UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

COMPROMIS ENTRE LE TEMPS DU RENDEZ-VOUS ET LA COMMUNICATION ENTRE LES AGENTS MOBILES

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

PAR

SAMIR ELOUASBI

SOUS LA SUPERVISION DU PROFESSEUR ANDRZEJ PELC

JUILLET 2018

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Cette Thèse intitulée :

COMPROMIS ENTRE LE TEMPS DU RENDEZ-VOUS ET LA COMMUNICATION ENTRE LES AGENTS MOBILES

présentée par

Samir Elouasbi

pour l'obtention du grade de Philosophiae Doctor (Ph.D)

Membres du jury :

Dr. Kamel Adi ..... Président du jury, UQO
Dr. Jurek Czyzowicz ..... Examinateur interne, UQO
Dr. Paola Flocchini ..... Examinatrice externe, Université d'Ottawa
Dr. Andrzej Pelc ..... Directeur de recherche, UQO

Thèse acceptée le : 31 octobre 2018

À mes chers parents.

À mon épouse et mes enfants.

À tous les membres de ma famille qui m'ont supporté.

Remerciement

Je ne remercierais jamais assez mon professeur et mon directeur de recherche Dr. Andrzej Pelc pour tout son soutien et son encadrement. Je suis fier d'avoir accompli ce travail mais je suis extrêmement fier d'être son étudiant, ceci pour moi est un privilège et un grand honneur et j'espère du fond du cœur que les liens d'amitié que nous avons tissés resteront toujours et vivront pour l'éternité.

Je tiens fortement aussi à remercier mon professeur Dr. Jurek Czyzowicz, non seulement pour le temps qu'il a consacré à lire et corriger cette thèse, mais aussi pour toutes les choses que j'ai apprises avec lui durant mes études doctorales.

J'aimerais aussi remercier Dr. Paola Flocchini et Dr. Kamel Adi qui m'ont honoré d'être des membres du jury de ma thèse. Seulement avoir les noms de ces grands chercheurs dans mon travail représente pour moi une grande fierté. Je leur suis très reconnaissant.

Mes pensées vont aussi à une personne qui m'a initié à la recherche, mon professeur et mon ami Dr. Mohamed Mejri avec qui j'ai appris beaucoup de choses qui m'étaient très utiles dans mes études.

Je profite aussi de cette occasion pour remercier tous mes amis pour leurs encouragements, la liste est longue mais j'aimerais surtout remercier Issam pour son aide et son support.

Enfin, un grand MERCI à tous ceux et celles qui ont cru en moi et dont leurs belles paroles sonnent toujours dans mes oreilles.

Table des matières

Résumé

Deux agents mobiles qui se trouvent initialement dans des positions différentes d'un environnement anonyme, doivent se rencontrer. Ils possèdent des identifiants différents représentés par des entiers positifs et se déplacent de manière synchrone. Cette tâche est connue dans la littérature sous le nom du *rendez-vous*. Elle est accomplie lorsque les agents réussissent à se trouver dans le même nœud dans le cas d'un réseaux modélisé par un graphe ou ils réussissent à se rencontrer dans le cas où les agents se trouvent dans un plan.

Dans cette thèse, nous nous intéressons à étudier l'impact de la communication sur la réalisation du rendez-vous et sur l'amélioration de son temps d'achèvement. Dans le modèle classique du rendez-vous, les agents n'ont pas la capacité de communiquer. Dans nos recherches, les agents sont capables d'utiliser trois moyens alternatifs de communication : l'émission d'un *bip* qui est un signal sonore très court, l'écriture sur un *tableau blanc* associé à chaque nœud du graphe et la possibilité de renifler grâce à des capteurs de détection d'odeur.

Nous présentons des algorithmes déterministes qui permettent aux agents de réaliser le rendez-vous dans ces trois situations de communication différentes. Nous prouvons aussi les bornes supérieures et inférieures, lorsque c'est possible, sur le temps d'exécution de ces algorithmes.

mots clés : agent mobile, rendez-vous, bip, détection, algorithme déterministe, communication, tableaux blancs, reniflement.

