UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

# EXPLORATION OPTIMALE D'UN SEGMENT DE DROITE PAR DEUX AGENTS MOBILES

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

FRÉDÉRICK LESSARD

JANVIER 2013

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Ce mémoire intitulé :

# EXPLORATION OPTIMALE D'UN SEGMENT DE DROITE PAR DEUX AGENTS MOBILES

présenté par
Frédérick Lessard

pour l'obtention du grade de maître ès science (M.Sc.)

a été évalué par un jury composé des personnes suivantes :

Dr. Jurek Czyżowicz ..... Directeur de recherche
Dr. Andrzej Pelc ..... Co-directeur de recherche
Dr. Larbi Talbi ..... Président du jury
Dr. Mohand Saïd Allili ..... Membre du jury

Mémoire accepté le : 9 janvier 2013

*Je dédie ce mémoire à Valérie, je t'aime. Je dédie aussi ce mémoire à ma grande famille et à mes amis pour leur patience.*

# Remerciements

Merci à Marie-Eve et Carmen pour leur aide à la correction de ce document. Merci aussi aux Dr. Jurek Czyżowicz et Dr. Andrzej Pelc pour une direction hors pair et l'équilibre qu'ils ont apporté à nos discussions. Finalement, merci maman et papa pour votre support tout au long de mes études.

# Liste des figures

# Table des matières

# Résumé

Le sujet de ce mémoire est l'exploration optimale d'un segment par deux agents mobiles ayant une connaissance complète de l'environnement. Chaque point du segment doit être visité par au moins un agent et le temps d'exploration doit être minimisé. Aucune étude complète de ce sujet n'a été publiée à ce jour. Ce mémoire vient combler ce vide et présente la combinaison optimale de chemins de chaque agent pour explorer tous les points d'un segment de droite. Cette étude a été réalisée grâce à l'élaboration d'un algorithme et des preuves nécessaires à la validation de ce dernier : la preuve d'exactitude et la preuve d'optimalité du parcours produit. La justification rigoureuse de ces dernières constitue le défi principal du présent mémoire.

# Abstract

The subject of this Master's thesis is the optimal exploration of a line segment by two mobile agents having complete knowledge of the environment. Each point of the segment must be visited by at least one agent and the exploration time must be minimized. To date, no comprehensive study of this subject has been published. This thesis fills this gap and presents the optimal combination of agent paths in order to explore all the points of a line segment. The results of this study were obtained via the construction of an algorithm and the proofs required to validate this algorithm : the correctness proof and the optimality proof of the algorithm's result. The rigorous justification of the latter properties is the main challenge of the following Master's thesis.

# Chapitre 1

# Introduction

L'exploration par une paire d'entités est le filon de ce mémoire. Nous discuterons de l'exploration d'un environnement dans son sens le plus large. Rao et al. [30] expliquent que l'exploration dans un environnement connu permet la planification *a priori* de chemins, tandis que pour l'exploration dans un environnement inconnu, les chemins doivent être élaborés au fur et à mesure que les entités progressent. Le sujet principal du présent mémoire est l'exploration d'un environnement connu, plus précisément, d'un segment de droite.

L'exploration d'un environnement par un groupe d'entités est un problème algorithmique fondamental ayant des applications directes dans plusieurs domaines, soient-ils physiques ou virtuels. En effet, on peut penser à l'exploration d'un labyrinthe par plusieurs personnes à la recherche d'un trésor, l'exploration d'un environnement inhabitable (tel qu'une planète lointaine) par un groupe de robots, la planification des meilleures routes pour la livraison de biens par une flotte de véhicules, l'inspection d'un bâtiment par un groupe de pompiers à la recherche de victimes ou l'exploration de nœuds dans un réseau informatique.

Dans le cadre de ce mémoire, nous considérons des entités ayant une capacité inhérente d'exploration qui sont souvent utilisées dans la littérature : les agents mobiles. Comme beaucoup de définitions des agents mobiles sont étudiées, nous utiliserons une définition similaire à celle de la plupart des auteurs cités dans le présent mémoire et très bien formulée dans l'article de Kranakis et Krizanc [24]. En effet, dans [24], Kranakis et Krizanc décrivent un agent mobile comme une entité accomplissant une tâche pour l'utilisateur et étant capable de se déplacer de façon autonome. Kranakis et Krizanc énumèrent quatre propriétés fondamentales des agents mobiles :

- *L'autonomie :* Un agent mobile devrait avoir un certain degré d'indépendance de son créateur. Certaines décisions devraient pouvoir être prises sans l'intervention d'une autorité centrale.
- *La mobilité :* Un agent mobile devrait avoir la capacité de se déplacer en conservant sa mémoire et ses capacités de traitement.
- *L'interaction avec son environnement :* Un agent mobile devrait être capable d'interagir avec les éléments visités ainsi qu'avec l'environnement. Ceci peut être fait à l'aide de capteurs visuels, tactiles, etc.
- *L'intelligence :* Un agent mobile devrait avoir la capacité de modifier son comportement afin de mieux répondre aux objectifs en utilisant des données recueillies au cours d'opérations précédentes.

Même si Kranakis et Krizanc parlent spécifiquement d'agents logiciels [1], les propriétés susmentionnées sont aussi applicables pour d'autres types d'agents mobiles, tels que les robots. Dans le cadre de ce mémoire, nous supposons des agents mobiles quelconques (logiciel, robotique ou autre).

1. Des processus informatiques autonomes s'exécutant sur des nœuds dans un système distribué.

## 1.1 Modèle

Ce mémoire discute spécifiquement de l'utilisation de deux agents mobiles n'ayant aucune restriction particulière lors de l'exploration d'un segment de ligne dans un environnement connu. Les agents ont un emplacement arbitraire sur le segment, mais toutes les données nécessaires pour l'exploration sont connues lors de l'élaboration des chemins permettant l'exploration complète du segment. Plus précisément, chaque agent connaît la longueur du segment et les positions des deux agents.

## 1.2 Problématique et objectifs

Ce mémoire vise à trouver le temps optimal d'exploration d'un segment par deux agents mobiles et les chemins parcourus par chacun des agents afin d'exécuter ladite exploration. Le parcours d'exploration est correct si chaque point du segment est visité par au moins un des agents. Le parcours d'exploration est optimal si le plus long des deux parcours construits est minimal parmi tous les parcours d'exploration corrects. Aucune étude n'a été publiée sur la façon optimale d'explorer un segment dans de telles conditions. Nous allons décrire un parcours d'exploration, trouver son temps d'exécution et faire les preuves d'exactitude et d'optimalité de ce dernier.

## 1.3 Méthodologie

Tout d'abord, nous allons construire le parcours d'exploration optimal d'un segment de ligne par un seul agent, suivi de la construction du parcours pour deux agents. Ensuite, nous allons prouver que ce dernier visite chaque point du segment et que la longueur est optimale.

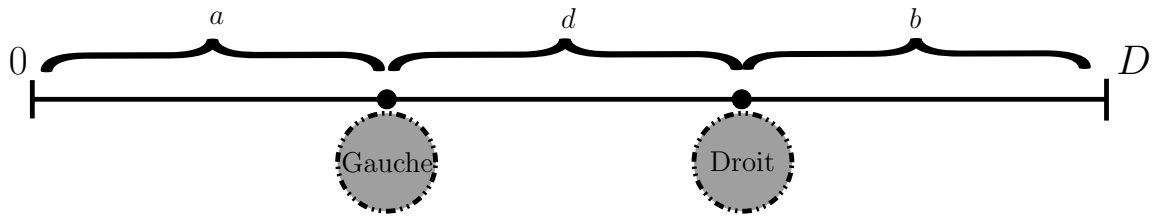


FIGURE 1.1 – Le segment de droite à explorer.

## 1.4 Résultats

Tel qu'illustre la figure 1.1, nous dénoterons la distance entre les deux agents avec $d$, la distance entre l'extrémité gauche et l'agent gauche avec $a$ et la distance entre l'extrémité droite et l'agent droit avec $b$. À la conclusion de ce mémoire, nous aurons démontré que, dans l'exploration optimale par deux agents d'un segment de ligne dans un environnement connu, le plus long des deux parcours a la longueur $minimum(\frac{2d+b+a}{2}, \frac{4b+2d+a}{3}, \frac{4a+2d+b}{3}, \frac{2a+2b+d}{2})$. De plus, nous aurons élaboré un algorithme qui produit les chemins d'un parcours optimal pour chacun des deux agents et nous prouverons son exactitude et son optimalité. La justification rigoureuse de ces dernières constitue le défi principal du présent mémoire.

# Chapitre 2

# État des connaissances

L'exploration d'un environnement par une ou des entités (soit des humains, des véhicules, des robots, ou des agents logiciels) est un sujet ayant capté l'intérêt de nombreux chercheurs, particulièrement dans les domaines des mathématiques et de l'informatique, depuis plusieurs décennies. Dans son sens le plus large, *l'exploration* se définit par le fait de percevoir (soit physiquement, visuellement, virtuellement, ou avec un capteur quelconque dans le cas d'un robot) tous les éléments de l'environnement dans un but donné. Le but de l'exploration peut être, par exemple : la découverte d'un objet, l'exécution d'une tâche à un endroit dans l'environnement, la cartographie de l'environnement, etc. L'efficacité des algorithmes d'exploration est souvent évaluée par rapport au chemin le plus court menant au but désiré. Un algorithme d'exploration est dit optimal si le coût du chemin produit par ce dernier n'est pas plus élevé que celui du chemin visitant tous les endroits à explorer à coût minimal. D'autre part, un algorithme d'exploration est dit « compétitif », si le ratio du coût de ce dernier par rapport au plus petit coût d'un tel chemin est borné par une constante [12]. Le « surplus » (« overhead ») est la borne supérieure (ou le supremum) de ce ratio [21]. Finalement, le coût est souvent défini comme la longueur du chemin parcouru.

Les problèmes d'exploration peuvent être classés selon plusieurs aspects : la connaissance de l'environnement, le type d'environnement, le synchronisme des opérations faites par les entités, le nombre d'entités en action, les qualités propres aux entités (p. ex. la capacité sensorielle d'un type d'entité), etc. Pour les besoins du sujet du présent mémoire, à savoir l'exploration par deux agents mobiles d'un environnement simple et connu, l'état des connaissances sera présenté en utilisant les trois aspects suivant : la connaissance de l'environnement, le nombre d'entités en action, ainsi que le type d'environnement. La revue de littérature qui suit divisera les travaux connexes à ce mémoire, d'abord selon le niveau de connaissance disponible sur l'environnement et ensuite selon le nombre d'entités en action pour exécuter l'exploration. Finalement, la littérature pertinente sera divisée à nouveau selon le type d'environnement (soit un terrain géométrique, soit un graphe).

## 2.1 Exploration dans un environnement inconnu

Un environnement inconnu est défini par le fait que l'information nécessaire à l'exploration à coût optimal n'est pas disponible lors de l'exécution de l'algorithme d'exploration. Par exemple, dans l'étude faite par Rao et al. [30], un terrain inconnu est défini par le manque d'information sur le modèle du terrain (les obstacles, les chemins, la carte du terrain, etc.). D'autres auteurs utilisent cette même classification (p. ex. [9, 12]). De même, le manque de connaissances dans un environnement non géométrique, tel un graphe, peut être défini par le manque d'information sur l'identification des sommets ou des arêtes (p. ex. [2, 6, 7, 8, 17, 21, 18]).

