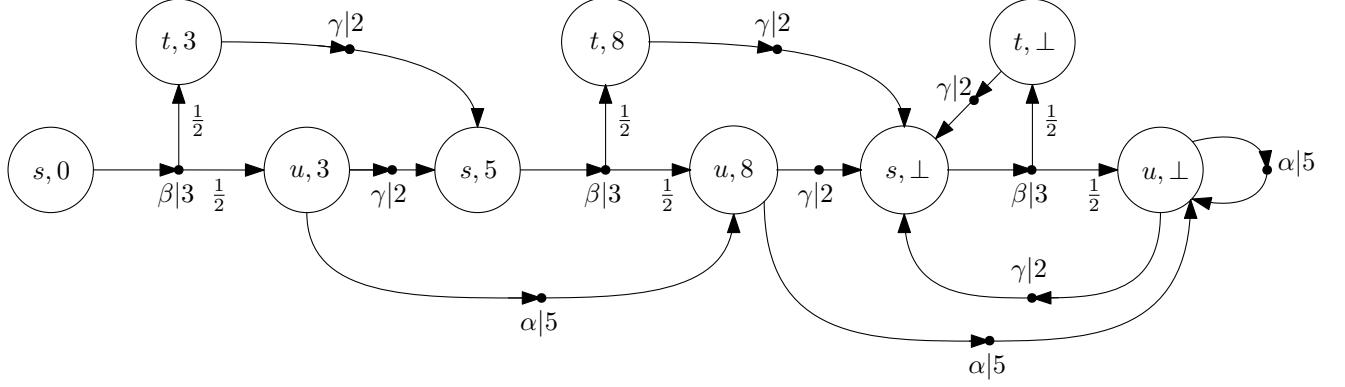


FIGURE 3.14 – Partie du PDMP $\mathcal{M}'_{\text{simple}}$ qui concerne le "dépliage" de l'état s de $\mathcal{M}_{\text{simple}}$. Par souci de lisibilité, les étiquettes telles que $\Delta(s, \alpha, s') = 1$ sont omises sur cette figure.

Pour calculer cette probabilité, on résout le PL défini au théorème 3.2 :

Tout d'abord, on a que $x_{(u,8)} = x_{(s,⊥)} = x_{(t,⊥)} = x_{(u,⊥)} = 0$ car ces sommets ne sont pas connexes à T dans le graphe sous-jacent de $\mathcal{M}'_{\text{simple}}$. On a également que $T = \{(t, 3), (t, 8)\}$, donc $x_{(t,3)} = x_{(t,8)} = 1$. Donc, le PL est défini comme suit :

$$\min_{s \in S} x_{(s,0)} + x_{(s,5)} + x_{(u,3)} + 2$$

sous les contraintes

$$x_{(s,0)} - \frac{1}{2}x_{(u,3)} \geq \frac{1}{2}$$

$$x_{(s,5)} + x_{(u,3)} \geq 0$$

$$x_{(s,5)} \geq \frac{1}{2}$$

$$1 \geq x_{(s,0)}, x_{(u,3)}, x_{(s,5)} \geq 0$$

On peut résoudre ce problème via la méthode du simplexe. La solution optimale $(v_s)_{s \in S}$ de ce PL est

$$\begin{aligned} v_{(s,0)} &= \frac{3}{4} & v_{(t,3)} &= 1 & v_{(u,3)} &= \frac{1}{2} \\ v_{(s,5)} &= \frac{1}{2} & v_{(t,8)} &= 1 & v_{(u,8)} &= 0 \\ v_{(s,⊥)} &= 0 & v_{(t,⊥)} &= 0 & v_{(u,⊥)} &= 0 \end{aligned}$$

Dès lors, on bien a que $v_{(s,0)} = \mathbb{P}_{(s,0)}^{\max}(\Diamond T') \geq \frac{3}{4}$. La stratégie σ résout donc le problème des plus courts chemins stochastiques de taille limitée par 8.

Chapitre 4

Implémentation

Ce chapitre concerne l'implémentation des solutions des problèmes de plus court chemin stochastique rencontrés dans le chapitre précédent pour les processus décisionnels de Markov. Ce chapitre traitera des structures de données optimisées pour représenter les processus décisionnels de Markov et pour résoudre les problèmes de plus court chemin stochastique. Il traitera également des algorithmes utilisés pour résoudre ces problèmes. L'implémentation du programme est réalisée dans le langage Python (`python 3`).

4.1 Représentation des Processus Décisionnels de Markov

En pratique, les différentes structures de données abordées dans la suite de ce document sont implémentées en Python. Ces implementations sont les suivantes :

	Python
tableau	list
table de hachage	set
hash map	dict
file	deque

La première étape requise pour implémenter les différents problèmes abordés au chapitre précédent est de définir une structure de données capable de représenter un PDMP. Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDMP fini. Afin d'aborder les algorithmes de la suite de ce document, la structure de données doit comporter les caractéristiques suivantes :

- Il doit être possible d'obtenir le poids de chaque action $\alpha \in A$, i.e., $w(\alpha)$.
- Pour chaque état $s \in S$, il doit être possible d'itérer sur les actions possibles pour s , i.e., sur les actions $\alpha \in A(s)$, et pour chaque action $\alpha \in A(s)$, il doit être possible d'itérer sur chaque successeur- α s' , associé à la valeur $\Delta(s, \alpha, s')$. Plus formellement, pour tout $s \in S$ et pour tout $\alpha \in A(s)$, il doit être possible d'itérer sur les éléments de l'ensemble

$$\text{SuccPr}(s, \alpha) = \{(s', pr) \in S \times [0, 1] \mid \Delta(s, \alpha, s') > 0 \wedge pr = \Delta(s, \alpha, s')\}$$

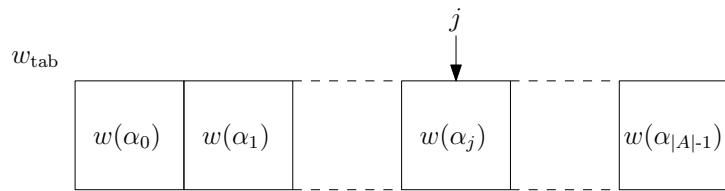
- Pour tout $s' \in S$, il doit être possible d'itérer sur les éléments de l'ensemble

$$Pred(s') = \{(s, \alpha) \in S \times A \mid \Delta(s, \alpha, s') > 0\}$$

- Pour tout $s \in S$, il doit être possible d'itérer sur les prédécesseurs de s dans le graphe sous-jacent $G^{\mathcal{M}}$, i.e., sur les éléments de l'ensemble

$$pred(s) = \{s^* \in S \mid s \in succ(s^*)\}$$

Comme \mathcal{M} est fini, il est possible d'énumérer les états ainsi que les actions du PDMP. En effet, $\forall i \in \{0, \dots, |S| - 1\}$ et $\forall j \in \{0, \dots, |A| - 1\}$, $s_i \in S$ est le $i^{\text{ème}}$ état de S et $\alpha_j \in A$ est la $j^{\text{ème}}$ action de A . En se servant de l'énumération des actions, il est simple d'associer à chaque action $\alpha_j \in A$ son poids, et cela à l'aide d'un tableau de taille $|A|$. On définit donc w_{tab} , un tableau de taille $|A|$, tel que pour tout $j \in \{0, \dots, |A| - 1\}$, $w_{\text{tab}}[j] = w(\alpha_j)$. De cette façon, récupérer le poids d'une action a une complexité en temps en $\mathcal{O}(1)$.

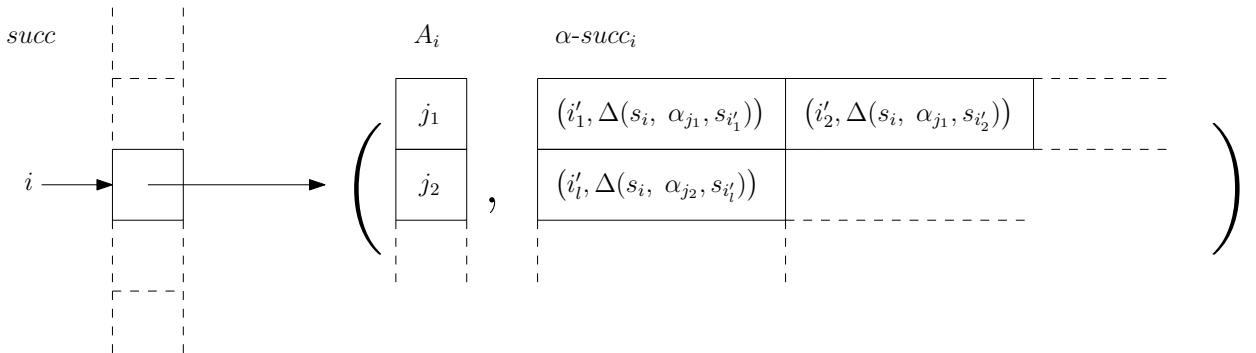


Afin d'itérer sur les actions possibles d'un état $s_i \in S$, i.e., $A(s_i)$, où $i \in \{0, \dots, |S| - 1\}$, on crée un tableau $succ$ de taille $|S|$. Pour tout $i \in \{0, \dots, |S| - 1\}$, l'entrée i du tableau contient un tuple, contenant lui-même deux tableaux. Le premier tableau, A_i , est un tableau contenant les indices des actions de $A(s_i)$, i.e.,

$$j \in A_i \iff \alpha_j \in A(s_i),$$

avec $j \in \{0, \dots, |A| - 1\}$ et $\alpha_j \in A$, la $j^{\text{ème}}$ action de A . Le second tableau est le tableau des successeurs de s_i , $\alpha\text{-succ}_i$. Il contient un tableau pour chaque action possible de s_i . Soit $k \in \{0, \dots, |A(s_i)| - 1\}$, ce tableau respecte la propriété suivante :

$$A_i[k] = j \iff \forall (s_{i'}, pr) \in SuccPr(s_i, \alpha_j), (i', pr) \in \alpha\text{-succ}_i[k]$$



Cette façon de stocker les successeurs de $s_i \in S$ est intéressante, car cela permet d'obtenir en $\mathcal{O}(1)$ les actions possibles de s_i (en récupérant A_i).

Il reste à définir deux autres tableaux de taille $|S|$ qui vont permettre de stocker les prédecesseurs de chaque état, pour de cette façon pouvoir accéder à leurs valeurs en $\mathcal{O}(1)$. Il s'agit des tableaux $pred$ et $\alpha\text{-}pred$ qui représentent respectivement les prédecesseurs de tout $s_i \in S$ dans le graphe sous-jacent de \mathcal{M} et les tuples $(s, \alpha) \in Pred(s_i)$, et cela pour tout $i \in \{0, \dots, |S| - 1\}$. Soient $k \in \{0, \dots, |A(s_i)| - 1\}$ et $j \in \{0, \dots, |A| - 1\}$ tels que $A_i[k] = j$, les deux tableaux respectent la propriété suivante :

$$\forall (i', pr) \in \alpha\text{-}succ_i[k], i \in pred[i'] \wedge (i, k) \in \alpha\text{-}pred[i']$$

De cette façon, pour tout $i' \in \{0, \dots, |S| - 1\}$,

- $pred[i']$ est une table de hachage contenant tous les indices des prédecesseurs de $s_{i'}$ dans le graphe sous-jacent de \mathcal{M} .
- $\alpha\text{-}pred[i']$ est un tableau de tuples contenant tous les indices des états et des actions possibles de ces états, qui eux-même sont éléments de l'ensemble $Pred(s_{i'})$.