Abstract

Two mobile agents, starting from different positions of an anonymous environment, have to meet. Agents have different identifiers that are positive integers and move synchronously. This task is known as *rendezvous*. The rendezvous occurs when agents meet at the same node in the case of a network modeled by a graph or they succeed to meet in the case where they are in the plane.

In this thesis, we study the impact of communication on accomplishing rendezvous and on the improvement of its completion time. In the classic rendezvous model, agents do not have the ability to communicate. In our research, we provide three alternative communication ways to the agents : transmit a *bip* which is a very short sound signal, write on a *whiteboard* associated to each node of the graph and sniff by using sensors.

We present deterministic algorithms that allow agents to accomplish rendezvous in these three different situations of communication. We also prove the upper and lower bounds, where possible, on the execution time of these algorithms.

Keywords : mobile agent, rendezvous, beep, detection, deterministic algorithm, communication, whiteboards, sniffing.

Chapitre 1

Introduction

1.1 Motivation

Les grands projets sont en général réalisés par des équipes. On parle maintenant de différents systèmes de gestion d'équipes : Agile, Scrum, Waterfall ... etc. Ces systèmes ont pour but de créer un environnement adéquat de travail, de communication, de partage de tâches et de collaboration. Dans le domaine de l'informatique, le temps coûte cher ; un logiciel qui traite une certaine tâche en quelques minutes vaut plus qu'un autre qui traite la même tâche en des heures. De nos jours et avec l'explosion technologique, le traitement centralisé qui consiste par exemple à utiliser un seul ordinateur pour exécuter plusieurs tâches n'est plus efficace. On parle maintenant du traitement des données massives (du mot anglais *Big Data*) qui sont quantifiées en *yottaoctet* qui vaut 10^{24} *octets*, d'où l'importance du traitement distribué.

Plusieurs chercheurs, non seulement en informatique mais aussi en mathématiques et ingénierie, s'intéressent au calcul distribué. Un exemple très simple pour comprendre l'intérêt de ce domaine est la construction d'une maison. Pensons au temps que cette tâche va prendre si elle est faite par une seule personne ! L'équipe qui va construire

cette maison doit se partager les tâches (collaboration), faire un suivi (communication) et se débarrasser des intrus qui ne connaissent rien en construction (sécurité). Parmi les problèmes qu'on trouve fort présents en calcul distribué est le problème du *rendez-vous*. Comme son nom l'indique, le rendez-vous permet à des entités mobiles (des logiciels informatiques qui parcourent la toile ou des robots qui se déplacent dans un environnement physique) de se rencontrer pour échanger de l'information ou se partager des tâches. Dans la littérature, ces entités sont connues sous le nom d'*agents mobiles*. L'importance d'étudier le problème du rendez-vous des agents mobiles (logiciels ou physiques) s'avère de plus en plus important à travers le temps. Les agents logiciels naviguant dans un réseau doivent se rencontrer pour échanger les données acquises préalablement et les robots mobiles doivent se rencontrer pour échanger des échantillons du terrain pris dans une mine contaminée.

Dans cette thèse, nous allons nous intéresser au problème du rendez-vous pour des agents mobiles dans différentes situations. Nous allons aussi présenter une revue de la littérature qui permettra aux lecteurs d'avoir une idée assez large sur le domaine et sur plusieurs résultats réalisés dans les dernières années.

1.2 Le modèle

Dans cette section, nous allons parler plus en détail du rendez-vous. Pour mieux comprendre ce problème, nous présentons l'exemple suivant. Deux agents mobiles doivent se rencontrer pour effectuer ensemble une tâche de calcul ou se partager de l'information. Cette phrase bien que simple, cache beaucoup de possibilités.