### Exploration d'un terrain géométrique inconnu par un robot

L'étude de Rao et al. [30] aborde spécifiquement l'exploration d'un terrain géométrique inconnu avec un robot et énumère une panoplie de méthodes

d'exploration datant d'aussi longtemps que 1873. Les auteurs caractérisent les différents algorithmes d'exploration selon deux critères : l'objectif de l'algorithme et la capacité sensorielle du robot. Plus précisément, les auteurs élaborent une taxinomie d'algorithmes non-heuristiques [1], d'abord selon les trois classes suivantes :

Classe A : Algorithmes garantissant uniquement l'exactitude du résultat et non la performance ;

Classe B : Algorithmes cherchant à optimiser certains paramètres de la solution du problème (p. ex. nombre d'opérations visuelles ou la distance parcourue) ;

Classe C : Algorithmes utilisant un modèle de robot restrictif (p. ex. l'utilisation d'un automate fini ayant une capacité de calcul limitée).

Ensuite, une sous-classification est effectuée selon la capacité sensorielle (tactile ou visuelle), ou une particularité des algorithmes observés. Dans le cas de la capacité sensorielle visuelle, les auteurs considèrent un robot doté d'une vision continue comme étant doté d'une mémoire contenant l'information sur tous les points visibles à partir de chaque point du chemin parcouru, tandis que ceux dotés d'une vision discrète n'ont qu'un sous-ensemble de points limités en raison du nombre fini d'opérations sensorielles.

Similairement au modèle d'exploration d'un terrain inconnu, présenté dans l'étude de Rao et al., l'article de Berman et al. [9] considère un robot qui doit explorer l'environnement à l'aide de connaissances limitées. En effet, la connaissance de l'environnement se limite au point de départ du robot, sa position actuelle, ainsi que son point d'arrivée. De plus, les auteurs de [9] élaborent un algorithme stochastique visant la minimisation de la distance parcourue par le robot ayant un ratio compétitif espéré de $O(n^{\frac{4}{9}} \log(n))$ par rapport au parcours optimal dit de type « Manhattan » [2]. Deux des particularités de l'algorithme présenté sont qu'il utilise

1. Un type d'algorithme qui garantit l'exactitude du résultat.
2. Un parcours à angle droit, tout comme les rues du secteur *Manhattan* de la ville de New York aux États-Unis d'Amérique.

un robot avec une capacité sensorielle tactile et que l'environnement contient uniquement des obstacles rectangulaires. De plus, le point d'arrivée se trouve sur une ligne verticale à une distance horizontale $n$ du point de départ. Ce type de problème est communément appelé le problème du mur ou « wall problem ».

Un environnement contenant des obstacles rectangulaires et rectilignes est un modèle souvent utilisé afin de faciliter la modélisation de certains environnements réels. En effet, l'article de Deng et al. [12] utilise un modèle où les obstacles et la pièce à explorer sont rectilignes. Dans cet article, on nous rappelle que le fait de vérifier qu'un chemin est optimal parmi tous les chemins possibles lorsque l'environnement quelconque est connu, est un problème NP-difficile. En effet, les auteurs remarquent que la variante « euclidienne » du problème du « commis-voyageur » (connu comme étant NP-difficile) peut être réduite au problème évoqué par les auteurs.

Ceci étant dit, les auteurs de [12] élaborent un algorithme $k$-compétitif avec un temps polynomial pour le problème d'exploration complète d'une pièce rectiligne ayant $k$ obstacles rectilignes inconnus. On parle ici du problème de la visite d'une galerie d'art (« gallery tour problem ») ou du problème de l'inspection par un gardien de sécurité (« watchman route problem »). Ces deux problèmes sont uniquement différenciés par le fait d'avoir un point d'entrée et un point de sortie différents pour le problème de l'agent de sécurité. La compétitivité de l'algorithme présenté s'appuie sur des résultats préalables démontrant qu'il est possible de trouver le chemin optimal explorant le périmètre d'un polygone simple en temps $O(n^2)$ où $n$ représente le nombre de sommets. En fait, les auteurs développent un algorithme pour la visite d'une galerie représentée par un polygone simple ayant $n$ côtés et $m$ « extensions essentielles » (c.-à-d. les côtés d'une enveloppe convexe) avec un coût d'exécution $O(nm^2)$. À partir de ces résultats, il est simple de constater que cet algorithme peut être utilisé à répétition pour $k$ obstacles. Mentionnons que le problème de l'inspection

par un agent de sécurité ajouterait, dans le pire des cas, une constante de 2 afin de « retrouver » le point de sortie au lieu de terminer au point de départ.

Plus récemment, Dumitrescu et al. [16] ont proposé des algorithmes polynomiaux pour l'inspection d'un ensemble de lignes. Les auteurs de [16] proposent un algorithme utilisant la programmation dynamique pour un ensemble $\mathcal{L}$ de $n$ lignes non-parallèles et exécutant en temps $O(n^8)$. Ensuite, un algorithme $O(\log^3 n)$-approximatif basé sur l'algorithme initial est proposé pour un ensemble de segments dans un environnement multi-dimensionnel. Finalement, les auteurs de [16] proposent aussi un algorithme s'exécutant en temps $O(n \log n)$ pour un ensemble de segments formant une grille dans un plan.

## Exploration d'un graphe inconnu par un agent

De la même façon qu'un robot est modélisé par un agent mobile, un graphe est une représentation ou un modèle d'un environnement réel. En effet, un graphe peut modéliser une variété de réseaux (p. ex. informatique, routier, biomoléculaire, de Pert (gestion de projet)). Ces représentations permettent de faciliter l'application de théories provenant d'une variété de champs de recherche, par exemple, l'utilisation de la recherche avec un algorithme de parcours en largeur (*BFS* ou « Breadth-First Search »). Plus spécifiquement, Awerbuch et al. [6] utilisent le *BFS* afin de résoudre le problème où un robot mobile, ayant une borne sur la distance maximale entre deux retours à la base, doit explorer un graphe inconnu connexe non-dirigé « par étapes » (« piecemeal »). Une des particularités du type de graphe utilisé par les auteurs est que les sommets sont identifiables et uniques, mais la « carte routière » de ce graphe n'est pas connue. Pour cet environnement, les auteurs de [6] décrivent un algorithme approximatif basé sur le *BFS* avec un temps d'exécution $O(E + V^{1+o(1)})$.

Plus récemment, Duncan et al. [17] ont développé un algorithme ayant un coût linéaire $\theta|E|$ par rapport au nombre d'arêtes dans le cas où un robot est limité par un câble. De plus, les auteurs remarquent que le problème décrit par Awerbuch et al. [6] (c.-à-d. un robot avec une limite de carburant), peut aussi être résolu avec ce même algorithme et un coût du même ordre. Il est important de noter que les auteurs de [17] utilisent les mêmes contraintes que pour [6] avec une restriction supplémentaire sur la longueur maximale du câble. De plus, contrairement à certains auteurs, les auteurs de [17] spécifient la méthode de différenciation entres les nœuds. En effet, ils utilisent la notion de « marquage » d'un nombre illimité de nœuds afin de pouvoir différencier ces derniers entre eux.

Le « marquage » de nœuds est aussi utilisé dans l'article de Bender et al. [7]. Par contre, contrairement à [17], Bender et al. minimisent le nombre de nœuds à « marquer ». Les auteurs utilisent un *caillou* laissé sur l'emplacement du ou des nœuds visités, pour modéliser les besoins en mémoire afin de pouvoir assurer un suivi des nœuds visités. Les auteurs proposent un algorithme déterministe avec lequel tous les nœuds d'un graphe dirigé inconnu fortement connexe peuvent être visités en utilisant un seul *caillou*. Cette dernière affirmation n'est vraie que s'il y a une borne supérieure connue, $\hat{n}$, sur le nombre de nœuds. De plus, si la borne supérieure n'est pas connue, alors il est nécessaire d'utiliser $O(\log \log n)$ *cailloux* pour visiter tous les $n$ nœuds du graphe inconnu.

Dans certains cas, l'exploration d'un graphe inconnu, où le « marquage » n'est pas possible, doit être exécutée à répétition. Donc, il est important de trouver un chemin court pour effectuer cette exploration périodique. Dans ce sens, Czyżowicz et al. [10] prouvent que, si un agent n'a aucune mémoire persistante (« oblivious agents »), alors il est possible de créer un cycle de longueur $4\frac{1}{3}n - 4$ couvrant les $n$ nœuds d'un graphe non-dirigé inconnu en utilisant une numérotation d'arête (ou de port de sortie de nœud) spécialement conçue pour chaque nœud, en temps $O(|E|)$. De plus,

les auteurs de [10] montrent que, si un agent possède une mémoire d'ordre constant, la borne supérieure de la longueur du cycle est de $3.5n$, tandis que la borne inférieure est de $2.8n$.

Albers et Henzinger [2] abordent l'exploration de graphe d'un autre angle. Ces auteurs utilisent la notion de « déficience » (« deficiency ») d'un graphe directionnel inconnu fortement connexe, soit du même type de graphe que l'article [7]. En effet, les auteurs se fondent sur les travaux de Koutsoupias [23], et Deng et Papadimitriou [13], qui utilisent aussi la notion de « déficience ». Les auteurs décrivent la « déficience » $d$ comme étant le nombre d'arêtes, nécessaires à ajouter, afin que le graphe soit « eulérien ». Un graphe dirigé est dit « eulérien » s'il existe un chemin pouvant visiter toutes les arêtes de ce graphe une seule fois. En s'appuyant sur la notion de « déficience » $d$, Albers et Henzinger font la démonstration du premier algorithme sub-exponentiel ayant une borne supérieure de $d^{O(\log d)}m$ et une borne inférieure de $d^{\Omega(d)}m$, avec $m$ étant le nombre d'arêtes du graphe. De plus, les auteurs spécifient que l'algorithme de type « diviser pour régner » ne requiert pas, *a priori*, la connaissance de $d$, pour exécuter l'exploration.

Fleischer et Trippen [18] utilisent aussi la « déficience » $d$. Plus précisément, les auteurs présentent un algorithme « en ligne » (« online ») et déterministe ayant un ratio compétitif $O(d^8)$. De l'aveu des auteurs, cet algorithme récursif, utilisant le *BFS*, et exécutant dans un temps $O(d^8m)$, pourrait être amélioré. En effet, la conclusion de l'article [18] porte sur une amélioration possible de l'algorithme.

## Exploration d'un graphe inconnu par plusieurs agents

Jusqu'à maintenant, les articles présentés ont tenté de résoudre le problème de l'exploration avec un seul agent. Par contre, il est clair qu'une optimisation possible pour l'exploration d'un environnement serait l'ajout d'explorateurs additionnels.

Ainsi, bon nombre d'auteurs ont proposé de résoudre le problème avec plus d'un agent afin de bénéficier de la force de la collaboration. En effet, certains auteurs vont même jusqu'à démontrer que, lorsqu'un graphe est inconnu, et qu'il est impossible de différencier de quelconque manière les nœuds et que la taille du graphe est inconnue, alors un seul agent ne peut pas faire l'exploration (Bender et Slonim [8]).