L'intérêt d'utiliser une table de hachage pour stocker les éléments de $pred[i']$ est que, contrairement aux éléments de $\alpha\text{-}pred[i']$, pour tout $i \in \{0, \dots, |S| - 1\}$, lorsqu'il existe une action $\alpha \in A(s_i)$ telle qu'un successeur- α de s_i est ajouté au tableau $\alpha\text{-}succ_i$, i peut déjà avoir été ajouté auparavant dans $pred[i']$ si il existe une autre action $\beta \in A(s_i)$ telle que $\Delta(s_i, \beta, s_{i'}) > 0$, i.e., si $s_{i'}$ est un successeur- β de s_i . La vérification que cet élément se trouve déjà dans $pred[i']$, afin de ne pas l'y ajouter si il s'y trouve déjà, se fait donc en moyenne en $\mathcal{O}(1)$.

Dans la suite de ce document, lorsque des algorithmes vont traiter de manipulation de PDMP $\mathcal{M} = (S, A, \Delta, w)$, on va considérer la représentation de \mathcal{M} par la structure de données décrite ci-dessus, i.e., soient $i \in \{0, \dots, |S| - 1\}$, $j \in \{0, \dots, |A| - 1\}$ et $k \in \{0, \dots, |A(s_i)| - 1\}$,

- $S := \{0, \dots, |S| - 1\}$
- $A := \{0, \dots, |A| - 1\}$
- $w(\alpha_j) := w_{\text{tab}}[j]$
- $Act(s_i) := A_i$
- $SuccPr(s_i, \alpha_j) := \alpha\text{-}succ_i[k]$ où $A_i[k] = j$ (i.e., $\alpha_j \in A(s_i)$)
- $pred(s_i) := pred[i]$
- $Pred(s_i) := \alpha\text{-}pred[i]$

4.2 Solveurs

Dans cette section, nous allons aborder la façon dont les problèmes rencontrés au chapitre précédent ont été implémentés.

4.2.1 Problème d'accessibilité

Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP et $T \subseteq S$, un sous-ensemble d'états cibles (cf. section 3.3). On souhaite définir une stratégie qui résout le problème d'accessibilité à T pour \mathcal{M} . Pour ce faire, on va construire un vecteur $(x_s)_{s \in S}$ tel que $\forall s \in S, x_s = \mathbb{P}_s^{\max}(\Diamond T)$. Résoudre le problème d'accessibilité à T pour le PDM \mathcal{M} se déroule en plusieurs étapes :

1. Déterminer les états s qui ne sont pas connexes à T dans $G^{\mathcal{M}}$ pour pouvoir fixer la valeur de x_s à 0.
2. Déterminer les états s pour lesquels $\mathbb{P}_s^{\max}(\Diamond T) = 1$ afin de fixer la valeur de x_s à 1.
3. Générer un PL comme décrit au théorème 3.2 sans générer de contraintes supplémentaires pour les états s dont x_s a déjà été fixé.
4. Déterminer les actions de A^{\max} et générer un nouveau PDM \mathcal{M}^{\max} qui ne considère que les actions de cette ensemble pour pouvoir ensuite construire une stratégie sans mémoire optimale qui résout le problème (cf. preuve du lemme 3.1).

Aborder la construction du vecteur $(x_s)_{s \in S}$ de cette façon permet de limiter les calculs effectués par le solveur de PL en limitant le nombre de contraintes générées.

Déterminer les états non-connexes à T

Afin de déterminer les états qui ne sont pas connexes à T , on va réaliser un parcours arrière du graphe sous-jacent $G^{\mathcal{M}}$ depuis le sous-ensemble d'états cibles T et marquer les états rencontrés lors de ce parcours. Lorsque le parcours est terminé, les états qui ne sont pas marqués ne sont pas connexes à T .

Algorithme 1 Parcours en largeur arrière

Entrées :

- $G = (S, E)$, un graphe.
- $T \subseteq S$, un sous-ensemble de sommets.

Sortie :

- $marque$, un tableau de taille $|S|$ tel que $\forall s \in S$, $marque[s] = Vrai$ ssi s est connexe à T dans G .

```
1: marque  $\leftarrow$  initialiser un tableau de taille  $|S|$  dont tous les éléments sont Faux
2: prédécesseurs  $\leftarrow$  initialiser file
3: pour tout  $t \in T$  faire
4:   marque[ $t$ ]  $\leftarrow Vrai$ 
5:   pour tout  $s' \in pred(t)$  faire
6:     enfiler(prédécesseurs,  $s'$ )
7:   tant que prédécesseurs non vide faire
8:      $s \leftarrow$  défiler(prédécesseurs)
9:     si non marque[ $s$ ] alors
10:      marque[ $s$ ]  $\leftarrow Vrai$ 
11:      pour tout  $s' \in pred(s)$  faire
12:        enfiler(prédécesseurs,  $s'$ )
13: retourner marque
```

L'algorithme parcourt les sommets du graphe sous-jacent $G^{\mathcal{M}} = (S, E)$ depuis le sous-ensemble T . En pratique, le parcours des sommets du graphe sous-jacent est réalisé en itérant sur l'ensemble de prédécesseurs *pred* défini dans la structure de données du PDM (cf. section 4.1). Dans le pire des cas, il existe un chemin de s à T pour tout $s \in S$, et tous les sommets sont visités et marqués à la fin de l'algorithme. Si les sommets ont déjà été marqués lors de l'exécution, alors ils ne peuvent plus être marqués par la suite. Chaque sommet n'est donc visité dans le pire des cas qu'une unique fois par l'algorithme. Lors de l'exécution, pour chaque sommet non-marqué que l'algorithme visite, celui-ci vérifie si ses prédecesseurs sont déjà marqués, i.e., l'algorithme vérifie chaque arc entrant du sommet considéré. Dans le pire des cas, l'algorithme a donc vérifié une unique fois tous les arcs du graphe à la fin de l'exécution. Le temps d'exécution de l'algorithme est donc en $\mathcal{O}(|S| + |E|)$.

Déterminer les états s pour lesquels $\mathbb{P}_s^{\max}(\Diamond T) = 1$

L'algorithme suivant provient du livre Principle of Model checking [1]. En pratique, bien que le PL se résout en temps polynomial, il est nettement plus efficace de prétraiter les états du PDM afin de déterminer les états s pour lesquels $\mathbb{P}_s^{\max}(\Diamond T) = 1$, afin d'ensuite générer les contraintes du PL pour les états s pour lesquels x_s n'a pas encore été fixé.

Algorithme 2 Trouver les états s tels que $\mathbb{P}_s^{\max}(\Diamond T) = 1$

Entrées :

- $\mathcal{M} = (S, A, \Delta)$, un PDM fini.
- $T \subseteq S$, un sous-ensemble d'états cibles.

Sortie :

- Les états s du PDM tels que $\mathbb{P}_s^{\max}(\Diamond T) = 1$.

1: $U \leftarrow \{s \in S \mid s \text{ est non-connexe à } T \text{ dans } G^{\mathcal{M}}\}$

2: **tant que** $U \neq \emptyset$ **faire**

3: $R \leftarrow U$

4: **tant que** $R \neq \emptyset$ **faire**

5: Soit $u \in R$

6: $R \leftarrow R \setminus \{u\}$

7: **pour tout** $(s, \alpha) \in \text{Pred}(u)$ tel que $s \notin U \cup T$ **faire**

8: supprimer α de $A(s)$

9: **si** $A(s) = \emptyset$ **alors**

10: $R \leftarrow R \cup \{s\}$

11: $U \leftarrow U \cup \{s\}$

{tous les arcs entrants de u ont maintenant été supprimés}

12: supprimer u et ses arcs sortants dans \mathcal{M}

{il faut maintenant recalculer les états qui ne peuvent pas atteindre T dans le PDM modifié}

13: $U \leftarrow \{s \in S \setminus U \mid s \text{ n'est pas connexe à } T \text{ dans } G^{\mathcal{M}}\}$

14: **retourner** les états restants du MDP

L'algorithme abordé ci-dessus est exact et est polynomial en $|\mathcal{M}|$. Cependant, le pseudo-code de cet algorithme est de très haut niveau. On va maintenant l'adapter à la structure de données du PDM que l'on a défini plus tôt dans ce document.

Algorithme 3 PrMax1

Entrées :

- $\mathcal{M} = (S, A, \Delta)$, un PDM fini.
- $T \subseteq S$, un sous-ensemble d'états cibles.

Sortie :

- $S_{=1}^{\max}$, une liste contenant les sommets du PDM tel que $\forall s \in S_{=1}^{\max}, \mathbb{P}_s^{\max}(\Diamond T) = 1$.

- 1: $connecté \leftarrow ParcoursLargeurArrière(G^{\mathcal{M}}, T)$
- 2: $supprimé \leftarrow$ initialiser tableau de taille $|S|$ dont les éléments sont *Faux*
- 3: $actionSupprimée \leftarrow$ initialiser tableau de taille $|S|$
- 4: **pour tout** $s \in S$ **faire**
- 5: $actionSupprimée[s] \leftarrow$ initialiser tableau de taille $|A(s)|$ dont les éléments sont *Faux*
- 6: $nombreActionsSupprimées \leftarrow$ initialiser tableau de taille $|S|$ dont les éléments sont 0
- 7: **tant que** $compter(\lambda s : \text{non } connecté[s] \text{ et non } supprimé[s], S) > 0$ **faire**
- 8: $R \leftarrow$ initialiser file
- 9: **pour tout** $s \in S$ **faire**
- 10: **si** non $connecté[s]$ et non $supprimé[s]$ **alors**
- 11: $enfiler(R, s)$
- 12: **tant que** R non vide **faire**
- 13: $u \leftarrow défiler(R)$
- 14: **pour tout** $(s, \alpha) \in Pred(u)$ **faire**
- 15: **si** $connecté[s]$ et non $actionSupprimée[s][\alpha]$ et $s \notin T$ **alors**
- 16: $actionSupprimée[s][\alpha] \leftarrow Vrai$
- 17: $nombreActionsSupprimées[s] \leftarrow nombreActionsSupprimées[s] + 1$
- 18: **si** $nombreActionsSupprimées[s] = |A(s)|$ **alors**
- 19: $enfiler(R, s)$
- 20: $connecté[s] \leftarrow Faux$
- 21: $supprimé[u] \leftarrow Vrai$
- 22: $G = (S, E) \leftarrow$ initialiser Graphe avec E vide
- 23: **pour tout** $s \in S$ **faire**
- 24: **si** non $supprimé[s]$ **alors**
- 25: **pour tout** $\alpha \in A(s)$ **faire**
- 26: **si** non $actionSupprimée[\alpha]$ **alors**
- 27: **pour tout** $(s', pr) \in SuccPr(s, \alpha)$ **faire**
- 28: **si** non $supprimé[s']$ **alors**
- 29: $ajouterArc(E, (s, s'))$
- 30: $connecté \leftarrow ParcoursLargeurArrière(G, T)$
- 31: $S_{=1}^{\max} \leftarrow$ initialiser liste
- 32: **pour tout** $s \in S$ **faire**
- 33: **si** non $supprimée[s]$ **alors**
- 34: $ajouter(S_{=1}^{\max}, s)$
- 35: **retourner** $S_{=1}^{\max}$

Notation : $\lambda x : v$ est une notation qui se base sur le lambda-calcul (cf. ligne 7). En effet, cela représente une fonction qui associe x à v , où v contient en général des occurrences de x (en lambda-calcul : $\lambda x.v$).