- les *deux agents mobiles* peuvent être soit des programmes qui naviguent dans un réseau informatique soit des robots. À part la mobilité, ces agents peuvent avoir des mémoires de différentes tailles comme ils peuvent avoir des mémoires volatiles qui servent à faire

seulement du traitement sans stocker les données. Chaque agent peut avoir un identifiant unique qui le différencie de l'autre ; dans le cas où les agents sont identiques, c'est à dire qu'il n'y a aucun moyen de les différencier, on parle d'agents *anonymes*. Les agents peuvent aussi, être capable ou non, de communiquer entre eux.

- *doivent se rencontrer*, les agents ont donc besoin d'un environnement de rencontre. Cet environnement peut être géométrique (terrain, labyrinthe, ...) ou réseau qui est modélisé par un graphe. L'environnement peut être connu ou non par les agents. Le déplacement dans l'environnement se fait de deux manières : synchrone ou asynchrone. Nous parlerons plus en détail de ces deux notions dans le chapitre 2.

- Les agents doivent se rencontrer, mais dans combien de temps ? Un rendez-vous qui se fait dans un temps *très long* n'est pas très intéressant. C'est pour cela que les chercheurs ne s'intéressent pas seulement à la réalisation du rendez-vous, mais aussi au temps que ça va prendre pour le réaliser (lorsque c'est possible de le calculer).

Dans notre étude, nous nous intéressons à deux agents mobiles qui possèdent deux identifiants différents et qui veulent se rencontrer dans un réseau ou dans le plan. Dans le premier cas, le réseau est modélisé par un graphe connexe, non orienté et le rendez-vous doit se faire dans un nœud du graphe. Dans le deuxième cas, les agents sont représentés par des disques dans le plan cartésien et la rencontre se réalise lorsque les disques se touchent. Les agents se déplacent de manière synchrone et peuvent communiquer entre eux.

Dans le cas du graphe, notre but est de réaliser des algorithmes déterministes de rendez-vous qui ne dépendent pas de la connaissance des étiquettes des nœuds et qui peuvent permettre la réalisation de la rencontre, lorsque c'est possible. Notre intérêt à concevoir de tels algorithmes est motivé par les deux arguments suivants : le premier est que si les agents ont la capacité de connaître les identifiants des nœuds alors ils peuvent tout simplement s'entendre à se rencontrer dans le nœud qui possède le plus

petit identifiant. Dans ce cas, le problème du rendez-vous devient une tâche de parcours du graphe pour trouver l'identifiant voulu. Le deuxième est que souvent les nœuds ne révèlent pas leurs identifiants pour des raisons de sécurité et de confidentialité. Par contre, les arêtes incidentes à un nœud v ont des numéros distincts qui appartiennent à l'ensemble $\{0, 1, ..., d-1\}$, où d est le degré de v. En d'autres termes, chaque arête $\{u, v\}$ du graphe possède deux étiquettes qui s'appellent des *numéros de port* en u et en v. Les agents ont la capacité de lire les numéros de port de chaque nœud qu'ils visitent. La numérotation des ports est *locale*, c'est à dire qu'il n'existe aucune relation entre les numéros de ports associés à deux nœuds différents. L'estimation du temps du rendez-vous se fait en calculant le nombre de rondes écoulées entre le démarrage du deuxième agent, celui qui commence plus tard l'exécution de l'algorithme, et le moment de la rencontre.

Dans le cas du plan, les agents sont modélisés par des disques de diamètre 1 et notre but est de réaliser des algorithmes déterministes qui peuvent permettre aux agents de se toucher pour accomplir la rencontre, lorsque c'est possible. L'estimation du coût de la rencontre se fait en calculant la distance totale parcourue par les deux agents pour la réalisation de la rencontre.

Notre évaluation du temps du rendez-vous dans les deux modèles du réseau se fait en pire cas, c'est à dire dans la pire situation qui rend la rencontre la plus compliquée possible.

1.3 Problèmes et résultats

Tous les résultats présentés sont démontrés par des preuves rigoureuses qui utilisent des méthodes combinatoires et des éléments de la théorie des graphes.

Premier problème.