Plus précisément, [8] indique que, même avec l'aide d'un ou de plusieurs *cailloux*, il est impossible pour un seul agent d'explorer un graphe directionnel fortement connexe si le nombre de nœuds est inconnu. Par contre, les auteurs démontrent qu'il est possible pour deux agents d'explorer un tel graphe en utilisant un algorithme probabiliste ayant une forte probabilité d'être exécuté en $O(d^2n^5)$ étapes, sans aucune information. Les auteurs de [8] spécifient que le degré maximal du graphe est identifié par $d$, tandis que le nombre de nœuds est dénoté par $n$.

Dans [8], les agents peuvent communiquer librement entre eux. Par contre, si une restriction est imposée sur la communication (p. ex. aucune communication possible), les auteurs indiquent qu'il suffit que les agents partagent une suite de caractères aléatoires afin de pouvoir mener à terme l'exploration au même coût. Plus récemment, Fraigniaud et al. [21] sont allés plus loin et ont démontré que, sans communication, un groupe de $k$ agents ne peut pas explorer, avec un surplus meilleur que $\Omega(k)$, certains graphes inconnus, non-orientés, acycliques et connexes (c.-à-d. certains arbres). De plus, les auteurs élaborent un algorithme d'exploration déterministe pour tout arbre inconnu travaillant avec un surplus de $O(k/\log k)$ si la communication est permise. Finalement, avec les algorithmes présentés, [21] démontre la force de la collaboration dans l'exploration d'un arbre avec une communication en *écriture-lecture* : les agents communiquent en laissant des messages sur les nœuds. En effet, les auteurs démontrent une accélération d'un ordre de grandeur lors de l'exploration d'un arbre avec $k$ agents au lieu d'un seul.

Plus récemment, Flocchini et al. [19] ont établi le nombre $k$ d'agents identiques, sans mémoire (« oblivious ») et asynchrones, nécessaire pour explorer, sans communication, une ligne de $n$ nœuds. Les auteurs de [19] prouvent qu'il est uniquement possible que de tels agents puissent explorer une ligne si $k = 3$, ou $k \geq 5$, ou $k = 4$ et que $n$ est impair. De plus, les auteurs proposent un algorithme d'exploration pour tout $k$ où l'exploration est possible.

## 2.2 Exploration d'un environnement connu

Un environnement connu ou partiellement connu est défini par le fait que l'information nécessaire à l'exploration à coût optimal est disponible ou partiellement disponible lors de l'exécution de l'algorithme d'exploration. En effet, dans le cas d'une exploration avec connaissances, les algorithmes bénéficient d'un avantage comparativement aux algorithmes vus dans les sections précédentes. Ces connaissances utiles peuvent prendre la forme d'une carte de l'environnement (p. ex. [1, 4, 5, 14, 22, 25, 26, 27, 29, 31, 32]), ou sous forme d'informations permettant la différenciation de nœuds ou d'arêtes (p. ex. [3, 11, 15]).

### Exploration d'un graphe connu ou partiellement connu par un agent

S'il est possible de différencier les nœuds ou les arêtes d'un graphe, mais que ces informations doivent être enregistrées par l'agent afin d'être utiles à l'exploration, il devient intéressant d'optimiser la quantité de mémoire nécessaire. Dans ce sens, Diks et al. [15] démontrent des bornes supérieures et inférieures de mémoire pour divers scénarios où les arêtes sortantes de tous les nœuds sont étiquetées. Effectivement, si le degré maximal $\Delta$ est connu pour un arbre, alors il est suffisant d'avoir $O(\log \Delta)$ bits de mémoire s'il n'est pas nécessaire de terminer sur un nœud précis. Par contre, pour ce même scénario, il est nécessaire d'utiliser $\Omega(\log \log \log n)$ bits de mémoire pour

un arbre ayant $n$ nœuds. S'il est nécessaire de terminer l'exploration sur un nœud spécifique, alors $\Omega(\log n)$ bits sont requis. Finalement, les auteurs de [15] présentent un algorithme nécessitant $O(\log^2 n)$ bits de mémoire pour tout arbre avec $n$ nœuds. Plus récemment, Gąsieniec et al. [3] ont amélioré cette borne supérieure en proposant un algorithme nécessitant $O(\log n)$ bits sous les mêmes restrictions.

Si l'on ajoute l'identification des nœuds au scénario précédent, on obtient maintenant une carte. Même incomplète, une carte peut s'avérer fort utile afin d'optimiser le temps d'exploration d'un graphe non-dirigé. En effet, l'article de Dessmark et Pelc [14] démontre que l'algorithme *DFS* (recherche en profondeur ou « Depth-First Search ») n'obtient pas le plus petit « surplus » possible, dans le cas d'un arbre, si certaines informations sont disponibles au sujet des nœuds. Plus précisément, Dessmark et Pelc décrivent deux scénarios où l'algorithme obtient un surplus optimal pour un arbre ou une ligne et confirment que le *DFS* a le surplus optimal pour la classe des graphes quelconques. Effectivement, si une carte décrivant les distances entre les nœuds et les arêtes les reliant (sans toutefois différencier les nœuds ou identifier le nœud de départ) est disponible, alors l'algorithme de [14] s'exécutera avec un « surplus » plus grand que $\sqrt{3}$, mais plus petit que 2. Finalement, si le nœud de départ est indiqué sur la carte, le « surplus » optimal est de $\frac{7}{5}$ pour la ligne et de $\frac{3}{2}$ pour les arbres.

Même avec une carte complète, certaines connaissances peuvent être manquantes dans le cadre de l'exploration optimale d'un graphe. Effectivement, explorer un graphe lorsqu'il existe des arêtes ou des nœuds « en panne » ou « fautifs », sans que leurs emplacements soient connus, complique la tâche d'un agent désirant explorer ce dernier. Par contre, Markou et Pelc [26] proposent deux algorithmes permettant l'exploration de tels graphes : un pour les lignes et l'autre pour les arbres. En présence de pannes inconnues, le « surplus » de l'algorithme explorant la partie accessible du graphe est la borne supérieure du ratio entre son coût et le coût de l'algorithme

optimal connaissant l'emplacement des pannes. Pour le problème avec une ligne, les auteurs décrivent un algorithme dont le « surplus » est optimal. En ce qui concerne les arbres, les auteurs proposent un algorithme basé sur une heuristique ayant un « surplus » $\frac{9}{8}$ fois plus grand qu'un algorithme « parfaitement compétitif » [3].

## Exploration d'un graphe connu ou partiellement connu par plusieurs agents

Certaines tâches d'exploration ne peuvent pas être exécutées par un seul agent, et ce, même avec une carte complète de l'environnement. En fait, Czyżowicz et al. [11] démontrent qu'afin de trouver un nœud malveillant ou hostile dans un arbre, un minimum de deux agents sont nécessaires. Notons qu'un tel nœud détruit tous les agents qui le visitent sans laisser de traces. De plus, les auteurs proposent deux algorithmes, exécutant en temps linéaire en nombre de nœuds, traitant la découverte de nœuds malveillants. Premièrement, un algorithme optimal pour l'exploration d'une ligne est présenté et ensuite un algorithme optimal pour l'exploration d'un arbre dont tous les nœuds ont au moins deux enfants. Finalement, les auteurs élaborent aussi un algorithme approximatif pour un arbre quelconque dont le coût est au plus $\frac{5}{3}$ fois plus grand que l'optimal.

Flocchini et al. [20] poursuivent le travail sur la recherche de nœuds malveillants. Contrairement à [11], les auteurs utilisent un modèle de réseau dynamique où des pannes cycliques sont observables. Plus spécifiquement, [20] fait référence au cas d'un système de train souterrain (ou métro) où certains liens tombent en panne de façon cyclique. De plus, les agents ne peuvent se déplacer librement, mais plutôt à l'aide d'un véhicule tel un train de métro. Les auteurs de l'article [20] proposent une borne minimale d'agents $k = \mathcal{Y}+1$ pour la découverte d'un nœud malveillant, avec $\mathcal{Y}$ étant le nombre d'arrêts (ou de stations de métro) affectés par un nœud malveillant. De

3. Autrement dit, un algorithme ayant le plus petit ratio compétitif possible.

plus, Flocchini et al. donnent une borne supérieure $O(k \cdot n_C^2 \cdot l_R + n_C \cdot l_R^2)$ au nombre de mouvements de train requis afin de trouver les nœuds malveillants. Notons que $n_C$ représente le nombre de trains, et $l_R$, la longueur de la ligne de transport ayant le plus d'arrêts.

Dans d'autres circonstances, il est souhaitable d'avoir plus d'un agent afin de pouvoir effectuer une tâche qui serait trop fastidieuse pour un seul agent. Un exemple d'une telle situation survient avec le *CPP* (problème du postier chinois ou « Chinese Postman Problem »), ou plus précisément, le *k-CPP*. Ce problème repose sur l'idée qu'il est nécessaire d'avoir plusieurs agents afin que la livraison postale soit effectuée dans un délai raisonnable. Formellement, l'algorithme résolvant le *k-CPP* doit trouver $k$ chemins (avec $k > 1$) pour un graphe non-dirigé ayant le même point de départ et d'arrivée. De plus, toutes les arêtes de ce graphe doivent faire partie d'au moins un des $k$ chemins [1]. Tel qu'indiqué dans [1, 22], il est souvent nécessaire de minimiser le chemin à coût maximal des $k$ chemins, ceci étant communément nommé le « minmax » ou « minimax ». De plus, même si des algorithmes polynomiaux pour la solution du *k-CPP* existent, pour $k = 1$, le problème du *k-CPP* demeure NP-difficile [29], compte tenu de la partition du graphe en $k$-chemins.

Afin de pallier à ces problèmes, des auteurs ont développé des algorithmes à coût polynomial basés sur des heuristiques. Plus spécifiquement, Averbakh et Berman [5] présentent un algorithme « minmax » polynomial avec un coût d'au plus $\frac{3}{2}$ fois le coût optimal lorsque les $k$ chemins ont différents points de départ, sinon avec un coût $\frac{3}{4}$ fois le coût optimal s'ils partagent le même. De leur côté, Osterhues et Mariak [29] présentent trois heuristiques pour le *k-CPP* n'utilisant pas le « minmax ». Le premier, un algorithme minimisant la variance du coût entre chaque chemin, le « MAD k-CPP » ou « Minimum Absolute Deviation » ; ensuite une méthode utilisant le carré de la différence du coût entre chaque chemin, le « MSD k-CPP » ou « Minimum Square

Deviation » ; troisièmement, une méthode basée sur la minimisation des chemins excédant une borne de coût, le « MOT k-CPP » ou « Minimum Overtime ». Ahr et Reinelt [1] ont développé un algorithme utilisant la méthode heuristique « tabu ». En résumé, la méthode « tabu » mise sur la sélection aléatoire d'un chemin et la comparaison avec des chemins semblables à ce dernier. De plus, la comparaison utilise des données accumulées durant les comparaisons précédentes.