Commentaires :

- Tous les tableaux de booléens peuvent être remplacés par des tables de hachage. En effet,
 - Définir la valeur d'un état à *Vrai* dans le tableau est équivalent à ajouter un état dans la table de hachage.
 - Vérifier que la valeur d'un état est *Vrai* dans le tableau équivaut à vérifier qu'un état se trouve dans la table de hachage.
- Les différences résident en la complexité en mémoire et en temps des 2 structures de données.
- L'espace occupé par les tableaux en mémoire est fixe, tandis que celui occupé par les tables de hachage est alloué dynamiquement en fonction du nombre d'éléments présents à l'intérieur. Les tables de hachage occupent dans la plupart des cas moins d'espace en mémoire que les tableaux, surtout dans le cas où la plupart des états sont connexes à T et lorsque peu d'actions mènent le système vers un état non-connexe à T . Cependant, par le fait que la mémoire est allouée dynamiquement, lorsqu'un nombre conséquent d'états sont liés à des états non-connexes à T , la mémoire allouée aux tables de hachage se rapproche fortement de celle des tableaux (et peut même dans le pire des cas la dépasser). Ce phénomène n'est pas rare dans le cas de la résolution du problème des plus courts chemins stochastiques de taille limitée.
 - La complexité en temps de la vérification qu'un état a pour valeur *Vrai* dans un tableau est toujours en $\mathcal{O}(1)$, tandis que la vérification que l'état se trouve dans la table de hachage est en moyenne en $\mathcal{O}(1)$, mais peut aller jusqu'à $\mathcal{O}(|S|)$, voir $\mathcal{O}(|S| \times |A|)$ dans le pire des cas pour certaines tables en Python (cf. TimeComplexity dans la documentation officielle des *set* de Python).

La performance en terme de complexité (en mémoire et en temps) de ces deux approches dépendent donc entièrement du PDM traité. On fait ici le choix d'assurer une complexité en temps optimale au détriment de la complexité en mémoire.

- On suppose que la fonction *compter* (cf. ligne 7) compte les éléments d'une structure de données qui respectent la propriété entrée en paramètre de la fonction. Dans notre cas, la fonction retourne le nombre d'états qui ne sont pas connectés à T et qui n'ont pas encore été supprimés.
- On suppose que l'action d'ajouter un arc (s, s') au graphe (cf. ligne 29) met à jour la table de hachage représentant les prédécesseurs de s' dans ce graphe, i.e., $pred(s') \leftarrow \{s\} \cup pred(s')$.

Générer le PL

Lors des 2 étapes précédentes, on a fixé la valeur de x_s à 0 pour les états s non-connexes à T dans le graphe sous jacent $G^{\mathcal{M}}$ et on a fixé les valeurs de x_s à 1 pour les états s tels que $\mathbb{P}_s^{\max}(\Diamond T) = 1$. Il reste à générer les contraintes du PL pour le reste des états (cf. théorème 3.2). Afin de générer un PL, on utilise le package PuLP de python 3. Ce package permet de modéliser des programmes linéaires de façon très intuitive en python (cf. figure 4.1) et de les résoudre à partir de solveurs externes au choix (GLPK, CPLEX, COIN CLP/CBC, GURUOBI, etc.). Par défaut, le solveur utilisé sera GLPK, mais le programme permet d'indiquer quel solveur utiliser.

```
for s in states:
    x[s] = pulp.LpVariable(s, lowBound=0, upBound=1)
for s in states:
    if s in pr_max1:
        x[s] = 1
    elif not connected[s]:
        x[s] = 0
    else:
        for (alpha, successors_list) in mdp.alpha_successors(s):
            linear_program += x[s] >= sum(pr * x[succ] for (succ, pr) in successors_list)
```

FIGURE 4.1 – Génération des contraintes du PL avec PuLP.

Construire la stratégie

Une fois que le vecteur $(v_s)_{s \in S}$ représentant la solution du PL est calculé, on peut construire la stratégie qui va résoudre le problème d'accessibilité du PDM \mathcal{M} . L'algorithme suivant se base sur la preuve du lemme 3.1 dans laquelle on construit la stratégie optimale du problème d'accessibilité.

Algorithme 4 Construire la stratégie optimale pour le problème d'accessibilité

Entrées :

- $\mathcal{M} = (S, A, \Delta)$, un PDM.
- v , un tableau pour lequel $v[s] = \mathbb{P}_s^{\max}(\Diamond T) \forall s \in S$.

Sortie :

- la stratégie optimale qui résout le problème d'accessibilité de \mathcal{M} .
- 1: $A^{\max} \leftarrow$ initialiser ensemble d'actions de PDM
 - 2: **pour tout** $s \in S$ **faire**
 - 3: **pour tout** $\alpha \in A(s)$ **faire**
 - 4: $pr \leftarrow \sum_{(s', pr) \in \text{SuccPr}(s, \alpha)} pr \cdot v[s']$
 - 5: **si** $pr = v[s]$ **alors**
 - 6: ajouter($A^{\max}(s)$, α)
 - 7: $\mathcal{M}^{\max} = (S, A^{\max}) \leftarrow$ initialiser PDM
 - 8: $\text{étapes} \leftarrow \text{PlusCourtChemin}(G^{\mathcal{M}^{\max}}, T)$
 - 9: $A^* \leftarrow$ initialiser tableau de taille $|S|$
 - 10: **pour tout** $s \in S$ **faire**
 - 11: $A^*[s] \leftarrow \alpha \in A^{\max}(s)$ {initialiser $A^*[s]$ avec une action arbitraire}
 - 12: **pour tout** $\alpha \in A^{\max}(s)$ **faire**
 - 13: **pour tout** $(s', pr) \in \text{SuccPr}(s, \alpha)$ **faire**
 - 14: **si** $\text{étapes}[s] - 1 = \text{étapes}[s']$ **alors**
 - 15: $A^*[s] \leftarrow \alpha$
 - 16: **retourner** $\lambda s : A^*[s]$
-

Il reste maintenant à définir l'algorithme des plus courts chemins (en terme d'étapes) dans le graphe sous-jacent pour tout sommet s vers le sous-ensemble des sommets T . Pour ce faire, on adapte l'algorithme de parcours en largeur arrière pour numérotter les sommets en fonction de l'itération, i.e., l'étape à laquelle les sommets ont été visités.

Algorithme 5 Parcours en largeur arrière : plus court chemin

Entrées :

- $G = (S, E)$, un graphe.
- $T \subseteq S$, un sous-ensemble de sommets.

Sortie :

- étapes , un tableau de taille $|S|$ tel que $\forall s \in S$, $\text{étapes}[s]$ correspond au plus court chemin en terme d'étapes, i.e., du nombre d'arcs minimum parcourus pour atteindre le sous-ensemble T .

```
1:  $\text{étapes} \leftarrow$  initialiser un tableau de taille  $|S|$  dont tous les éléments sont  $\infty$ 
2:  $\text{étapeSuivante} \leftarrow$  initialiser file
3:  $i \leftarrow 0$ 
4: pour tout  $t \in T$  faire
5:    $\text{étapes}[t] \leftarrow i$ 
6:   pour tout  $s' \in \text{pred}(t)$  faire
7:     enfiler( $\text{étapeSuivante}$ ,  $s'$ )
8:   tant que  $\text{étapeSuivante}$  non vide faire
9:      $i \leftarrow i + 1$ 
10:     $\text{prédécesseurs} \leftarrow \text{étapeSuivante}$ 
11:     $\text{étapeSuivante} \leftarrow$  initialiser liste vide
12:    tant que  $\text{prédécesseurs}$  non vide faire
13:       $s \leftarrow \text{défiler}(\text{prédécesseurs})$ 
14:      si  $\text{étapes}[s] = \infty$  alors
15:         $\text{étapes}[s] = i$ 
16:        pour tout  $s' \in \text{pred}(s)$  faire
17:          enfiler( $\text{étapeSuivante}$ ,  $s'$ )
18: retourner  $\text{étapes}$ 
```

La complexité en temps de cet algorithme est identique au parcours en largeur classique ; dans le pire des cas, les états sont visités une unique fois et les arcs sont également visités une unique fois. La complexité en temps de l'algorithme est donc en $\mathcal{O}(|S| + |E|)$. On va prouver que la boucle externe de l'algorithme (cf. ligne 8) possède l'invariant suivant : $\forall s \in S$, $\text{étapes}[s]$ est la taille du chemin le plus court de s à T dans G avec un chemin de taille inférieure ou égale à i , où i est le nombre d'itérations effectuées par la boucle externe.

Cas de base : $i = 0$, i.e., lorsque l'algorithme entre dans la boucle externe. L'algorithme initialise $\text{étapes}[t] \forall t \in T$ à 0 (cf. lignes 4 et 5). On a bien que $\text{étapes}[t]$ correspond à la taille du chemin le plus court de t à T .

Cas général : Soit $i \in \mathbb{N}$. On suppose qu'on a, à la fin de la $i^{\text{ème}}$ itération de la boucle externe, que $\text{étapes}[s]$ est la taille du chemin le plus court de s à T avec un chemin de taille inférieure ou égale à i . Pendant l'étape i , tous les arcs entrants (s, s') des sommets s' tels que $\text{étapes}[s'] = i$ sont visités (cf. ligne 17). Les prédécesseurs de s' sont de cette manière ajoutés à la file étapeSuivante . Il s'agit des sommets qui vont être traités à l'étape courante $i + 1$ (cf. lignes 10 et 12). Pour chaque sommet $s \in S$ traité à l'étape

courante,

- Si $\text{étapes}[s] < i + 1$, alors on a que le sommet a déjà été visité lors des itérations précédentes de la boucle externe et on a donc par hypothèse que $\text{étapes}[s]$ est la taille du chemin le plus court depuis s vers T .
- Si $\text{étapes}[s] > i$, alors c'est que $\text{étapes}[s]$ n'a pas encore été visité par l'algorithme et que $\text{étapes}[s] = \infty$. Vu que le sommet n'a pas encore été visité, l'algorithme le marque en fixant la valeur de $\text{étapes}[s]$ à $i + 1$ (cf. lignes 14 et 15). En effet, vu que $\text{étapes}[s]$ n'avait pas été traité lors des itérations précédentes, cela signifie qu'il n'existe pas de chemin de longueur strictement inférieure à $i + 1$ de s à T dans le graphe (sinon, par hypothèse, le sommet aurait été marqué lors des itérations précédentes). De plus, vu qu'il existe un arc $(s, s') \in E$ tel que $\text{étapes}[s'] = i$ et tel que $\text{étapes}[s']$ est le plus court chemin de s' à T , on a bien qu'il existe un chemin de s à T , que ce chemin est de longueur $i + 1$ et qu'il est le plus court pour atteindre T depuis s .