Nous avons présenté en premier lieu un algorithme déterministe qui permet à deux agents de faire le rendez-vous avec détection dans un réseau arbitraire et anonyme. Cet algorithme est valable pour le modèle de communication par bips entre les agents. Pour ce qui est du temps, notre algorithme réalise le rendez-vous avec détection en temps polynomial en n et $\log \ell$, où n est la taille du réseau et ℓ est la plus petite étiquette des deux agents. Notre deuxième résultat est une réflexion sur le fonctionnement de notre premier algorithme. Nous avons remarqué que les agents dépensent beaucoup d'énergie dans la solution que nous avons proposée. Plus précisément, le nombre total de parcours d'arêtes qu'un agent doit faire est proportionnel au temps de rendez-vous. Nous nous sommes donc posés la question suivante : si les agents ont une énergie limitée, c'est-à-dire qu'ils sont capables de faire au plus c mouvements pour un certain entier c, seront-ils toujours capables de faire le rendez-vous avec détection dans le modèle de bips. Nous avons prouvé, un résultat qui peut paraître surprenant, que le temps de ce rendez-vous est exponentiellement supérieur à celui des agents à énergie illimitée. Le dernier résultat que nous présentons dans ce projet est un algorithme déterministe qui résout le problème du rendez-vous avec détection dans le modèle de bips global pour des agents à énergie limitée en temps aussi rapide que pour les agents à énergie illimitée dans des réseaux de taille bornée. Le modèle global est un modèle plus fort qui permet aux agents de communiquer à distance.

Une version préliminaire [54] de cette étude a été présentée et publiée dans

Elouasbi, S., and Pelc, A. Deterministic rendezvous with detection using beeps. In *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers* (2015), pp. 8597.

Pour la version complète [56], elle a été publiée dans

Elouasbi, S., and Pelc, A. Deterministic rendezvous with detection using beeps. *Int. J. Found. Comput. Sci. 28*, 1 (2017), 77.

Deuxième problème.

Nous répondons à la question suivante : est-il possible d'accomplir le rendez-vous déterministe entre deux agents en un temps de l'ordre du temps optimal d'exploration ? Une réponse négative est immédiate si les agents n'ont pas la possibilité de marquer les nœuds du graphe, même dans la classe des arbres. Il est donc naturel de se demander quelle est la plus petite taille des tableaux blancs associés aux nœuds, qui peuvent être utilisés par les agents pour laisser des messages, pour leur permettre de réaliser le rendez-vous en un temps proportionnel à celui de l'exploration la plus rapide.

Nous montrons que pour les arbres de taille n, cette plus petite taille est d'ordre $\Theta((\log L)/n)$, où $\{1, \cdots, L\}$ est l'ensemble des étiquettes que peut avoir un agent. Nous fournissons un algorithme de rendez-vous déterministe qui fonctionne en temps $O(n)$ en utilisant de tels tableaux blancs. Nous prouvons aussi une borne inférieure sur la taille des tableaux blancs qui est valide même pour la classe des lignes.

Troisième problème.

À l'opposé des deux premiers problèmes, nous nous intéressons dans ce cas à la rencontre entre deux agents mobiles dans le plan. Les agents sont modélisés par des disques de diamètre 1 et la rencontre se produit lorsque ces disques se touchent. Les agents sont équipés de boussoles identiques, ont des horloges synchronisées et se déplacent à la même vitesse constante, normalisée à 1. Ils possèdent aussi des capteurs qui leur permettent de détecter l'odeur ; autrement dit, les agents peuvent communiquer en utilisant le *reniflement*.

Nous supposons que ces capteurs permettent aux agents d'estimer la distance qui les séparent sans qu'aucun d'eux ne connaît dans quelle direction se trouve l'autre. Nous considérons deux modèles d'estimation. Dans les deux modèles, un agent lit son capteur au moment de son apparition dans le plan puis à la fin de chaque déplacement. Cette lecture (avec les précédentes) détermine la décision concernant le prochain déplacement. Dans les deux modèles, la lecture du capteur indique aussi à l'agent si l'autre agent est déjà présent.