### Exploration d'un terrain géométrique connu ou partiellement connu par un ou plusieurs robots

On parle de routage de véhicules ou d'individus lorsqu'un ou plusieurs véhicules ou individus doivent effectuer une ou des tournées afin d'atteindre une liste d'objectifs (ordonnés ou non) dans un environnement connu. Les objectifs peuvent représenter des villes, des clients, des routes, etc. Donc, si l'on représente les objectifs à l'aide d'un graphe, le but d'un algorithme de routage est d'atteindre chacun des objectifs dans l'ordre désiré, à coût minimal. Comparativement à un problème d'exploration où tous les éléments d'un environnement doivent être atteints, les problèmes de routage considèrent uniquement un sous-ensemble d'éléments d'un environnement. Donc, si seulement un sous-ensemble de nœuds et d'arêtes sont à visiter, au lieu de visiter l'intégralité de l'environnement, le problème d'exploration optimale de ce « sous-environnement » devient en fait le même que celui du routage avec une tournée optimale. En effet, l'article [28] discute d'un tel lien où la représentation d'un réseau routier par un graphe sert à effectuer l'exploration de toutes les routes d'un secteur avec des drones [4] afin de faire l'inspection visuelle de chacune des routes. Les auteurs de [28] font référence au problème de la recherche dans un réseau routier et discutent de deux façons d'aborder le problème : le « Travelling Salesman Problem (TSP) » ou le « Chinese Postman Problem (CPP) ». Similairemment au « CPP », le « TSP » est

4. Un drone est un véhicule (typiquement un aéronef) commandé à distance.

un problème algorithmique classique où une entité doit visiter tous les nœuds d'un graphe une seule fois tout en revenant au nœud de départ.

Dans un article de 1976, Frederickson et al. [22] présentent des algorithmes approximatifs capables de résoudre le « $k$-CPP » ou le « $k$-TSP » en temps $O(n^3)$ pour un nombre $n$ de nœuds. Les auteurs de [22] décrivent aussi une variante du « $k$-CPP » et du « $k$-TSP », le « $k$-SCP ». Dans le cas du « Stacker Crane Problem (SCP) », une grue doit distribuer du matériel à partir d'un dépôt central à coût optimal. Plusieurs auteurs se sont inspirés des algorithmes présentés dans [22]. Un exemple de ceci provient d'un article de Muslea [27] où l'auteur commente une variété de problèmes de routage de véhicules dans un terrain géométrique représenté par différents types d'arbres (p. ex. binaires ou non, avec ou sans poids). L'auteur présente principalement des heuristiques pour le problème de routage à deux véhicules, mais aussi à $k$ véhicules. Muslea dénote une démarcation entre les problèmes de routage en ligne, hors ligne et très hors ligne. Effectivement, l'auteur décrit les problèmes de routage très hors ligne comme étant des situations où les objectifs sont connus d'avance, ainsi que tous les points à visiter, et où aucun ordre d'exécution des objectifs n'est imposé. Donc, ceci est l'opposé du problème de routage en ligne où les objectifs sont inconnus avant l'exécution de l'algorithme et doivent être atteints dans l'ordre d'arrivée de ceux-ci.

Plus récemment, Arkin et al. [4] et Xu et al. [32] prennent aussi une approche semblable à celle de Frederickson et al. En effet, les auteurs de [4] considèrent une variété de problèmes de routage de véhicules. Ces derniers proposent des algorithmes approximatifs avec un ratio constant ayant pour but de, soit minimiser le nombre $k$ de véhicules, ou la distance maximale $\lambda$ pour un nombre de véhicules $k$. De leur côté, les auteurs de [32] abordent le problème du « minmax » de manière un peu plus restrictive et ajoutent la notion d'entrepôts situés à divers endroits dans le territoire à explorer. Les auteurs démontrent que le problème décrit demeure NP-difficile et que

ce dernier ne peut pas être résolu à l'aide un algorithme d'approximation ayant un ratio inférieur à $\frac{4}{3}$. De plus, [32] propose un algorithme approximatif ayant un ratio constant pouvant résoudre le problème décrit : le « min–max LRP » ou « min-max Location Routing Problem ».

D'autres approches ont aussi fait l'objet de recherche dans le domaine de l'exploration d'un terrain connu avec de multiples agents. L'une de ces approches repose sur un principe très populaire de nos jours, soit la mise à l'enchère. En effet, Lagoudakis et al. [25] proposent qu'un groupe d'agents, ou de robots dans ce cas, puissent utiliser les principes de base des marchés économiques afin de miser sur l'exploration d'un endroit précis sur le territoire à couvrir. Les auteurs démontrent les forces et les faiblesses de plusieurs types de calcul pour la mise de chacun des robots, ainsi que les bornes pour les algorithmes approximatifs associés aux types de mise présentés.

Finalement, l'inspection d'un territoire connu afin de trouver des anomalies dans un terrain hostile est certainement l'une des applications les plus classiques pour un robot. Donc, la mention d'un article allant dans ce sens, mais avec l'utilisation de multiples robots, semble très naturelle étant donné le sujet du présent mémoire. En ce sens, l'article de Williams et Burdick [31] aborde l'inspection d'un territoire planaire avec un certain nombre d'obstacles dont le périmètre doit être exploré. Afin de simplifier le problème, l'algorithme présenté par les auteurs transforme d'abord les périmètres à visiter en les reliant pour former un graphe à l'intérieur du plan. Avec ce graphe, le problème devient maintenant un problème d'exploration par $k$ robots du même type que le *k-CPP*.

# Chapitre 3

# Les bases de l'algorithme d'exploration optimale d'un segment de droite par deux agents

## 3.1 Description de l'environnement

Considérons un segment de droite $L$. Deux agents mobiles, $A_1$ et $A_2$, se déplacent à une vitesse constante sur le segment de droite $L$. Les agents $A_1$ et $A_2$ se trouvent respectivement à une distance de $a$ et $a + d$ du début du segment de droite $L$. De plus, l'agent $A_2$ se trouve à une distance $b$ de la fin du segment $L$. Donc, le segment est de longueur $D = a+b+d$. Finalement, notons les points $u_1$ et $u_2$, respectivement les positions initiales des agents $A_1$ et $A_2$.
Nous désirons établir un algorithme qui découvrira le parcours d'exploration optimal du segment par les deux agents et le coût d'exécution de ce parcours. Chaque point du segment doit être visité par au moins un agent. Pour ce faire, voici quelques définitions qui seront nécessaires pour l'élaboration de l'algorithme et son analyse.

**Définition 1.** *Le* parcours d'un agent *est une fonction continue*

$$f : [0, T] \to [0, D] \tag{3.1}$$

*tel qu'il existe des moments dans le temps*

$$t_0, t_1, t_2, \ldots, t_k \in [0, T] \tag{3.2}$$

*tel que soit* $\forall x \leq |t_{i+1} - t_i|$,

$$f(t_i + x) = \begin{cases} f(t_i) + x \text{ pour } i \text{ pair et} \\ f(t_i) - x \text{ pour } i \text{ impair} \end{cases} \tag{3.3}$$

*soit* $\forall x \leq |t_{i+1} - t_i|$

$$f(t_i + x) = \begin{cases} f(t_i) + x \text{ pour } i \text{ impair et} \\ f(t_i) - x \text{ pour } i \text{ pair} \end{cases} \tag{3.4}$$

*La figure 3.2 illustre la progression de la fonction f pour le cas de l'équation 3.3, tandis que la figure 3.3 illustre la progression pour le cas de l'équation 3.4.*

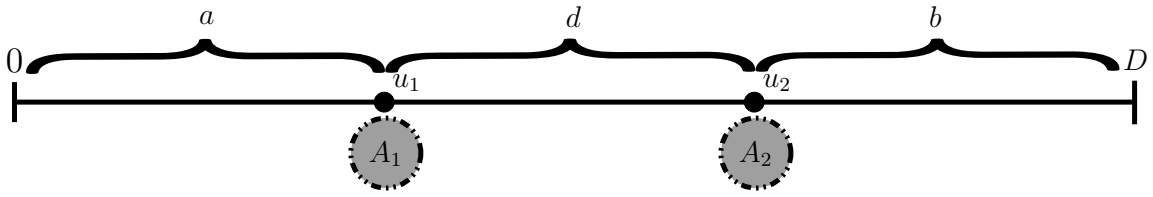


FIGURE 3.1 – Le segment de droite $L$

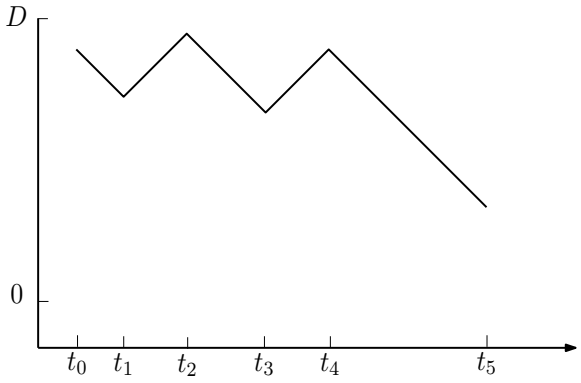


FIGURE 3.2 – Fonction continue $f$ d'un parcours pour le cas de l'équation 3.3

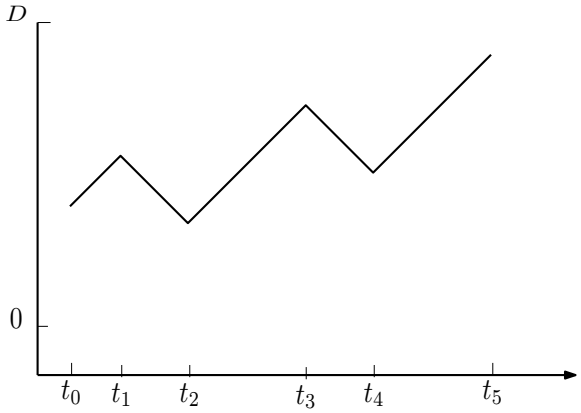


FIGURE 3.3 – Fonction continue $f$ d'un parcours pour le cas de l'équation 3.4

**Définition 2.** *Le* coût du parcours $f$ *est égal à* $t_k$*, soit la durée du parcours. On dit qu'un point* $x \in [0, D]$ *est visité par l'agent effectuant le parcours* $f$ *au moment* $t$*, ssi* $f(t) = x$*.*

**Définition 3.** *Un* parcours d'exploration *est une paire ordonnée de parcours d'agent* $(f_1, f_2)$ *tel que* $f_1(0) \leq f_2(0)$ *et que chaque* $x \in [0, D]$ *est visité par au moins un agent.*

**Définition 4.** *Le* coût du parcours d'exploration $(f_1, f_2)$ *est le maximum entre les coûts des parcours* $f_1$ *et* $f_2$.

**Définition 5.** *Un instant de temps* $c$ *est dit un* moment de croisement *si* $f_1(c) = f_2(c)$ *et* $\exists \epsilon > 0$ *tel que dans l'intervalle* $[c - \epsilon, c + \epsilon]$, *la fonction* $f_1$ *est croissante et* $f_2$ *est décroissante ou* $f_1$ *est décroissante et* $f_2$ *est croissante.*

**Définition 6.** *Un parcours d'exploration* $(f_1, f_2)$ *est dit* sans croisement *ssi* $\forall t \in [t_0, \ldots, t_k], f_1(t) \leq f_2(t)$

Observons qu'un parcours d'exploration sans croisement est un parcours pour lequel il n'existe aucun moment de croisement.

**Définition 7.** *Un parcours d'exploration* $(f_1, f_2)$ *est dit* optimal *s'il n'existe pas un parcours d'exploration* $(f_1', f_2')$ *ayant un coût plus petit.*

## 3.2 Les bases de l'algorithme

Afin de bien cerner l'algorithme, plusieurs lemmes seront nécessaires à la démonstration de la validité de l'algorithme qui trouvera le parcours d'exploration optimal, soit celui ayant le coût le plus faible.