Par induction sur i , on prouve l'invariant. Comme l'algorithme visite dans le pire des cas une unique fois tous les sommets et tous les arcs de G , on a forcément que l'algorithme s'arrête car la file étapeSuivante finit par être vide (cf. ligne 8). Cela mène au fait que, pour tout sommet $s \in S$, $\text{étapes}[s]$ est la longueur du chemin le plus court de s à T .

4.2.2 Problème de l'espérance du plus court chemin stochastique

Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $T \subseteq S$, un sous-ensemble d'états cibles et $l \in \mathbb{N}$, un seuil de longueur. On souhaite définir une stratégie qui résout le problème de l'espérance du plus court chemin stochastique pour le PDMP \mathcal{M} jusqu'au sous-ensemble d'états cibles T sous le seuil de longueur l (cf. sous-section 3.4.1). Tout d'abord, on génère le PL du théorème 3.3 à l'aide du package PuLP de Python. On utilise ensuite un solveur supporté par PuLP (e.g., GLPK) afin de résoudre le PL. La solution de ce PL est un tableau v de taille $|S|$ tel que $v[s] = \min_{\sigma} \mathbb{E}_s^{\sigma}(TS^T)$ pour tout $s \in S$. On construit la stratégie σ comme décrit dans le théorème 3.3. Soit $s \in S$. Dès lors, $v[s] = \mathbb{E}_s^{\sigma}(TS^T)$. Si $v[s] \leq l$, alors la stratégie σ résout le problème d'espérance du plus court chemin stochastique sous le seuil de longueur l de l'état s aux états cibles de T dans \mathcal{M} .

Algorithme 6 Résoudre le problème de l'espérance du plus court chemin stochastique

Entrées :

- $\mathcal{M} = (S, A, \Delta, w)$, un PDMP.
- $s \in S$, un état de \mathcal{M} .
- $T \subseteq S$, un sous-ensemble d'états de \mathcal{M} .
- $l \in \mathbb{N}$, un seuil de longueur.

Sortie :

- *Vrai* si il existe une stratégie σ pour laquelle $\mathbb{E}_s^\sigma(TS^T) \leq l$, *Faux* sinon.

1: $(v, \sigma) \leftarrow \text{ConstruireStratégieOptimale}(\mathcal{M}, T)$

2: **si** $v[s] \leq l$ **alors**

3: **retourner** *Vrai*

4: **sinon**

5: **retourner** *Faux*

Algorithme 7 Construire la stratégie optimale pour le problème de l'espérance du plus court chemin stochastique

Entrées :

- $\mathcal{M} = (S, A, \Delta, w)$, un PDMP.
- $T \subseteq S$, un sous-ensemble d'états cibles.

Sortie :

- v , un tableau de taille $|S|$ tel que $\forall s \in S, v[s] = \min_\sigma \mathbb{E}_s^\sigma(TS^T)$.
- σ , la stratégie optimale telle que $\forall s \in S, v[s] = \mathbb{E}_s^\sigma(TS^T)$.

1: $x \leftarrow \text{initialiser un tableau de taille } |S|$

2: $pl \leftarrow \text{initialiser PL}$

3: **pour tout** $s \in S$ **faire**

4: *initialiserVariable*($pl, x[s]$)

5: $p_1 \leftarrow \text{prMax1}(\mathcal{M}, T)$

6: $\text{objectif}(pl, \max, \sum_{s \in p_1} x[s])$

7: **pour tout** $s \in S$ **faire**

8: **si** $s \in T$ **alors**

9: $x[s] \leftarrow 0$

10: **sinon si** $s \notin p_1$ **alors**

11: $x[s] \leftarrow \infty$

12: **sinon**

13: **pour tout** $\alpha \in A(s)$ **faire**

14: *ajouterContrainte*($pl, x[s] \leq w(\alpha) + \sum_{(succ, pr) \in \text{SuccPr}(s, \alpha)} pr \cdot x[succ]$)

15: $v \leftarrow \text{résoudre}(pl)$

16: **retourner** $(v, \lambda s : \arg \min_{\alpha \in A(s)} (w(\alpha) + \sum_{(succ, pr) \in \text{SuccPr}(s, \alpha)} pr \cdot v[succ]))$

4.2.3 Problème des plus courts chemins stochastiques de taille limitée

Soient $\mathcal{M} = (S, A, \Delta, w)$, un PDMP, $l \in \mathbb{N}$, la longueur maximale des chemins que l'on va traiter, $s \in S$, un état de \mathcal{M} et $b \in [0, 1] \cap \mathbb{Q}$, un seuil de probabilité. On souhaite définir une stratégie qui résout le problème des plus courts chemins stochastiques limités par le seuil l depuis l'état s jusqu'au sous-ensemble d'états cibles T au dessus du seuil de probabilité b (cf. sous-section 3.4.2). Le problème se résout en 3 étapes.

1. "*Déplier*" le PDMP \mathcal{M} depuis l'état s afin de créer un nouveau PDM $\mathcal{M}' = (S', A', \Delta')$.
2. Construire un vecteur $(v_{s'})_{s' \in S'}$ tel que pour tout $s' \in S'$, $v_{s'} = \mathbb{P}_{s'}^{\max}(\Diamond T')$ (cf. sous-section 4.2.1).
3. Si $v_{(s,0)} \geq b$, alors la stratégie σ optimale qui résout le problème d'accessibilité de \mathcal{M}' résout le problème des plus courts chemins stochastiques de taille limitée par le seuil l depuis l'état s vers le sous-ensemble d'états cibles T .

On veut construire le PDM $\mathcal{M}' = (S', A', \Delta')$ en dépliant le PDMP \mathcal{M} afin de résoudre le problème d'accessibilité de $(s, 0)$ à $T' = \{(t, v) \in S' \mid t \in T \wedge v \leq l\}$. Afin d'assurer une complexité en temps et en mémoire optimale, certaines optimisations peuvent être réalisées lors de la construction de \mathcal{M}' par rapport à sa définition dans la sous-section 3.4.2.

- Premièrement, pour tout état $t' \in T'$, et pour chaque action possible $\alpha \in A(t')$, il n'est pas nécessaire de considérer les successeurs- α de t' . En effet, par le théorème 3.2, on aura forcément que $x[t'] = \mathbb{P}_{t'}^{\max}(\Diamond T) = 1$.
- Ensuite, il n'est pas nécessaire de considérer les états (s, v) de \mathcal{M}' tels que $v = \perp$. En effet, par définition de T' , tous ces états sont non-connexes à T' et donc, pour tout $s' \in \{(s, v) \in S' \mid v = \perp\} \subseteq S'$, on aura forcément que $x[s'] = \mathbb{P}_{(s,\perp)}^{\max}(\Diamond T') = 0$ par le théorème 3.2.
- Finalement, comme on veut résoudre le problème de $(s, 0)$ à T' , il n'est pas nécessaire de construire le PDM déplié complet. Par le fait que pour tout état $(s, v) \in S'$, v représente la longueur du chemin courant dans \mathcal{M}' , vu que pour tout $\alpha \in A((s, v))$, $w(\alpha) > 0$, on a pour tout successeur- α (s', v') de (s, v) que $v' > v$. Si on ne prend pas en compte les états de $S'_\perp = \{(s, v) \in S' \mid v = \perp\}$, le graphe sous-jacent $G^{\mathcal{M}'} = (S' \setminus S'_\perp, E)$ prend la forme d'un DAG (i.e., graphe orienté acyclique). La particularité des DAG est qu'il n'existe pas de circuit dans de tels graphes. Un circuit d'un graphe orienté $G = (S, E)$ est un chemin fini dont le premier et dernier sommet sont identiques, i.e., une séquence de sommets $\pi = s_0 \dots s_n \in S^+$ où pour tout $i \in \{0, \dots, n\}$, $(s_i, s_{i+1}) \in E$ et $s_0 = s_n$. En effet, dans \mathcal{M}' , soit $s' \in S' \setminus S'_\perp$, il n'existe pas de chemin fini $\pi = s_0 \alpha_1 s_1 \dots \alpha_n s_n \in \text{Paths}_{fin}(s')$ tel que $s_0 = s_n$. Il est donc uniquement nécessaire de considérer les états accessibles depuis l'état $(s, 0)$ dans \mathcal{M}' .

Algorithme 8 Construire \mathcal{M}'

Entrées :

- $\mathcal{M} = (S, A, \Delta, w)$, un PDMP.
- $T \subseteq S$, le sous-ensemble d'états cibles dans \mathcal{M} .
- $s \in S$, l'état à partir duquel \mathcal{M} va être déplié.
- $l \in \mathbb{N}$, le seuil de longueur des chemins de \mathcal{M} .

Sorties :

- \mathcal{M}' , le PDM déplié depuis l'état $(s, 0)$ de \mathcal{M} .
- $T' \subseteq S'$, le sous-ensemble d'états cibles dans \mathcal{M}'

- 1: $\mathcal{M}' = (S', A', \Delta') \leftarrow$ initialiser PDM avec S' et A' vides
 {on remplace l'ensemble S_\perp par un unique état s_\perp }
 - 2: *ajouterÉtat*(S' , s_\perp)
 {on ajoute l'action α_{loop} afin de faire en sorte que $\forall t' \in T'$, $\Delta(t', \alpha_{loop}, t') = \Delta(s_\perp, \alpha_{loop}, s_\perp) = 1$ }
 - 3: *ajouterAction*(A' , α_{loop})
 - 4: *ajouterAction*($A'(s_\perp)$, α_{loop})
 - 5: *ajouterSuccesseur*(\mathcal{M}' , s_\perp , α_{loop} , $(s_\perp, 1)$)
 - 6: $T' \leftarrow$ initialiser sous-ensemble d'états de \mathcal{M}'
 {on associe tout état $s \in S$ de \mathcal{M} et la valeur de la longueur de son chemin courant $v \in \mathbb{N}$ à un état de \mathcal{M}' à l'aide d'une matrice V , i.e., $V[s][v] = (s, v) \in S'$.}
 - 7: $V \leftarrow$ initialiser matrice de taille $|S| \times (l + 1)$ dont les éléments sont -1
 - 8: *déplier*(s , 0)
 - 9: **retourner** (\mathcal{M}' , T')
-

Commentaires :

- On suppose que la fonction *ajouterAction* (ligne 4) met à jour le tableau qui représente $A(s_\perp)$ dans \mathcal{M}' .
- On suppose que la fonction *ajouterSuccesseur* (ligne 5) met à jour les tableaux qui représentent les ensembles *SuccPr*, *Pred* et la table de hachage qui représente l'ensemble *pred* dans \mathcal{M}' . Note : ici, $(s_\perp, 1) \in \text{SuccPr}(s_\perp, \alpha_{loop})$.
- La matrice V peut être remplacée par un tableau de hash map tel que $\forall s \in S$, $V[s]$ contient une hash map qui possède comme clés les longueurs possibles des chemins courants lorsqu'on se situe dans l'état s dans \mathcal{M} . En d'autres termes, soit $s \in S$, un état de \mathcal{M} et $v \in \mathbb{N}$, une clé de la hash map $V[s]$, alors $V[s][v] = (s, v)$ où $(s, v) \in S'$ est un état de \mathcal{M}' . La différence réside en la complexité en temps et en mémoire des deux approches. En effet, $V[s][v]$ se récupère en $\mathcal{O}(1)$ dans la matrice et en moyenne en $\mathcal{O}(1)$ dans le tableau de hash map. Cependant, supposons que $\forall \alpha \in A$, $w(\alpha)$ est très grand. Dans ce cas, beaucoup de cases sont inutilisées dans la matrice car elles ne seront jamais remplies lors du dépliage de \mathcal{M} . La matrice est particulièrement efficace quand les valeurs des poids des actions sont petites car très peu de cases de la matrice sont "gaspillées" et celle-ci permet d'assurer de récupérer les états de \mathcal{M}' en $\mathcal{O}(1)$.