Nous considérons deux modèles de reniflement. Dans le modèle monotone, chaque agent peut conclure pour n'importe quelles deux lectures à des moments t_1 et t_2, si la distance qui le sépare de l'autre agent à l'instant t_1 était plus petite, égale ou plus grande que celle à l'instant t_2. Dans le modèle binaire qui est plus faible, chaque agent peut savoir, à n'importe quelle lecture, s'il est à une distance inférieure à ρ ou à une distance d'au moins ρ de l'autre agent, pour un certain réel $\rho > 1$ inconnu par les agents. Dans les deux modèles, l'intensité de l'odeur diminue avec la distance. Dans le modèle monotone, nous supposons que le capteur est idéalement précis et peut mesurer tout changement d'intensité. Dans le modèle binaire, nous supposons seulement que le capteur peut détecter l'odeur au-dessous d'une certaine distance (sans pouvoir mesurer l'intensité) et au-dessus de laquelle l'odeur est trop faible pour être détectée.

Nous présentons des algorithmes qui permettent de faire la rencontre déterministe dans le plan. Nous montrons aussi l'impact des deux façons de renifler sur le coût de la rencontre défini comme la distance totale parcourue par les deux agents jusqu'à la réalisation de la rencontre. Pour le modèle monotone, nous prouvons que notre algorithme permet de réaliser la rencontre au coût $O(D)$, où D est la distance initiale qui sépare les agents. Cette complexité est évidemment optimale. Pour ce qui est du modèle binaire, nous fournissons un algorithme qui permet, à deux agents qui se trouvent initialement à une distance inférieure à ρ, de faire la rencontre au coût $O(\rho \log \lambda)$, où λ est la plus

grande étiquette. Nous prouvons aussi que ce coût ne peut pas être amélioré en général. Enfin, nous remarquons que dans le modèle binaire, si les agents se trouvent initialement à une distance $\alpha\rho$, pour une certaine constante $\alpha > 1$, le reniflement n'ajoute aucune amélioration au coût de la rencontre, c'est-à-dire que le coût de la rencontre optimale en pire cas est du même ordre de grandeur que celui dans le cas où les agents ne possèdent pas la capacité de renifler.

Une version préliminaire [55] de cette étude a été présentée et publiée dans
Elouasbi, S., and Pelc, A. Deterministic meeting of sniffing agents in the plane. In *Structural Information and Communication Complexity - 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers* (2016), pp. 212-227.

Pour la version complète [57], elle a été publiée dans
Elouasbi, S., and Pelc, A. Deterministic meeting of sniffing agents in the plane. *Fundam. Inform. 160*, 3 (2018), 281-301.

1.4 Structure de la thèse

Nous avons structuré ce document de la manière suivante :

- Le chapitre 1 présente une introduction au problème du rendez-vous et une brève description du modèle, du problème et des résultats.
- Le chapitre 2 comporte une revue de la littérature : nous présentons plusieurs exemples des travaux réalisés dans le domaine.
- Le chapitre 3 présente notre premier résultat concernant le rendez-vous déterministe avec détection utilisant les bips comme moyen de communication.
- Le chapitre 4 présente notre deuxième résultat concernant le rendez-vous déterministe utilisant les tableaux blancs comme moyen de communication.

- Le chapitre 5 présente notre troisième et dernier résultat concernant la rencontre déterministe utilisant le reniflement comme moyen de communication.
- Le chapitre 6 présente une conclusion générale de cette recherche et quelques problèmes ouverts.
- Cette thèse se termine par une bibliographie.