**Lemme 1.** *Pour chaque parcours d'exploration* $(f_1, f_2)$, *il existe un parcours d'exploration* $(f_1', f_2')$ *sans croisement tel que le coût d'exploration de* $(f_1', f_2')$ *n'a pas un coût plus élevé que* $(f_1, f_2)$.

*Démonstration.* Considérons deux agents $A_1$ et $A_2$, tels que $f_1(0) \leq f_2(0)$. Ces deux agents effectuent un parcours d'exploration $(f_1, f_2)$ avec un nombre $j+1$ de moment de croisements. Soit $C = \{c_0, \ldots, c_j\}$ l'ensemble des moments de croisements tel que $c_0 < c_1 < \cdots < c_j$.

On démontre que l'on peut diminuer le nombre de moments de croisement en obtenant un parcours d'exploration au coût égal à $(f_1, f_2)$. Pour ceci, modifions le

parcours de l'agent $A_1$ de sorte que, jusqu'au temps $c_0$, il réalise son parcours, et qu'après le temps $c_0$, il réalise le parcours de l'agent $A_2$. Donc, nous avons un nouveau parcours $f_1^*$ tel que :

$$f_1^* = \begin{cases} f_1(t) \text{ si } t \leq c_0, \\ f_2(t) \text{ si } t > c_0 \end{cases} \tag{3.5}$$

De façon similaire, l'agent $A_2$ réalise son parcours jusqu'au temps $c_0$, et après le temps $c_0$, il réalise le parcours de l'agent $A_1$ avec un nouveau parcours $f_2^*$ tel que :

$$f_2^* = \begin{cases} f_2(t) \text{ si } t \leq c_0, \\ f_1(t) \text{ si } t > c_0 \end{cases} \tag{3.6}$$

Observons que les deux agents réalisent des parcours corrects du segment $L$ et que le parcours d'exploration $(f_1^*, f_2^*)$ est de même coût que $(f_1, f_2)$. En plus, l'ensemble de croisement $C$ est maintenant réduit d'un élément avec le nouveau parcours $(f_1', f_2')$.

Par induction sur le nombre de moments de croisement, on peut éliminer tous les moments de croisement en obtenant le même coût de parcours d'exploration. □

**Lemme 2.** *Il existe un point $Z \in [0, D]$ et il existe un parcours d'exploration optimal $(f_1, f_2)$ tel que les points du segment $L$ visités par l'agent $A_1$ forment le segment $[0, Z]$ et que les points visités par l'agent $A_2$ forment le segment $[Z, D]$.*

*Démonstration.* Prenons un parcours d'exploration optimal sans croisement qui existe selon le lemme 1. Puisque le parcours d'un agent est une fonction continue, l'ensemble des points visités par un agent forme un segment contenant son point de départ. Puisque c'est un parcours d'exploration, les points 0 et $D$ doivent être tous

les deux visités. Puisque c'est un parcours sans croisement, le point 0 doit être visité par l'agent $A_1$ et $D$ doit être visité par l'agent $A_2$. La réunion des segments parcourus par $A_1$ et $A_2$ doit couvrir tous les points du segment $L$. Donc, le segment parcouru par $A_1$ doit être de la forme $[0, F_1]$ et par $A_2$ de la forme $[F_2, D]$ avec $F_1 \geq F_2$. Supposons que $F_1 > F_2$. Soit $Z \in [u_1, u_2] \cap [F_1, F_2]$ un point sur le segment $L$. On modifie le parcours de l'agent $A_1$ la de façon suivante : soit $0 \leq s_1 < s'_1 < s_2 < s'_2 < \cdots < s_p < s'_p \leq T$ des points de temps tel que l'agent $A_1$ visite des points plus grands que $Z$ dans les intervalles de temps $]s_1, s'_1[$, $]s_2, s'_2[$, ..., $]s_p, s'_p[$. Le parcours modifié $f'_1$ de $A_1$ est défini comme suit pour $s'_i \leq t \leq s_{i+1}$ :

$$f'_1(t - \sum_{j \leq i} \big(s'_j - s_j\big)) = f_1(t) \tag{3.7}$$

Le nouveau parcours d'exploration $(f'_1, f_2)$ est toujours sans croisement et est tel que l'agent $A_1$ visite les points de l'intervalle $[0, Z]$ et l'agent $A_2$ visite les points de l'intervalle $[F_2, D]$. Le nouveau parcours d'exploration est optimal car son coût n'est pas plus grand que celui de $(f_1, f_2)$.

De façon similaire, soit $0 \leq r_1 < r'_1 < r_2 < r'_2 < \cdots < r_p < r'_p \leq T$ des points de temps tel que l'agent $A_2$ visite des points plus petits que $Z$ dans les intervalles de temps $]r_1, r'_1[$, $]r_2, r'_2[$, ..., $]r_p, r'_p[$. Le parcours modifié $f'_2$ de $A_2$ est défini comme suit pour $r'_i \leq t \leq r_{i+1}$ :

$$f'_2(t - \sum_{j \leq i} \big(r'_j - r_j\big)) = f_2(t) \tag{3.8}$$

Le couple $(f'_1, f'_2)$ est un parcours d'exploration sans croisement tel que l'agent $A_1$ visite l'intervalle $[0, Z]$ et l'agent $A_2$ visite l'intervalle $[Z, D]$. Ce parcours d'exploration est optimal car son coût n'est pas plus grand que celui du parcours $(f'_1, f_2)$ □

**Définition 8.** *Un parcours d'exploration* $(f_1, f_2)$ *est dit* régulier *s'il satisfait les conditions du lemme 2.*

**Lemme 3.** *Le parcours optimal d'un segment par un agent consiste à se diriger tout droit vers l'extrémité la plus proche de l'agent et ensuite parcourir entièrement le segment en terminant le parcours à l'autre extrémité du segment.*

*Démonstration.* Prenons un segment $S$ de longueur $s$. Soit $u$, la position initiale de l'agent à une distance $\alpha$ d'une des extrémités et $\beta$ la distance de l'autre extrémité. Donc, $\alpha + \beta = s$. Supposons, sans perte de généralité, que $\alpha \leq \beta$. Il existe un algorithme de parcours avec coût $\alpha+s$ qui consiste à aller directement vers l'extrémité la plus proche et retourner directement vers l'autre. Il reste à prouver que c'est un algorithme optimal. Considérons n'importe quel algorithme optimal$\mathcal{A}$ de parcours de segment. Cet algorithme doit d'abord visiter une extrémité et ensuite l'autre. Le temps jusqu'à la visite de la première extrémité est au moins $\alpha$ et le temps entre la visite de la première extrémité et la seconde extrémité est au moins $s$. Donc, le temps total de l'algorithme optimal $\mathcal{A}$ est au moins $\alpha + s$, ce qui montre que notre algorithme est optimal car le temps de notre algorithme est $\alpha + s$.

□

Conséquemment au lemme 3, la figure 3.4 illustre les deux possibilités afin de parcourir un segment par un seul agent de façon optimale. Notons que l'agent peut se trouver initialement à une extrémité d'un segment (la distance de l'extrémité la plus proche est nulle).

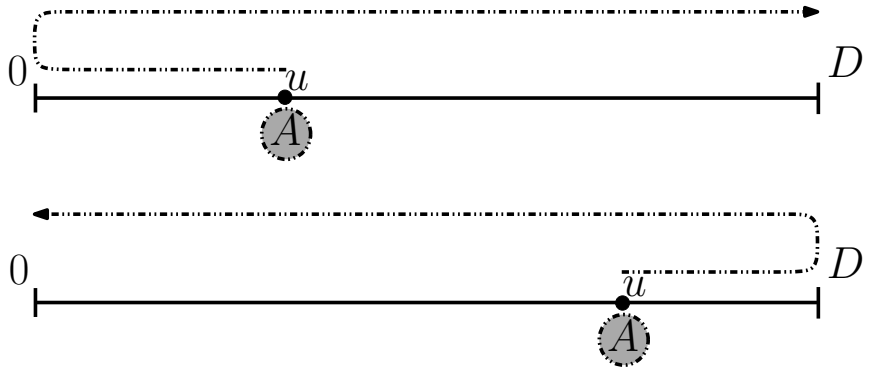


FIGURE 3.4 – Les deux parcours optimaux d'exploration possibles par un agent.

Observons maintenant que, en connaissant le point $Z$ du lemme 2, nous pouvons identifier un nombre maximal de quatre possibilités pour parcourir un segment avec deux agents de façon optimale et que nous pouvons exprimer le temps de ce parcours à l'aide des paramètres $a, b$, et $d$ du poblème. Ceci est attribuable au fait que nous avons deux possibilités de parcours optimal d'un segment par un agent et que nous avons deux segments à parcourir, donc quatre combinaisons sont possibles (observons les différentes combinaisons dans la figure 3.5).

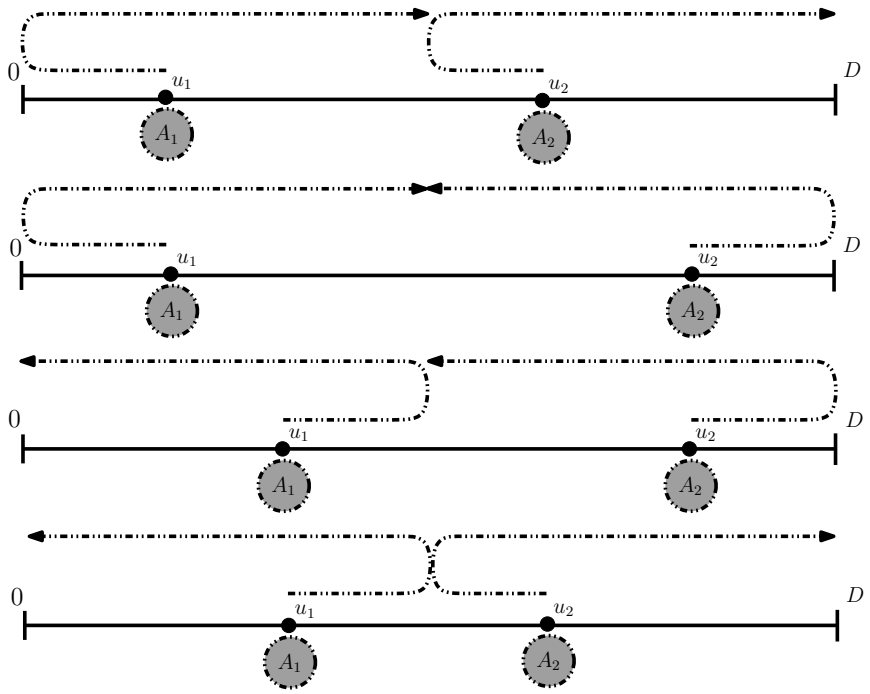


FIGURE 3.5 – Les quatre parcours optimaux d'exploration possibles par une paire d'agents.

Les lemmes précédents nous ont indiqué qu'il existe un parcours d'exploration optimal pour un segment $L$ tel que les deux segments visités par les agents s'intersectent en un point $Z$. Avec ces lemmes, nous pourrons continuer à établir les bases de l'algorithme pour trouver le temps et le parcours d'exploration optimal.

**Définition 9.** *Considérons un agent à une distance $x$ d'une extrémité du segment et à une distance $y$ de l'autre. Alors $OPT(x, y)$ est une procédure qui fait le parcours optimal du segment pour cet agent et $opt(x, y)$ est la distance parcourue par l'agent qui exécute la procédure $OPT(x, y)$.*

Observons que le lemme 3 implique la formule suivante :

$$opt(x, y) = x + y + \min(x, y) \tag{3.9}$$

Soit $T_m$ le temps du parcours d'exploration optimal du segment $[0, D]$ par deux agents.