Algorithme 9 Déplier

Entrées :

- $s \in S$, l'état à partir duquel \mathcal{M} va être déplié.
- $v \in \mathbb{N}$, la longueur du chemin courant.

Effet de bord :

- Met à jour et construit le PDM déplié \mathcal{M}' , le sous-ensemble d'états cible T' et la matrice V .

```

1:  $s' \leftarrow V[s][v]$ 
2: si  $s' = -1$  alors
3:    $V[s][v] \leftarrow |S'|$ 
4:   ajouter  $\text{Etat}(S', s')$ 
5: si  $s \in T$  alors
6:   ajouter  $\text{Etat}(T', s')$ 
7:   ajouter  $\text{Action}(A'(s'), \alpha_{loop})$ 
8:   ajouter  $\text{Successeur}(\mathcal{M}', s', \alpha_{loop}, (s', 1))$ 
9: sinon
10:  pour tout  $\alpha \in A(s)$  faire
11:    si  $v + w(\alpha) > l$  alors
12:      ajouter  $\text{Successeur}(\mathcal{M}', s', \alpha, (s_\perp, 1))$ 
13:    sinon
14:      pour tout  $(succ, pr) \in SuccPr(s, \alpha)$  faire
15:         $v' \leftarrow v + w(\alpha)$ 
16:        déplier( $succ, v'$ )
17:        ajouter  $\text{Action}(A'(s'), \alpha)$ 
18:         $succ' \leftarrow V[succ][v']$ 
19:        ajouter  $\text{Successeur}(\mathcal{M}', s', \alpha, (succ', pr))$ 

```

La complexité en temps du dépliage est intuitive grâce à la matrice V . Pour rappel, la matrice V est de taille $|S|(l+1)$ et chaque case remplie correspond à un sommet de \mathcal{M}' . À chaque appel récursif, on remplit une case de la matrice uniquement si celle-ci n'a pas encore été remplie. Le pire des cas correspond donc à celui où toutes les cases de la matrice sont remplies (i.e., celui où le graphe sous-jacent du PDMP est complet et où toutes les actions α ont un poids de $w(\alpha) = 1$). À chaque fois qu'une case est remplie, l'algorithme itère sur les actions α de chaque état, et ensuite sur ses successeurs- α . On suppose qu'ajouter un état, ajouter une action et ajouter un successeur se fait en $\mathcal{O}(1)$. De ce fait, traiter une case de V se fait en $\mathcal{O}(|A||S|)$ (le pire des cas est celui où $\forall s \in S, |A(s)| = |A|$, i.e., toutes les actions sont possibles pour chaque s et celui où $\forall \alpha \in A(s), |Succ(s, \alpha)| = |S|$). De ce fait, on a que le dépliage du PDMP se fait en $\mathcal{O}(|V||S||A|) = \mathcal{O}(l|S|^2|A|)$. Comme la complexité du dépliage dépend de l , celle-ci est pseudo-polynomiale. Ensuite, on s'intéresse à la taille du PDM déplié, i.e., $|\mathcal{M}'|$. La taille d'un PDM dépend du nombre d'arcs de son graphe sous-jacent (ici, $G^{\mathcal{M}'} = (S', E')$) ainsi que des actions possibles pour chaque état. Le nombre de sommets du PDM déplié correspond au nombre de cases remplies de V et le pire des

cas correspond à celui où toutes les cases de V sont remplies et donc celui où le graphe sous-jacent de \mathcal{M} est complet et où le poids de toutes les actions est de 1. Comme le graphe sous-jacent de \mathcal{M}' prend la forme d'un DAG (cf. figure 4.2), la taille du PDM déplié est en $\mathcal{O}(|E'| |A'|) = \mathcal{O}(l |S|^2 |A|)$, ce qui correspond à la complexité en temps de la construction de \mathcal{M}' .

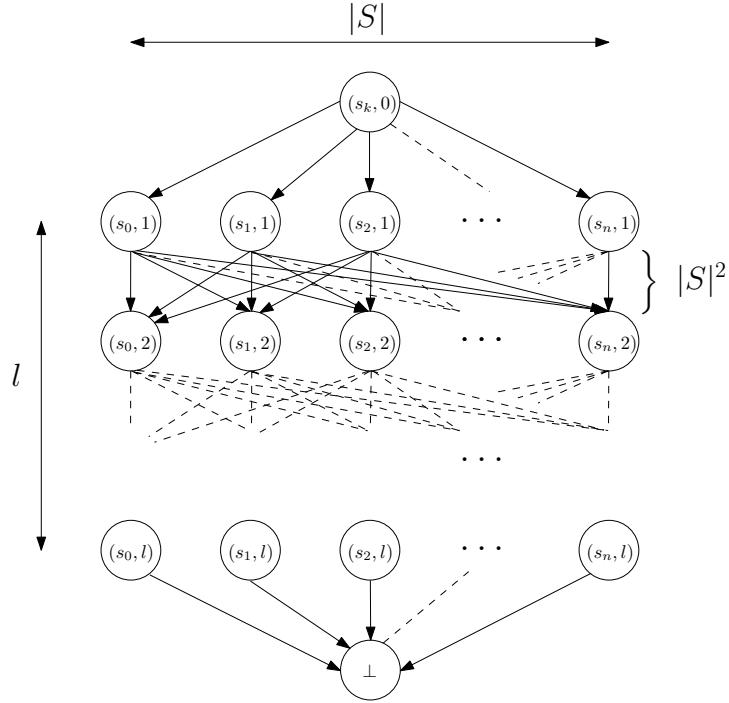


FIGURE 4.2 – Forme que prend le graphe sous-jacent de \mathcal{M}' lorsque le graphe sous-jacent de \mathcal{M} est complet et que $w(\alpha) = 1$ pour tout $\alpha \in A$.

Maintenant qu'on a construit \mathcal{M}' , il reste à résoudre le problème d'accessibilité de $(s, 0) \in S'$ à T' . Pour ce faire, on calcule $\mathbb{P}_{(s,0)}^{\max}(\Diamond T')$ par le théorème 3.2, et ensuite, si $\mathbb{P}_{(s,0)}^{\max}(\Diamond T') \geq b$, alors on résout le problème des plus courts chemins stochastiques pour \mathcal{M} depuis l'état s jusqu'au sous-ensemble T de taille limitée par l avec un seuil de probabilité supérieur à b . Dès lors, il existe une stratégie optimale σ pour laquelle

$$\mathbb{P}_s^\sigma(\{\pi \in \text{Paths}(s) \mid TS^T(\pi) \leq l\}) \geq b$$

Algorithme 10 Résoudre le problème des plus courts chemins stochastiques de taille limitée pour un état de \mathcal{M}

Entrées :

- $\mathcal{M} = (S, A, \Delta, w)$, un PDMP.
- $T \subseteq S$, un sous-ensemble d'états cibles.
- $s \in S$, l'état pour lequel on veut calculer le problème des plus courts chemins stochastiques de taille limitée.
- $l \in \mathbb{N}$, un seuil de longueur.
- $b \in [0, 1]$, le seuil de probabilité.

Sortie :

- *Vrai* si il existe une stratégie optimale qui résout le problème des plus courts chemins stochastiques pour le seuil de longueur l et le seuil de probabilité b .

```

1:  $(\mathcal{M}', T') \leftarrow construirePDMdéplié(\mathcal{M}, T, s, l)$ 
2:  $v \leftarrow Pr^{\max}(\mathcal{M}', T')$ 
   {on suppose que  $(s, 0)$  est le premier sommet de  $\mathcal{M}'$  }
3: si  $v[0] \geq b$  alors
4:    $\sigma \leftarrow ConstruireStratégieOptimaleAccessibilité(\mathcal{M}', v)$ 
5:   retourner Vrai
6: sinon
7:   retourner Faux

```

Comme le calcul de l'accessibilité de \mathcal{M}' à T' est polynomial en la taille de \mathcal{M}' et comme celle-ci dépend de l , de $|S|$ et de $|A|$, la complexité en temps de l'accessibilité est bien pseudo-polynomiale en fonction de l'encodage de l .

4.3 Benchmarks

On va maintenant générer des processus décisionnels de Markov afin de mesurer le temps CPU requis pour calculer le problème d'espérance du plus court chemin stochastique ainsi que celui des plus courts chemins stochastiques de longueur limitée. Les tests sont réalisés sur un ordinateur équipé d'un processeur Intel Core i5-3470 cadencé à 3.2 GHz, de 8 Go de mémoire ram et sous Windows 10 Professionnel. Soit $\mathcal{M} = (S, A, \Delta, w)$, un PDMP. Le temps de calcul de la résolution des deux problèmes dépend de la taille de \mathcal{M} . On va donc considérer le pire cas, i.e., le cas où la taille de \mathcal{M} est la plus grande. Il s'agit du cas où $\forall s \in S, |A(s)| = |A|$ et $\forall \alpha \in A(s), \forall s' \in S \Delta(s, \alpha, s') > 0$. On appelle ce type de PDMP un PDMP *complet*. En effet, dans ce cas, soit $G^{\mathcal{M}} = (S, E)$, $|\mathcal{M}| = |E| |A|$. Les transitions des PDMPs que l'on va générer auront des probabilités arbitraires et différentes pour toute action, i.e., $\forall s, s' \in E, \forall \alpha_1, \alpha_2 \in A(s), \Delta(s, \alpha_1, s') \neq \Delta(s, \alpha_2, s')$. La fonction de poids de ces PDMP renvoie toujours 1, de sorte à traiter le pire cas du problème des plus courts chemins stochastiques de taille limitée.