Chapitre 2

Revue de la littérature

2.1 Introduction

Le problème du rendez-vous est parmi les sujets les plus intéressants et les plus pertinents dans le domaine de l'algorithmique distribué. Comment des programmes informatiques mobiles sur Internet peuvent échanger de l'information et la traiter si le processus de leur rencontre n'est pas bien défini ? Comment des robots mobiles peuvent collaborer pour faire par exemple la carte d'un terrain vaste en un temps minimal s'ils ne peuvent pas se rencontrer pour se partager les tâches ? Même pour les êtres humains, comment deux personnes peuvent se rencontrer dans une ville qu'ils visitent pour la première fois lorsque les deux personnes viennent de différents endroits ? Cette problématique a été initiée par Schelling [92] dans les années soixante : Comment deux personnes parachutées dans deux endroits différents peuvent se rencontrer en utilisant seulement deux cartes identiques et dans l'absence de tout moyen de communication ? À travers cette section, nous allons faire le tour de quelques écrits qui ont traité le problème du rendez-vous. Nous commencerons par un cas particulier du rendez-vous qui est l'exploration. On peut dire que l'exploration est un rendez-vous dans lequel une entité

reste immobile et l'autre entité essaye de la trouver. Nous verrons quelques exemples de l'exploration d'un emplacement géométrique tel que les terrains et d'autres exemples de l'exploration des réseaux. Nous allons aussi présenter quelques articles sur un sujet qui est très proche de l'exploration : le patrouillage. La grande partie de cette revue de la littérature va être consacrée au rendez-vous. Nous allons parler des modèles du rendez-vous dans des environnements sains pour ensuite parler du rendez-vous dans des circonstances où il faut tolérer les pannes, une réalité qu'on ne peut pas détourner. Nous terminerons cette section par des recherches qui ont été faites sur la communication dans les réseaux et sur son impact sur le problème du rendez-vous.

2.2 Exploration

Le problème d'exploration a été largement étudié dans la littérature et il reste toujours comme un vaste champ d'étude grâce à son intérêt et son rapport avec notre vie quotidienne ou avec l'évolution technologique des réseaux informatiques et d'Internet. On peut facilement distinguer deux types d'environnement pour faire l'exploration : L'environnement géométrique et les réseaux. Lorsqu'on parle de l'environnement géométrique, on fait référence aux terrains, aux pièces, aux océans ... etc. Ces espaces géométriques peuvent contenir des obstacles de différentes formes ce qui rend l'exploration plus compliquée. Pour ce qui est des réseaux, on les représente par des graphes et c'est toute la théorie des graphes qui embarque pour permettre aux chercheurs de définir plusieurs modèles avec différentes restrictions : des graphes orientés versus non-orientés, possibilité de marquer ou de ne pas marquer les nœuds, les nœuds peuvent être identifiés ou non ... etc.

L'exploration avec les robots ou les agents mobiles s'avère très importante lorsqu'on veut connaître les caractéristiques d'un endroit, ou chercher des virus ou des intrus

dans les réseaux, ou décontaminer une zone dangereuse (déchets, nucléaires, terrains dangereux ...) et nous allons voir à travers les quelques recherches présentées que le problème de l'exploration est et restera toujours un domaine plein de défis.

2.2.1 Exploration dans le plan

Lorsqu'on parle de l'exploration dans le plan, on veut faire référence à l'aspect géométrique de l'environnement exploré. Un plan peut être un terrain, un océan ou un plancher d'une maison. Ce genre d'exploration peut avoir plusieurs objectifs : faire la carte complète d'un terrain pour connaître sa topologie, pour identifier les obstacles et les limites ou pour chercher un trésor. Dans cette section, on va présenter quelques résultats de recherche qui touchent à ce domaine. Mais avant ça, on aimerait parler d'un article qui donne une vue générale et détaillée des différents types d'exploration planaire. Rao et al.[90] commencent cet article par une classification des différents algorithmes d'exploration, qu'ils nomment des « algorithmes de navigation ». Il y a les algorithmes qui étaient conçus pour faire seulement de l'exploration comme par exemple traverser un labyrinthe sans tenir compte du temps que ça va prendre. Un autre objectif est de s'intéresser à la performance de l'exploration comme par exemple chercher la plus courte distance entre deux positions dans un terrain. Le dernier objectif est de s'intéresser plus à la performance des robots mobiles qui font l'exploration comme par exemple améliorer leurs capacités de calcul. L'autre critère de classification est celui de la manière de détection qu'utilise un robot mobile pour explorer le plan. Les auteurs de [90] en définissent deux types : la vision qui permet au robot de voir un obstacle et le sens de toucher qui va lui permettre de le toucher dans l'absence de la vision. À la fin de leur article, Rao et al. présentent quelques résultats sur l'exploration des graphes planaires(ajouter une référence) et dressent une liste de problèmes qui touchent l'exploration dans les plans.