Selon les valeurs de paramètres $a, b$, et $d$, le calcul du temps $T_m$ tombera dans une des deux catégories de cas à traiter : les cas spéciaux et les cas généraux.

## Cas spéciaux

Nous parlons d'un cas spécial lorsque la position initiale d'un des deux agents ou des deux agents coïncide avec le point $Z$ de division de segment vu au lemme 2. Nous traitons ces situations comme des cas spéciaux car le temps d'exploration optimal par deux agents sera alors, soit $a$, soit $b$, et donc le parcours d'un des deux agents est sans conséquence sur le coût total d'exploration.

### Cas spécial 1

Lorsque $a \geq b + d + min(d, b)$, la valeur $a$ domine le temps du parcours d'exploration optimal. Cela implique que la position initiale de l'agent $A_1$ se trouvera au point de division du segment. Il est clair que ceci est vrai car aucun parcours de l'agent $A_1$ ne pourrait être moins coûteux que le parcours de l'agent $A_2$. L'agent $A_1$ doit parcourir la distance $a$ en utilisant le parcours $OPT(a, 0)$ et l'agent $A_2$ doit parcourir la distance $b + d + min(d, b)$ en utilisant le parcours $OPT(d, b)$.

Dans ce cas :

$$T_m = a \tag{3.10}$$

### Cas spécial 2

De façon similaire, lorsque $b \geq a + d + min(a, d)$, la valeur $b$ domine le temps du parcours d'exploration optimal. Donc, la position initiale de l'agent $A_2$ se trouvera

au point de division du segment et aucun parcours de l'agent $A_2$ pourrait être moins coûteux que le parcours de l'agent $A_1$. L'agent $A_2$ doit parcourir la distance $b$ en utilisant le parcours $OPT(0, b)$ et l'agent $A_1$ doit parcourir la distance $a+d+min(a, d)$ en utilisant le parcours $OPT(a, d)$.
Dans ce cas :

$$T_m = b \tag{3.11}$$

## Cas généraux

Soit $(f_1, f_2)$ un parcours d'exploration régulier, soit $Z$ un point de division, tel que mentionné au lemme 2, et soit $z$ la distance entre $u_1$ et $Z$.

Les cas généraux se caractérisent par le fait que le point $Z$ du segment $L$ en deux segments explorés individuellement par chaque agent n'est occupé initialement par aucun agent. Donc, nous avons $a < b + d + min(b, d)$ et $b < a + d + min(a, d)$. L'exploration de chacun de ces segments par l'agent correspondant résulte en un parcours de même coût et, conséquemment, au coût optimal du parcours.

**Lemme 4.** *Considérons un parcours d'exploration régulier $(f_1, f_2)$. Si $a < b + d + min(b, d)$ et $b < a + d + min(a, d)$, alors il existe une valeur unique $0 < z < d$ tel que $opt(a, z) = opt(d - z, b)$ et $f_1 = OPT(a, z)$ et $f_2 = OPT(d - z, b)$.*

*Démonstration.* Considérons un parcours d'exploration régulier $(f_1, f_2)$ tel que $f_1 = OPT(a, z)$ et $f_2 = OPT(d-z, b)$. Donc, selon la définition de OPT (définition 9), les distances parcourues par chacun des agents $A_1$ et $A_2$ sont respectivement régies par les fonctions $\psi(z) = opt(a, z)$ et $\phi(z) = opt(d - z, b)$. Observons que les fonctions $\psi$ et $\phi$ sont des fonctions continues et dénotons $\Delta(z) = \phi(z) - \psi(z)$. Nous avons $\psi(0) = opt(a, 0) = a < b + d + min(b, d) = opt(d, b) = \phi(0)$. Donc, $\Delta(0) = \phi(0) - \psi(0) > 0$. De même, nous avons $\psi(d) = opt(a, d) = a + d + min(a, d) > b = opt(0, b) = \phi(d)$. Donc, $\Delta(d) = \phi(d) - \psi(d)$. En vue de la continuité de la fonction $\Delta : [0, d] \to \mathbb{R}$,

il existe un $z \in (0, d)$, tel que $\Delta(z) = 0$. Donc, pour ce $z$, on a $opt(a, z) = \psi(z) = \phi(z) = opt(d - z, b)$.

Il reste à démontrer que la valeur $z$ est unique. Supposons que non. Donc, il existe $z_1 < z_2$ t.q. $\phi(z_1) = \psi(z_1)$ et $\phi(z_2) = \psi(z_2)$. Observons que $\phi$ est strictement croissant et $\phi$ est strictement décroissant dans $(0, d)$.

Donc $\phi(z_2) > \phi(z_1) = \psi(z_1) > \phi(z_2)$. Alors, nous avons une contradiction.

□

# Chapitre 4

# L'algorithme

## 4.1 Formulation de l'algorithme pour trouver le meilleur parcours d'exploration

À l'aide des sections précédentes, il nous est possible d'élaborer l'algorithme de création d'un parcours d'exploration régulier. L'algorithme 1 (dans l'encadré ci-dessous) prend en entrée les valeurs $a, b$ et $d$. Lors de l'exécution, l'algorithme produit un parcours d'exploration régulier $(f_1, f_2)$ ainsi que le temps d'exécution $T$ pour ce dernier. L'algorithme 1 comprend deux phases. Dans un premier temps, l'algorithme établit la valeur de $z$ et le temps $T$. Deuxièmement, l'algorithme établit les instructions à communiquer aux agents $A_1$ et $A_2$.

Observons tout d'abord que la deuxième phase de l'algorithme (les lignes 10 et 11) donne les instructions aux agents $A_1$ et $A_2$ en utilisant la valeur de $z$ trouvée, soit à la ligne 1, 4, ou 8.

Considérons maintenant la première phase de l'algorithme. Cette dernière est divisée en trois parties correspondant aux trois cas à considérer vus dans la section

précédente : le cas spécial 1, le cas spécial 2 et les cas généraux. Observons que la ligne 1 correspond exactement à la définition du cas spécial 1 et que la ligne 4 correspond à la définition du cas spécial 2. Les cas généraux, quant à eux, sont traités sous la condition *Else* à la ligne 7. Sous cette clause *Else*, on voit un appel à la fonction *BAL* (la ligne 8). La fonction BAL (dans l'encadré subséquent) permet de trouver un $z$ associé à un temps $T$ minimal. En somme, la fonction BAL calcule d'abord les temps $T[i]$ pour les quatre cas généraux. Ensuite, la fonction trouve le temps $T[i]$ le plus petit ainsi que la valeur de $z$ associée à ce temps $T[i]$ minimal. Finalement, la fonction retourne le temps $T$ minimal ainsi que la valeur $z$ associée.

**entrée** : entier $a$, $b$, $d$
**sortie** : entier $T$, $f_1$ le parcours pour $A_1$, $f_2$ le parcours pour $A_2$

1 **if** $a \geq b + d + \min(b, d)$ **then**
2 | $z = 0$;
3 | $T = a$;
4 **if** $b \geq a + d + \min(a, d)$ **then**
5 | $z = d$;
6 | $T = b$;
7 **else** *{Trouver un z qui équilibrera le coût de chacun des parcours.}*
8 | $<z, T>$ = BAL($a$, $b$, $d$);
9 **end**
10 $A_1$ = `OPT` ($a$,$z$);
11 $A_2$ = `OPT` ($d - z$,$b$);

**Algorithme 1:** Production d'un parcours d'exploration régulier avec les valeurs $a, b$, et $d$

**entrée** : entier $a$, $b$, $d$
**sortie** : valeur de $z$ et le temps $T$

1 $T[0] = \frac{2d+b+a}{2}$;
2 $T[1] = \frac{4b+2d+a}{3}$;
3 $T[2] = \frac{4a+2d+b}{3}$;
4 $T[3] = \frac{2a+2b+d}{2}$;
5 $i = \min\{j : 0 \leq j \leq 3, \forall k \leq 3\ T[j] \leq T[k]\}$;
6 $T = T[i]$;
7 **switch** $T[i]$ **do**
8 | **case** $T[0]$ *{Cas général 1}*
9 | | $z = \frac{2d-a+b}{4}$
10 | **case** $T[1]$ *{Cas général 2}*
11 | | $z = \frac{2b+d-a}{3}$
12 | **case** $T[2]$ *{Cas général 3}*
13 | | $z = \frac{2d+b-2a}{3}$
14 | **case** $T[3]$ *{Cas général 4}*
15 | | $z = \frac{2b+d-2a}{2}$
16 | BAL $= <z, T>$;
17 **endsw**

**Fonction** BAL($a$, $b$, $d$)

## 4.2 Preuve d'exactitude et d'optimalité du résultat de l'algorithme

Afin de démontrer la validité de l'algorithme 1 et de la fonction BAL décrites ci-haut, nous devons d'abord démontrer les lemmes 5 et 6 qui visent respectivement le cas spécial 1 et le cas spécial 2.

**Lemme 5.** *Considérons un parcours d'exploration régulier $P = (f_1, f_2)$ avec $Z = u_1$ et $d > 0$. Alors $a \geq opt(d, b)$.*

*Démonstration.* Supposons que non, donc $a < opt(d, b) = x$. Soit $\epsilon = min(\frac{d}{2}, \frac{x-a}{4})$. Considérons le parcours suivant : l'agent $A_1$ exécute $OPT(a, \epsilon)$ et l'agent $A_2$ exécute le parcours $OPT(d - \epsilon, b)$.

On a :

Fait 1. $opt(a, \epsilon) \leq a + 2\epsilon$, car il y a un parcours avec le temps $a + 2\epsilon$.

Fait 2. $opt(d - \epsilon, b) \leq opt(d, b) - \epsilon$.

*Démonstration du fait 2*

Cas 1. $opt(d - \epsilon, b) = 2b + d - \epsilon$

alors $d - \epsilon \geq b$ donc $d \geq b$. Donc $opt(d, b) = 2b + d$ et $opt(d - \epsilon, b) = 2b + d - \epsilon \leq opt(d, b) - \epsilon$.

Cas 2. $opt(d - \epsilon, b) = 2(d - \epsilon) + b$, alors $d - \epsilon \leq b$.

Cas 2.1 $b \geq d$

alors $opt(d, b) = 2d + b$ et $opt(d - \epsilon, b) = 2(d - \epsilon) + b \leq 2d + b - \epsilon = opt(d, b) - \epsilon$.

Cas 2.2 $d - \epsilon \leq b < d$

alors $opt(d, b) = 2b + d$ et $opt(d - \epsilon, b) = 2(d - \epsilon) + b \leq 2b + d - \epsilon = opt(d, b) - \epsilon$.

*Fin démonstration du fait 2.*

Donc, le temps de ce parcours d'exploration est au plus $max(a + 2\epsilon, x - \epsilon) \leq max(\frac{x+a}{2}, x - \epsilon)$. Puisque $a < x$, on a $\frac{x+a}{2} < x$. Donc, $max(\frac{x+a}{2}, x - \epsilon) < x$.

Nous avons donc trouvé un parcours d'exploration avec un temps $< x$, ce qui contredit l'optimalité du parcours d'exploration régulier $P$ car le temps de celui-ci est $max(a, x) = x$.