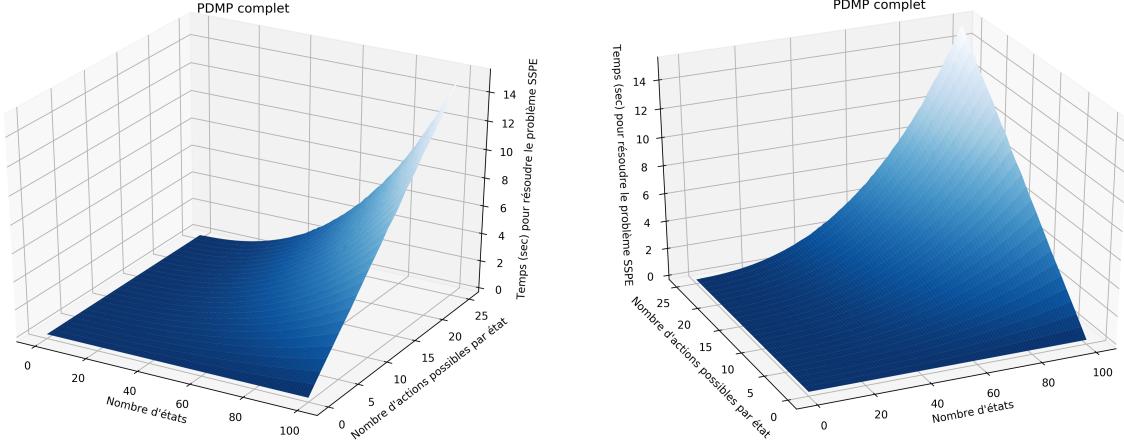


FIGURE 4.3 – Résolution du problème de l’espérance du plus court chemin stochastique en faisant varier le nombre d’actions et d’états du PDMP complet.

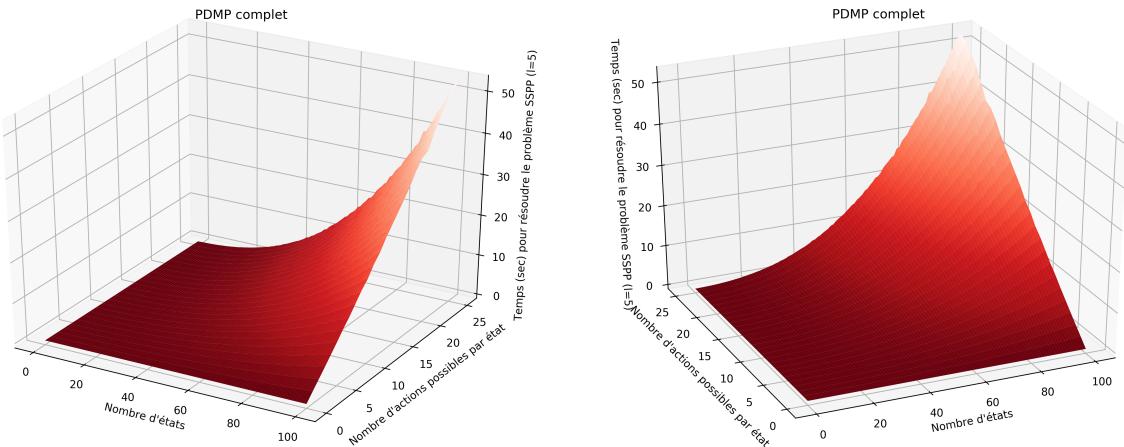


FIGURE 4.4 – Résolution du problème des plus courts chemins stochastiques de taille limitée en faisant varier le nombre d’actions et d’états du PDMP complet. Le seuil de longueur est constant.

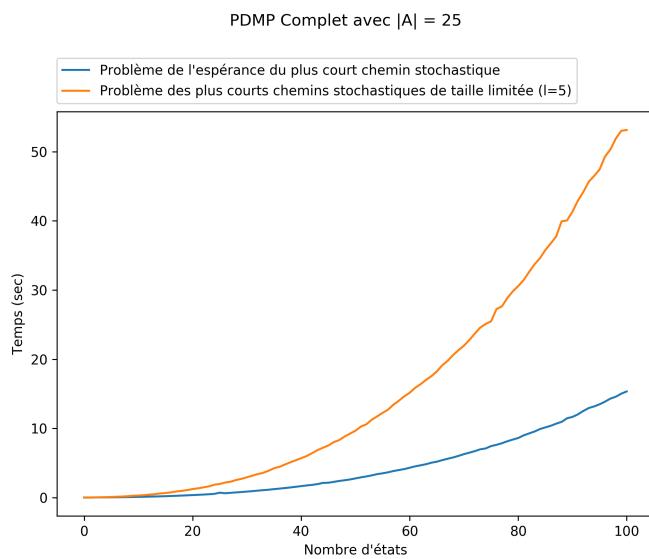


FIGURE 4.5 – Résolution des 2 problèmes en faisant varier le nombre d’états et en fixant le nombre d’actions. Le seuil de longueur est constant.

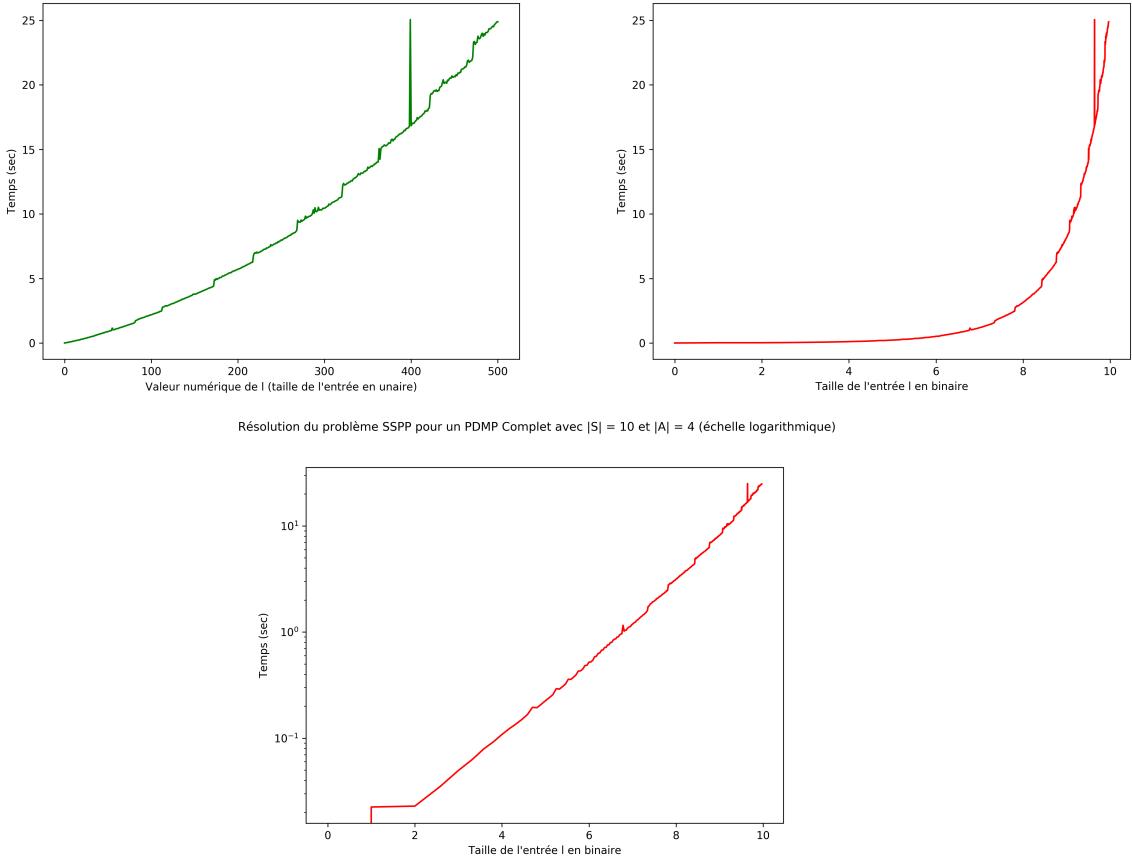


FIGURE 4.6 – Résolution du problème des plus courts chemins stochastiques de taille limitée en fixant la taille du PDM et en faisant varier le seuil de longueur.

Les graphiques générés (cf. figures 4.3, 4.4 et 4.5) suggèrent effectivement que le temps de calcul pour la résolution des deux problèmes est représenté par une fonction polynomiale en la taille de \mathcal{M} quand le seuil de longueur l est fixe. Si on fait varier l pour le problème des plus courts chemins stochastiques de taille limitée par l en fixant la taille du PDM, le temps d'exécution est polynomial en la valeur numérique de l (i.e., en la taille de la représentation unaire de l), mais est exponentiel en la taille de l'entrée l (i.e., en la taille de la représentation binaire de l) (cf. figure 4.6). Cela reflète le caractère pseudo-polynomial de ce problème.

On génère ensuite des PDMP aléatoires présentant les propriétés suivantes :

- Le PDMP généré est complètement aléatoire.
- Le graphe sous-jacent du PDMP généré est complet.
- Soit s , un état du PDMP généré. Certaines actions $\alpha \in A(s)$ respectent la propriété suivante : $\Delta(s, \alpha, s) = 1$. Chaque action possible de s a 70% de chance de posséder cette propriété. Cela a pour effet de rendre le graphe sous-jacent du PDMP faiblement connexe à un ensemble d'états cibles.
- Pour tout sommet s du PDMP généré, $|A(s)| = |A|$.
- Le graphe sous-jacent du PDMP généré est complet et, pour tout sommet s de ce PDMP, $|A(s)| = |A|$.

On mesure ensuite le temps CPU pour résoudre les différents problèmes abordés dans

ce document en faisant varier le nombre de sommets et d'actions des PDMP générés. Les logs du benchmark complet sont disponibles en annexe (cf. figure 4.12). Ceux-ci nous permettent de constater que

- Lorsqu'on fixe le nombre de sommets d'un PDMP, augmenter le nombre d'actions augmente considérablement le temps de calcul. En effet, les problèmes se résolvent avec une génération de PL. Chaque action supplémentaire engendre une contrainte supplémentaire par état et engendre donc des calculs supplémentaires par le solveur de PL. Cette tendance s'intensifie lorsqu'on force les PDMP à avoir un nombre strict (plus précisément, le nombre maximum) d'actions possibles par état.
 - Lorsqu'on rend le graphe sous-jacent du PDMP faiblement connexe à un sous-ensemble d'états cibles T , on constate que le temps de calcul augmente très fortement pour le problème d'accessibilité à T . En effet, comme on force les états du PDMP à être connectés avec des états qui ne sont pas connexes à T , cela engendre un grand nombre de contraintes supplémentaires pour le PL car il est nécessaire d'ajouter des contraintes pour les états s tels que $0 < \mathbb{P}_s^{\max}(\Diamond T) < 1$ et qui ne sont donc pas en sortie du prétraitement qui consiste à trouver les états tels que $\mathbb{P}_s^{\max}(\Diamond T) = 1$. Ensuite, on remarque que la résolution du problème de l'espérance du plus court chemin stochastique se fait, au contraire, plus rapidement. En effet, comme le graphe sous-jacent est faiblement connexe à T , tous les états s pour lesquels $\mathbb{P}_s^{\max}(\Diamond T) < 1$ ont une espérance fixée à ∞ . Il y a donc moins de contraintes générées pour le PL. En ce qui concerne le problème des plus courts chemins stochastiques de taille limitée, le temps de calcul est également réduit par rapport aux autres PDM car un grand nombre d'états est non-connexe à T , ce qui est traité par l'algorithme de parcours en largeur arrière. Le nombre de contraintes du PL est donc plus faible.
 - En terme de temps de calcul, une tendance s'intensifie avec la taille des PDM aléatoires générés. En effet, dans tous les cas, on aura pour les deux problèmes de plus court chemin stochastique que :
- PDM faiblement connecté à $T <_{\text{tempsCPU}}$ PDM complètement aléatoire $<_{\text{tempsCPU}}$
 PDM avec un graphe sous-jacent complet $<_{\text{tempsCPU}}$ PDM avec un nombre fixe d'actions par état $<_{\text{tempsCPU}}$ PDM avec un nombre fixe d'actions par état et avec un graphe sous-jacent complet. Cette tendance correspond bien à la complexité en temps théorique des différents problèmes.

4.4 Utilisation du programme

4.4.1 Importer des Processus Décisionnels de Markov

Afin de créer les PDM avec plus de facilité, on utilise le langage YAML, qui est un langage permettant de représenter des données par sérialisation unicode. Le but principal de ce langage est d'encoder des informations, comme le fait par exemple le langage XML, mais avec une lisibilité plus simple et de façon plus intuitive. Pour encoder

un PDMP, on utilisera la syntaxe suivante :

```
mdp:
  states:
    - name: s #label d'un état du PDMP
      enabled actions:
        - name: alpha #label d'une action possible de s
          transitions:
            - target: s' #label d'un successeur alpha de s
              probability: #Δ(s, alpha, s')
            - target: ...
        - name: ...
    - name: ...
  actions:
    - name: alpha #label d'une action du PDMP
      weight: #coût de l'action alpha : w(alpha)
    - name: ...
```

Exemple 4.1. On veut importer dans notre programme un simple PDMP. Le YAML utilisé pour importer le PDMP est le suivant :

```
mdp:
  states:
    - name: s
      enabled actions:
        - name: beta
          transitions:
            - target: t
              probability: 1/2
            - target: u
              probability: 1/2
    - name: t
      enabled actions:
        - name: gamma
          transitions:
            - target: s
              probability: 1
    - name: u
      enabled actions:
        - name: alpha
          transitions:
            - target: u
              probability: 1
            - name: gamma
              transitions:
                - target: s
                  probability: 1
  actions:
    - name: alpha
      weight: 5
    - name: beta
      weight: 3
    - name: gamma
      weight: 2
```

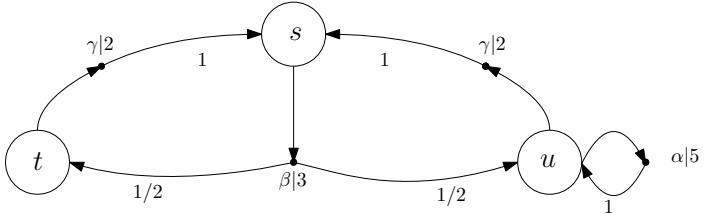


FIGURE 4.7 – Simple PDMP généré à partir du YAML

4.4.2 Exporter des Processus Décisionnels de Markov YAML

Le programme offre la possibilité de sérialiser une instance de PDM sous la forme d'un fichier YAML, avec la syntaxe décrite à la sous-section précédente.

Graphviz

Le programme offre la possibilité d'exporter sous forme d'image vectorielle les instances de PDM. Tout PDM instancié peut être exporté à l'aide du package **GraphViz** de Python.

Exemple 4.2. Reprenons le PDMP importé de l'exemple 4.1. **GraphViz** permet d'exporter le PDMP sous la forme d'une image vectorielle (cf. figure 4.8).

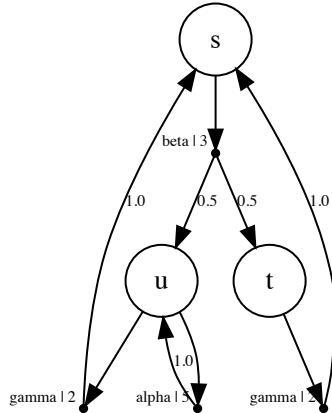


FIGURE 4.8 – Simple PDMP de l'exemple 4.1 exporté à l'aide de **GraphViz**

4.4.3 Générer des Processus Décisionnels de Markov

Il est possible de générer des PDM de façon aléatoire ou non, avec les propriétés citées à la section 4.3 (cf. figure 4.11 en annexe).

4.4.4 Solveurs

En ce qui concerne l'implémentation des solveurs, ces derniers permettent de récupérer les stratégies sous forme de fonctions, mais il est également possible de lancer chaque solveur comme programme, en passant en argument le label des états, les seuils, etc. Les solutions optimales des différents problèmes sont renseignés sur la sortie standard, tandis que `graphviz` dessine le PDMP en mettant en évidence en rouge les actions choisies par la stratégie si celle-ci existe.

Exemple 4.3 (Espérance du plus court chemin stochastique pour un agent dans un labyrinthe stochastique). Soit \mathcal{M}_{maze} , le PDMP de l'exemple 3.9. On veut résoudre le problème d'espérance du plus court chemin stochastique de l'état $(1, 1)$ aux états cibles $T = \{t_1, t_2\}$ sous un seuil de longueur 10 (cf. exemple 3.10). On lance le programme :

```
python3 solvers/sspe.py examples/agent_stochastic_maze.yaml --from '(1, 1)' --threshold 10 t1 t2
```

La sortie du programme est la suivante :

```
v[(1, 1)] = 9.83051  v[(1, 2)] = 10.7288
v[(1, 2)'] = 10.8305  v[(1, 3)] = 9.72881
v[(1, 4)] = 14.3559  v[(2, 1)] = 8.35593
v[(2, 3)] = 1  v[t1] = 0
v[t2] = 0  v[(4, 2)] = 13.8305
v[(4, 3)] = 4.35593  v[(5, 3)] = 19.7288
```

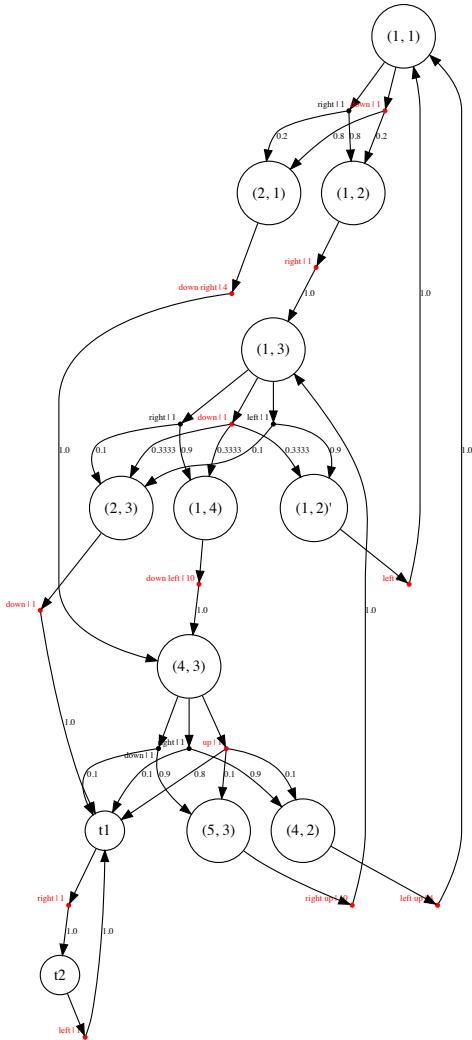


FIGURE 4.9 – PDMP $\mathcal{M}_{\text{maze}}$ sur lequel la stratégie a été mise en évidence par le programme.

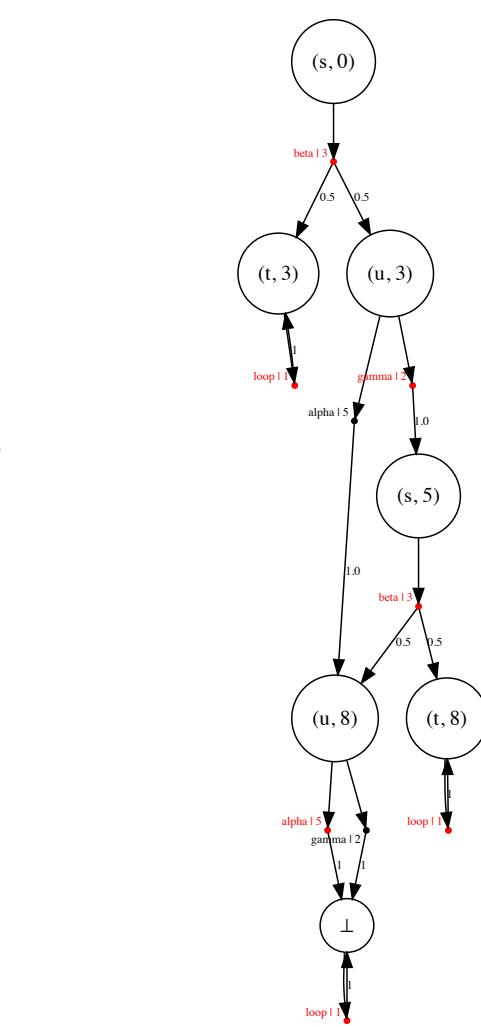


FIGURE 4.10 – PDM $\mathcal{M}'_{\text{simple}}$ sur lequel la stratégie a été mise en évidence par le programme.

Comme $v_{(1,1)} = 9.83$, on a bien qu'il existe une stratégie σ optimale telle que $v_{(1,1)} = \mathbb{E}_{(1,1)}^{\sigma}(TST) \leq 10$. Le programme construit la stratégie et illustre les actions choisies par celle-ci sur le PDMP (cf. figure 4.9).

Exemple 4.4 (Résolution du problème des plus courts chemins stochastiques de taille limitée sur un PDM simple). Soit $\mathcal{M}_{\text{simple}}$, un PDMP (cf. figure 4.7). On souhaite résoudre le problème des plus courts chemins stochastiques de longueur limitée par le seuil de taille 8 depuis l'état s (cf. exemple 3.11) et cela avec une probabilité supérieure au seuil $\frac{3}{4}$. On lance le programme :

```
python3 solvers/sspp.py examples/simple_mdp.yaml s 8 0.75 t
```

La sortie du programme est la suivante :

```
v[(s, 0)] = 0.75    v[(t, 3)] =      1
```

```
v[(u, 3)] = 0.5   v[(u, 8)] =     0  
v[(s, 5)] = 0.5   v[(t, 8)] =     1  
v[bot]        =     0
```

Comme $v_{(s,0)} = 0.75$, on a bien qu'il existe une stratégie optimale σ qui résout le problème, avec $v_{(s,0)} = \mathbb{E}_{(s,0)}^\sigma \geq 0.75$. Le programme construit la stratégie et illustre les actions choisies par celle-ci sur le PDMP (cf. figure 4.10).