□

**Lemme 6.** *Considérons un parcours d'exploration régulier $P = (f_1, f_2)$ avec $Z = u_2$ et $d > 0$. Alors $b \geq opt(a, d)$.*

*Démonstration.* Observons que ce lemme est symétrique au lemme 5 et donc que la preuve est symétrique aussi. □

Avec les lemmes 5 et 6, nous pourrons démontrer la validité de l'algorithme 1 et l'optimalité du parcours d'exploration produit par ce dernier.

**Théorème 1.** *L'algorithme 1 produit un parcours d'exploration régulier correct d'un segment par deux agents avec les paramètres $a, b$ et $d$.*

*Démonstration.* Observons que, dans l'algorithme 1, il existe toujours un point $Z$ tel que $A_1$ parcourt le segment $0Z$ et que $A_2$ parcourt le segmet $ZD$. Donc, en vertu du lemme 2, le parcours produit est correct.

La partie restante de la démonstration du théorème est consacrée à la démonstration de l'optimalité du parcours d'exploration produit par l'algorithme. Dans un premier temps, divisons la démonstration de l'optimalité du résultat en deux parties : les cas où $d = 0$ et les cas où $d > 0$.

Tout d'abord, observons que si $d = 0$, alors $Z = u_1 = u_2$. Donc, en vertu du fait que $(f_1, f_2)$ est un parcours d'exploration régulier, le coût du parcours est $max(a, b)$.

L'algorithme 1 produit un parcours ayant un coût $T = a$ (voir ligne 3 de l'algorithme 1), si $d = 0$ et $a \geq b$. De plus, à la ligne 6 de l'algorithme, un parcours ayant un coût $T = b$ est produit si $d = 0$ et $b \geq a$.

Donc, l'algorithme 1 produit toujours un parcours d'exploration optimal si $d = 0$. Considérons maintenant les cas où $d > 0$.

Prenons un parcours d'exploration optimal $(f_1, f_2)$. D'après le lemme 2, il existe un point $Z$ tel que l'agent $A_1$ parcourt le segment $[0, Z]$ et l'agent $A_2$ parcourt le segment $[Z, D]$. Remarquons que $Z \in [u_1, u_2]$.

*Cas 1 ($Z = u_1$) ou cas spécial 1*

Observons que, d'après le lemme 5, $a \geq opt(d, b)$. Donc, le temps du parcours d'exploration $(f_1, f_2)$ est $a$. Observons que, dans ce cas, le temps de l'algorithme 1 tel que vu à la ligne 3 de l'algorithme est $a$.

*Cas 2 ($Z = u_2$) ou cas special 2*

Observons que, d'après le lemme 6, $b \geq opt(a, d)$. Donc, le temps du parcours d'exploration $(f_1, f_2)$ est $b$. Observons que, dans ce cas, le temps de l'algorithme 1, tel que vu à la ligne 6 de l'algorithme, est $b$.

*Cas 3 ($u_1 < Z < u_2$) ou cas général*

Observons que, dans ce cas, $0 < z < d$. Donc, d'après le lemme 4 : $opt(a, z) = opt(d - z, b)$. Donc, le temps du parcours d'exploration $(f_1, f_2)$ est $\alpha = opt(a, z) = opt(d - z, b)$.

Puisque la valeur de $z$ est unique, il reste à montrer que le temps d'exécution de l'algorithme 1 est aussi $\alpha$.

*Cas 3.1 ($z < a$ et $(d-z) < b$)*
Dans ce cas, $opt(a,z) = 2z+a$ et $opt(d-z,b) = 2(d-z)+b$. Donc, si nous avons un $z$ qui est la solution de $2z+a = 2(d-z)+b$ , alors $T = opt(a,z) = opt(d-z,b) = \alpha$. Trouvons d'abord la valeur de $z$ pour ce cas :

$$\begin{aligned} opt(a,z) &= 2z+a \\ opt(d-z,b) &= 2(d-z)+b \end{aligned} \tag{4.1}$$

$$\begin{aligned} opt(a,z) &= opt(d-z,b) \\ 2z+a &= 2d-2z+b \\ 4z &= 2d-a+b \\ z &= \frac{2d-a+b}{2} \end{aligned} \tag{4.2}$$

Donc, $z = \frac{2d-a+b}{2}$. Observons maintenant la valeur de $T$ dans ce cas :

$$\begin{aligned} T &= opt(a,z) = opt(d-z,b) \\ T &= 2z+a \\ T &= 2(\frac{2d-a+b}{4})+a \\ T &= \frac{4d-2a+2b}{4}+a \\ T &= \frac{4d-2a+4a+2b}{4} \\ T &= \frac{4d+2a+2b}{4} \\ T &= \frac{2d+a+b}{2} \end{aligned} \tag{4.3}$$

Donc, dans ce cas, $T = \alpha = \frac{2d+a+b}{2}$.

*Cas 3.2 ($z < a$ et $(d-z) \geq b$)*
Dans ce cas, $opt(a,z) = 2z+a$ et $opt(d-z,b) = 2b+(d-z)$. Donc, si nous avons un $z$ qui est la solution de $2z+a = 2b+(d-z)$ , alors $T = opt(a,z) = opt(d-z,b) = \alpha$. Trouvons d'abord la valeur de $z$ pour ce cas :

$$\begin{aligned} opt(a,z) &= 2z+a \\ opt(d-z,b) &= 2b+d-z \end{aligned} \tag{4.4}$$

$$\begin{aligned} opt(a,z) &= opt(d-z,b) \\ 2z+a &= 2b+d-z \\ 3z &= 2b+d-a \\ z &= \frac{2b+d-a}{3} \end{aligned} \tag{4.5}$$

Donc, $z = \frac{2b+d-a}{3}$. Observons maintenant la valeur de $T$ dans ce cas :

$$\begin{aligned} T &= opt(d-z,b) \\ T &= 2b+d-z \\ T &= 2b+d-\frac{2b+d-a}{3} \\ T &= \frac{6b-2b+3d-d+a}{3} \\ T &= \frac{4b+2d+a}{3} \end{aligned} \tag{4.6}$$

Donc, dans ce cas, $T = \alpha = \frac{4b+2d+a}{3}$.

*Cas 3.3 ($z \geq a$ et $(d-z) < b$)*
Dans ce cas, $opt(a, z) = 2a + z$ et $opt(d-z, b) = 2(d-z) + b$. Donc, si nous avons un $z$ qui est la solution pouvant résoudre $2a + z = 2(d-z) + b$ , alors $T = opt(a, z) = opt(d-z, b) = \alpha$. Trouvons d'abord la valeur de $z$ pour ce cas :

$$\begin{aligned} opt(a, z) &= 2a + z \\ opt(d-z, b) &= 2(d-z) + b \end{aligned} \tag{4.7}$$

$$\begin{aligned} opt(a, z) &= opt(d-z, b) \\ 2a + z &= 2(d-z) + b \\ 2a + z &= 2d - 2z + b \\ 3z &= 2d + b - 2a \\ z &= \frac{2d + b - 2a}{3} \end{aligned} \tag{4.8}$$

Donc, $z = \frac{2d+b-2a}{3}$. Observons maintenant la valeur de $T$ dans ce cas :

$$\begin{aligned} T &= opt(a, z) \\ T &= 2a + z \\ T &= 2a + \frac{2d + b - 2a}{3} \\ T &= \frac{6a + 2d + b - 2a}{3} \\ T &= \frac{4a + 2d + b}{3} \end{aligned} \tag{4.9}$$

Donc, dans ce cas, $T = \alpha = \frac{4a+2d+b}{3}$.

*Cas 3.4 ($z \geq a$ et $(d-z) \geq b$)*
Dans ce cas, $opt(a,z) = 2a+z$ et $opt(d-z,b) = 2b+(d-z)$. Donc, si nous avons un $z$ qui est la solution de $2a+z = 2b+(d-z)$ , alors $T = opt(a,z) = opt(d-z,b) = \alpha$. Trouvons d'abord la valeur de $z$ pour ce cas :

$$\begin{aligned} opt(a,z) &= 2a+z \\ opt(d-z,b) &= 2b+d-z \end{aligned} \tag{4.10}$$

$$\begin{aligned} opt(a,z) &= opt(d-z,b) \\ 2a+z &= 2b+d-z \\ 2z &= 2b+d-2a \\ z &= \frac{2b+d-2a}{2} \end{aligned} \tag{4.11}$$

Donc, $z = \frac{2b+d-2a}{2}$. Observons maintenant la valeur de $T$ dans ce cas :

$$\begin{aligned} T &= opt(a,z) \\ T &= 2a+z \\ T &= 2a+\frac{2b+d-2a}{2} \\ T &= \frac{4a+2b+d-2a}{2} \\ T &= \frac{2a+2b+d}{2} \end{aligned} \tag{4.12}$$

Donc, dans ce cas, $T = \alpha = \frac{2a+2b+d}{2}$.

Finalement, nous voyons que, pour tous les cas, l'algorithme 1 obtient un temps $T = \alpha$ et est donc optimal.

□

# Chapitre 5

# Conclusion et problèmes futurs

Les segments de droite sont des éléments de base pour une panoplie de formes plus complexes (géométriques ou non). Donc, il est naturel de penser que la solution présentée dans ce mémoire pourrait être appliquée d'une manière ou d'une autre à des problèmes d'exploration d'autres environnements. Par exemple, il est clair que notre solution s'applique aussi pour deux agents situés sur une courbe quelconque finie sans intersections dont les extrémités sont différentes (voir figure 5.1). Par contre, la solution présentée pour un segment de droite connu ne peut pas facilement être transposée vers l'exploration d'un arbre quelconque connu car on peut réduire un problème NP-complet à l'exploration optimale d'un arbre par deux agents.

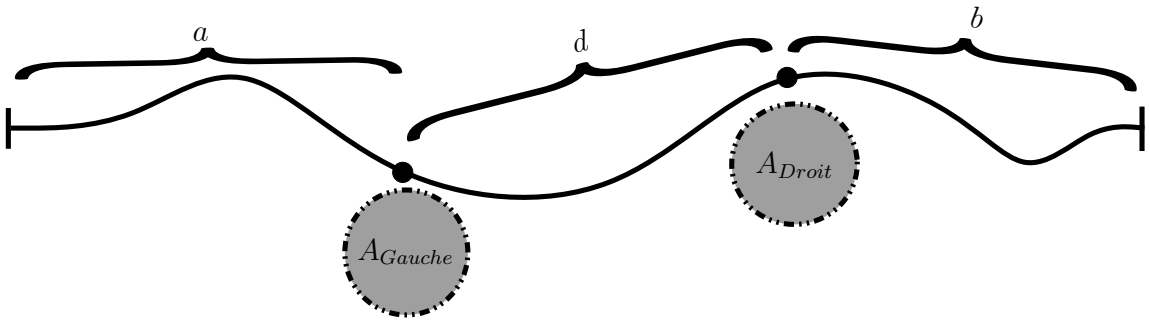


FIGURE 5.1 – Deux agents sur une courbe.