Conclusion

Le but de ce projet était d'implémenter les processus décisionnels de Markov ainsi que les deux problèmes de plus court chemin stochastique. Pour ce faire, on a d'abord introduit les notions de probabilités, indispensables à la compréhension de ces problèmes, à savoir la notion de σ -algèbre, de mesure de probabilité, espace probabiliste, distribution de probabilité et espérance mathématique.

Ensuite, nous avons introduit les CM à temps discret. On a alors observé qu'il existe un espace probabiliste sur les chemins de toute CM et que la probabilité de ces chemins est mesurable. Premièrement, cela nous a permis de résoudre le problème d'accessibilité dans une CM. On a constaté que le problème d'accessibilité est résolvable à l'aide d'un système d'équations linéaires. On a alors défini ce qu'était une CM pondérée, qui est en réalité une CM à temps discret dont les transitions sont enrichies par une fonction de poids. Deuxièmement, l'étude du problème d'espérance de l'accessibilité a été réalisée sur de telles CM. On a remarqué que ce problème était également résolvable à l'aide d'un système d'équations linéaires. Troisièmement, on a étudié le problème d'accessibilité limitée par un coût l . On a vu qu'il est nécessaire de déplier la CM jusqu'à ce que la somme tronquée de ses chemins dépasse la longueur l et que résoudre le problème d'accessibilité sur cette CM dépliée revenait en réalité à résoudre le problème d'accessibilité limitée par le coût l .

Par la suite, on a défini ce qu'était un PDM afin d'étudier les deux problèmes de plus court chemin stochastique. On a vu qu'il n'existe pas d'espace probabiliste sur les chemins des PDM dû au non-déterminisme lié au processus de décision de ce modèle. On a donc résolu le non-déterminisme à l'aide de stratégies, qui induisent des chaînes de Markov sur lesquelles il est possible de mesurer la probabilité des chemins. Trois types de stratégies ont été abordées, à savoir les stratégies générales (à mémoire infinie), les stratégies à mémoire finie et les stratégies sans mémoire. Premièrement, on a abordé le problème d'accessibilité dans un PDM et le fait que ce problème se résout à l'aide d'un programme linéaire. La solution optimale de ce programme linéaire permet de définir une stratégie optimale qui résout le problème. On a alors abordé l'existence des PDM pondérés. Ceux-ci sont en réalité des PDM pour lesquels une fonction de poids est définie sur les actions. Deuxièmement, on a étudié le problème d'espérance du plus court chemin stochastique. Ce problème peut se résoudre par un programme linéaire dont la solution optimale permet de construire une stratégie optimale sur le PDM qui résout ce problème. Cette stratégie peut donc être construite en temps polynomial en la taille du PDM. Troisièmement, on a abordé le problème des plus courts chemins stochastiques de taille limitée par l . Afin de résoudre ce problème, on a vu qu'il est nécessaire de

déplier le PDM jusqu'à ce que la somme tronquée de ce PDM soit supérieure à l . On a finalement vu que la résolution du problème d'accessibilité sur ce PDM déplié revient à résoudre le problème des plus courts chemins stochastiques limités par la taille l et que celui-ci est résolvable en temps pseudo-polynomial, en fonction de la taille de la représentation de l .

Bibliographie

- [1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*, chapter 10, pages 745–898. MIT Press, 2008.
- [2] Nicolas Gillis. *Modèles aléatoires de recherche opérationnelle*. UMONS, 2016.
- [3] Arnaud Guyader. Introduction aux probabilités. <http://www.lsta.lab.upmc.fr/modules/resources/download/labsta/Pages/Guyader/IntroProba.pdf>.
- [4] D. Knuth and A. Yao. *Algorithms and Complexity : New Directions and Recent Results*, chapter The complexity of nonuniform random number generation. Academic Press, 1976.
- [5] Mickael Randour. *Formal verification of computer systems : Model Checking Probabilistic Systems*. ULB, 2016.
- [6] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Variations on the stochastic shortest path problem. In *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*, pages 1–18, 2015.

Annexes

Génération aléatoire de PDMP et Benchmarks

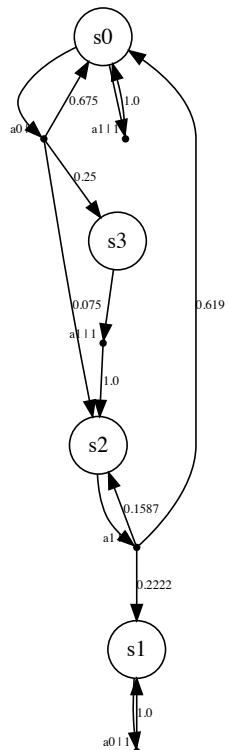


FIGURE 4.11 – Exemple de PDMP aléatoire généré avec 4 états et 2 actions.

Random MDP properties	# states (n)	# actions (a)	time to generate	reachability to T	SSPE to T	SSPP from s0 to T (l=5)
Any	100	5	0.061591	0.000598	0.874063	2.256931
Complete	100	5	0.062771	0.000641	0.948528	3.130787
Weakly connected to T	100	5	0.043328	0.513226	0.001885	1.631149
$ A(s) = a$	100	5	0.090549	0.000714	1.321811	4.819078
Complete & $ A(s) = a$	100	5	0.106925	0.000701	1.383723	5.174022
Any	100	10	0.096145	0.000670	1.399846	4.677277
Complete	100	10	0.107039	0.000703	1.483059	5.108018
Weakly connected to T	100	10	0.115053	1.041527	0.002855	3.309735
$ A(s) = a$	100	10	0.181859	0.000853	2.619248	9.981688
Complete & $ A(s) = a$	100	10	0.196197	0.000714	2.799335	10.846341
Any	100	20	0.252251	0.000783	2.777017	10.364763
Complete	100	20	0.260325	0.000753	2.814865	10.825774
Weakly connected to T	100	20	0.164216	2.043968	0.005472	7.533893
$ A(s) = a$	100	20	0.366944	0.000906	5.264044	22.280799
Complete & $ A(s) = a$	100	20	0.390464	0.000736	5.367351	22.506757
Any	100	50	0.498117	0.000812	7.046523	30.461853
Complete	100	50	0.499714	0.000804	6.862223	27.719840
Weakly connected to T	100	50	0.370560	0.000817	4.553729	17.931266
$ A(s) = a$	100	50	0.941298	0.000733	13.611149	67.500105
Complete & $ A(s) = a$	100	50	0.996318	0.000866	13.325984	65.980851
Any	200	5	0.258412	0.002378	4.748038	14.397156
Complete	200	5	0.250172	0.002523	5.366937	17.365234
Weakly connected to T	200	5	0.171181	2.931607	0.006174	6.268177
$ A(s) = a$	200	5	0.391847	0.002515	7.737186	27.231529
Complete & $ A(s) = a$	200	5	0.376895	0.002504	7.729245	29.121687
Any	200	10	0.512062	0.002489	8.709599	28.467161
Complete	200	10	0.429755	0.002902	9.047589	30.192939
Weakly connected to T	200	10	0.327784	5.718710	0.007702	19.268681
$ A(s) = a$	200	10	0.737520	0.002717	16.133774	63.763487
Complete & $ A(s) = a$	200	10	0.949529	0.002626	15.621766	62.684269
Any	200	20	0.807769	0.002807	16.461097	60.688702
Complete	200	20	0.839061	0.002589	17.244568	65.466203
Weakly connected to T	200	20	0.542157	9.638500	0.010367	29.748839
$ A(s) = a$	200	20	1.432856	0.002284	31.297535	145.807876
Complete & $ A(s) = a$	200	20	1.570347	0.002962	32.105518	151.451073
Any	200	50	1.941490	0.002464	40.754092	183.475744
Complete	200	50	2.279090	0.003247	37.132269	162.939896
Weakly connected to T	200	50	1.839604	27.651988	0.036093	110.598731
$ A(s) = a$	200	50	3.703754	0.002166	77.896352	468.165368
Complete & $ A(s) = a$	200	50	3.956135	0.002561	78.727642	451.932957
Any	500	5	1.500487	0.014071	59.897426	168.570606
Complete	500	5	1.551755	0.016209	69.445136	215.439071
Weakly connected to T	500	5	1.131669	37.218555	0.060024	93.361819
$ A(s) = a$	500	5	2.232614	0.015755	99.381720	342.450293
Complete & $ A(s) = a$	500	5	2.470336	0.015753	102.121433	372.512135
Any	500	10	2.552147	0.015762	107.970088	343.001960
Complete	500	10	2.767859	0.015952	116.805562	372.962212
Weakly connected to T	500	10	1.991970	69.426019	0.046253	220.001869
$ A(s) = a$	500	10	4.593806	0.015554	199.958573	765.422272
Complete & $ A(s) = a$	500	10	4.858108	0.016609	202.398526	821.034947
Any	500	20	5.933944	0.016825	200.011500	610.372405
Complete	500	20	5.107917	0.016500	213.068183	738.781099
Weakly connected to T	500	20	5.084185	143.494319	0.051343	490.817506
$ A(s) = a$	500	20	9.298514	0.016249	400.248861	1835.560748
Complete & $ A(s) = a$	500	20	9.768658	0.016856	391.946658	1772.984561
Any	500	50	12.108991	0.018037	514.890148	2397.729348
Complete	500	50	12.186274	0.017590	501.656399	2163.867175
Weakly connected to T	500	50	9.515607	383.929057	0.085080	1406.942466
$ A(s) = a$	500	50	23.620388	0.017172	991.854540	6250.731632
Complete & $ A(s) = a$	500	50	26.766797	0.018948	993.989694	5996.783858
Any	1000	5	8.225531	0.057768	471.145356	1327.996031
Complete	1000	5	6.589358	0.063697	565.398221	1702.675928
Weakly connected to T	1000	5	4.658813	292.596662	0.146345	767.875632
$ A(s) = a$	1000	5	9.193412	0.059480	801.636856	2953.730398
Complete & $ A(s) = a$	1000	5	9.975392	0.066582	826.089594	3097.353540
Any	1000	10	9.975131	0.062431	841.508871	2673.272387
Complete	1000	10	11.155221	0.065848	927.889785	3047.011120
Weakly connected to T	1000	10	8.591634	598.551723	0.181327	1868.813994
$ A(s) = a$	1000	10	18.689653	0.065067	1636.531268	6807.720383
Complete & $ A(s) = a$	1000	10	21.828283	0.067625	1626.185402	6883.157484
Any	1000	20	21.571935	0.065998	1696.871592	5868.766838
Complete	1000	20	22.697073	0.069220	1759.433657	6333.157337
Weakly connected to T	1000	20	16.438283	1199.861903	0.208766	4236.922507
$ A(s) = a$	1000	20	38.065079	0.066905	3374.658104	21095.683374
Complete & $ A(s) = a$	1000	20	98.708795	0.069787	3386.503908	

FIGURE 4.12 – Logs de la génération aléatoire de PDMP comme décrit à la section 4.3.