## 5.1 Le problème d'exploration d'un arbre par deux agents mobiles est un problème NP-complet

Observons maintenant un cas plus complexe : celui d'un arbre quelconque. Afin de démontrer que l'exploration optimale d'un arbre quelconque est un problème NP-complet, nous allons vous présenter la réduction du *problème de partition* d'un ensemble d'entiers (« Partition Problem ») à celui de l'exploration optimale d'un arbre quelconque :

**Définition 10.** *Le problème d'exploration optimale de l'arbre par deux agents se formule ainsi :*

*Données : arbre $T$ avec poids entiers positifs d'arêtes (qui indiquent le temps de parcours de l'arête par un agent), sommet $v$, nombre entier positif $k$.*

*Question : Est-ce que deux agents qui partent du sommet $v$ peuvent explorer l'arbre $T$ en temps $k$ ?*

**Définition 11.** *Le* problème de partition *se définit comme suit :*

*Considérons un ensemble de $n$ nombres entiers positifs $A = \{x_1, x_2, \ldots, x_n\}$, tel que $\sum_{i=1}^{n} x_i = S$. Trouver s'il est possible d'effectuer la partition de l'ensemble $A$ en deux sous-ensembles $B$ et $C$ tel que $\sum_{x_i \in B} x_i = \sum_{x_i \in C} x_i = \frac{S}{2}$. Le problème de la partition d'un ensemble d'entiers est NP-complet.*

**Théorème 2.** *L'exploration d'un arbre par deux agents mobiles est un problème NP-complet.*

*Démonstration.* Nous désirons effectuer l'exploration optimale d'un arbre avec deux agents. Prenons un arbre $T$, tel qu'illustré à la figure 5.2, composé de $n$ arêtes ayant chacune un coût d'exploration $x$ tel que représenté par l'ensemble $A = \{x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}\}$. Supposons que $x_{n+1}$ et $x_{n+2}$ sont de longueur $2S$, et que le point ou intersecte toutes les arêtes est un sommet $v$. Dans la solution optimale, chaque agent doit parcourir un ensemble disjoint d'arêtes. Toutes ces arêtes, sauf la dernière, doivent être traversées dans les deux directions. Il est clair que la dernière arête parcourue dans une seule direction doit être $x_{n+1}$ pour un agent et $x_{n+2}$ pour l'autre. Supposons que $A_1$ et $A_2$ sont des sous-ensembles de $A$ parcouru par les deux agents. La somme des coûts de l'exploration pour les deux agents est alors égal à :

$$2(max\left(\sum_{x_i \in A_1} x_i, \sum_{x_i \in A_2} x_i\right) + S) \tag{5.1}$$

La réponse au problème d'exploration optimale de l'arbre pour l'arbre construit, pour le sommet $v =$ et pour $k = 3S$ est « oui », si et seulement si la réponse au *problème de partition* pour $A = \{x_1, x_2, \ldots, x_n\}$ est « oui ». Cette réduction prouve que le problème d'exploration optimale de l'arbre par deux agents est NP-complet.

□

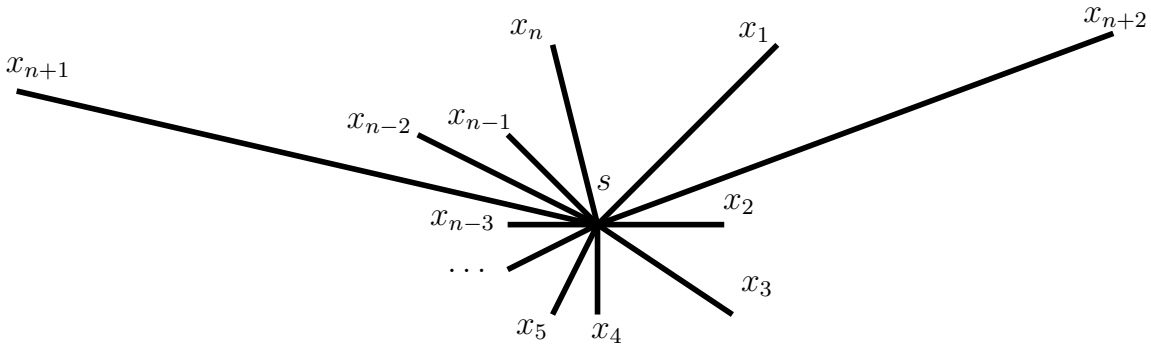


FIGURE 5.2 – L'arbre construit pour la preuve de NP-complétude du problème d' exploration optimale de l'arbre par deux agents.

## 5.2 Problèmes ouverts

Une panoplie de variations découlent du sujet du présent mémoire. En effet, il est possible d'ajouter un facteur de complexité de plusieurs manières. Le premier problème ouvert est de transposer le problème d'exploration à plus de deux agents sur un segment.

Une extension naturelle concerne l'exploration optimale par des agents sur un segment lorsque les agents ne possèdent pas d'information sur la longueur du segment ni sur leurs positions respectives.

Une autre extension possible concerne les agents ayant une visibilité non-nulle. Dans ce cas, si un agent $a_i$ a un rayon de visibilité $r_i$, un point $x$ est considéré exploré si l'agent $a_i$ se trouve à une distance au plus $r_i$ de $x$. Les agents peuvent avoir des rayons de visibilité égaux ou différents.

Il est aussi possible de considérer les agents qui ont une capacité de mobilité variée. Par exemple, il est intéressant de considérer des agents pouvant se déplacer à des vitesses différentes.

Comme le montre l'exemple de la section précédente, il semble moins prometteur d'étendre la considération aux environnements plus généraux. Cependant, la considération de certains environnements particuliers (par ex. anneaux, grilles, polygones en deux dimensions, etc.) peut s'avérer intéressante.

# Bibliographie


[1] Ahr, D., and Reinelt, G. A tabu search algorithm for the mini-max k-chinese postman problem. *Computers & Operations Research 33* (2006), pp. 3403–3422.

[2] Albers, S., and Henzinger, M. R. Exploring unknown environments. *SIAM Journal on Computing 29* (2000), pp. 1164–1188.

[3] Ambuehl, C., Gąsieniec, L., Pelc, A., Radzik, T., and Zhang, X. Tree exploration with logarithmic memory. *ACM Transactions on Algorithms 7* (2011), Article 17.

[4] Arkin, E., Hassin, R., and Levin, A. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms 59* (2006), pp. 1–18.

[5] Averbakh, I., and Berman, O. A heuristic with worst-case analysis for minimax routing of two travelling salesmen on a tree. *Discrete Applied Mathematics 68* (1996), pp. 17–32.

[6] Awerbuch, B., Betke, M., Rivest, R. L., and Singh, M. Piecemeal graph exploration by a mobile robot (extended abstract). In *Proceedings of the Eighth Annual Conference on Computational Learning Theory* (1995), pp. 321–328.

[7] Bender, M. A., Fernández, A., Ron, D., Sahai, A., and Vadhan, S. The power of a pebble : exploring and mapping directed graphs. In *Proceedings of the Thirtieth Annual Symposium on Theory of Computing* (1998), pp. 269–278.

[8] Bender, M. A., and Slonim, D. K. The power of team exploration : two robots can learn unlabeled directed graphs. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 75–85.

[9] Berman, P., Blum, A., Fiat, A., Karloff, H., Rosén, A., and Saks, M. Randomized robot navigation algorithms. In *Proceedings of the Seventh Annual Symposium on Discrete Algorithms* (1996), pp. 75–84.

[10] Czyżowicz, J., Dobrev, S., Gąsieniec, L., Ilcinkas, D., Jansson, J., Klasing, R., Lignos, I., Martin, R., Sadakane, K., and Sung, W.-K. More efficient periodic traversal in anonymous undirected graphs. *Structural Information and Communication Complexity, Lecture Notes in Computer Science 5869* (2010), pp. 167–181.

[11] Czyżowicz, J., Kowalski, D., Markou, E., and Pelc, A. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing 16* (2007), pp. 595–619.

[12] Deng, X., Kameda, T., and Papadimitriou, C. How to learn an unknown environment. $i$ : the rectilinear case. *Journal of the ACM 45* (1998), pp. 215–245.

[13] Deng, X., and Papadimitriou, C. H. Exploring an unknown graph. *Journal of Graph Theory 32*, 3 (1999), pp. 265–297.

[14] Dessmark, A., and Pelc, A. Optimal graph exploration without good maps. *Theoretical Computer Science 326* (2004), pp. 343–362.

[15] Diks, K., Fraigniaud, P., Kranakis, E., and Pelc, A. Tree exploration with little memory. *Journal of Algorithms 51* (2004), pp. 38–63.

[16] Dumitrescu, A., Mitchell, J., and Żyliński, P. Watchman routes for lines and segments. *Algorithm Theory, Lecture Notes in Computer Science – SWAT 2012 7357* (2012), pp. 36–47.

[17] Duncan, C. A., Kobourov, S. G., and Kumar, V. S. A. Optimal constrained graph exploration. *ACM Trans. Algorithms 2* (2006), pp. 380–402.

[18] Fleischer, R., and Trippen, G. Exploring an unknown graph efficiently. In *Proceedings of the 13th Annual European Symposium on Algorithms* (2005), pp. 11–22.

[19] Flocchini, P., Ilcinkas, D., Pelc, A., and Santoro, N. How many oblivious robots can explore a line. *Information Processing Letters 111* (2011), pp. 1027–1031.

[20] Flocchini, P., Kellett, M., Mason, P., and Santoro, N. Searching for black holes in subways. *Theory of Computing Systems 50* (2012), pp. 158–184.

[21] Fraigniaud, P., Gąsieniec, L., Kowalski, D. R., and Pelc, A. Collective tree exploration. *Networks 48* (2006), pp. 166–177.

[22] Frederickson, G. N., Hecht, M. S., and Kim, C. E. Approximation algorithms for some routing problems. In *Proceedings of the 17th Symposium on the Foundations of Computer Science* (1976), pp. pp. 216–227.

[23] Koutsoupias, E. Communications privés. *Dans le cadre de l'article [13]* (1999).

[24] Kranakis, E., and Krizanc, D. An algorithmic theory of mobile agents. In *Proceedings of the 2nd international conference on Trustworthy global computing* (2006), pp. 86–97.

[25] Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., and Koeni, S. Auction-based multi-robot routing. In *Proceedings of Robotics : Science and Systems* (2005), pp. 343–350.

[26] Markou, E., and Pelc, A. Efficient exploration of faulty trees. *Theor. Comp. Sys. 40* (2007), pp. 225–247.

[27] Muslea, I. The very offline k-vehicle routing problem in trees. In *Proceedings of the XVII International Conference of the Chilean Computer Science Society* (1997), pp. 155–163.

[28] Oh, H., Shin, H., Tsourdos, A., White, B., and Silson, P. Coordinated road network search for multiple uavs using Dubins path. *Advances in Aerospace Guidance, Navigation and Control* (2011), pp. 55–65.

[29] Osterhues, A., and Mariak, F. On variants of the k-chinese postman problem. In *Proceedings of Operations Research und Wirtschaftsinformatik (Universitat Dortmund)* (2005), vol. 30.

[30] Rao, N. S. V., Kareti, S., Shi, W., and Iyengar, S. S. Robot navigation in unknown terrains : Introductory survey of non-heuristic algorithms. *Oak Ridge National Laboratory for the U.S. Department of Energy, under contract DE-AC05-84OR21400* (1993).

[31] Williams, K., and Burdick, J. Multi-robot boundary coverage with plan revision. In *Proceedings 2006 IEEE International Conference on Robotics and Automation* (2006), pp. 1716–1723.

[32] Xu, Z., Xu, D., and Zhu, W. Approximation results for a min–max location-routing problem. *Discrete Applied Mathematics 160* (2012), pp. 306–320.