Considérons maintenant le problème suivant : on vous met dans le coin d'une chambre rectangulaire obscure. On vous dit que le côté de la chambre est n et que la table se trouve au centre de la chambre. Mais il y a une contrainte : la chambre contient des chaises rectangulaires alignées à ses murs et vous n'avez pas le droit de les déplacer. On voit que la formulation du problème est très simple mais ce n'est pas évident de trouver une solution algorithmique pour vous permettre de rejoindre le centre de la chambre. Ce problème est connu dans la littérature par « le problème de la pièce ». La seule manière de détecter les obstacles est de les toucher. Bar-Eli et al.[7] ont présenté un algorithme déterministe qui permet de résoudre ce problème. Leur algorithme réussit à trouver un chemin de longueur $O(n \log n)$. De plus, cette exploration se fait avec un ratio de compétitivité de valeur $O(\log n)$. Le ratio de compétitivité est le maximum des rapports du coût de l'exploration faite par un robot et le coût de l'exploration faite en temps optimal. À la fin de leur article, les auteurs prouvent qu'il n'existe aucun algorithme déterministe qui peut faire mieux même dans le cas où la chambre est éclairée par la lumière.

Czyzowicz et al.[29] ont étudié le même type de problème mais dans un autre scénario. Cette fois-ci, on fait affaire à un terrain inconnu sous forme de polygone, et à des obstacles de la même forme. Les auteurs ont distingué deux cas dans leur étude : le cas où le robot possède une vision illimitée, c'est à dire qu'à partir d'une position donnée, il peut voir toutes les parties du terrain qui ne sont pas cachées par les obstacles ; dans ce cas, les auteurs ont construit un algorithme déterministe qui permet de faire l'exploration du terrain en parcourant une distance en $O(P + D\sqrt{k})$ où P est le total du périmètre du terrain plus les périmètres de tous les obstacles, D est le diamètre de l'enveloppe convexe du terrain (le plus petit polygone convexe qui englobe tout le terrain) et k représente le nombre des obstacles dans le terrain. Ils ont aussi prouvé que cette distance est optimale même dans le cas où le terrain est connu par le robot. L'autre cas est celui où le robot

possède une vision limitée, c'est à dire qu'il ne peut voir que la partie du terrain qui se trouve dans un rayon égal à 1 dans son champ de vision. Avec cette nouvelle contrainte, les auteurs ont réalisé un autre algorithme qui permet de faire l'exploration du terrain en faisant un trajet de longueur $O(P + A + \sqrt{Ak})$ où A représente la surface du terrain excluant les surfaces de tous les obstacles. Ils prouvent aussi que cette distance est optimale.

Jusqu'ici nous avons vu des situations où l'exploration se fait dans le but d'aboutir à un emplacement précis ou de visiter tous les points du plan. Nous allons voir maintenant un autre aspect de l'exploration qui est la construction de la carte complète d'un terrain. Avoir une carte permet au robot de bien faire ses déplacements et de trouver la meilleure manière pour parcourir le plan. Deng et al. [40] se sont intéressés au problème suivant : un robot doit construire une carte d'une pièce anonyme où se trouve des obstacles. Il doit commencer par un point d'entrée et quitter par un point de sortie une fois l'exploration est terminée. Il est équipé d'une mémoire qui lui permet d'enregistrer tous les points de la pièce. La carte que le robot va construire doit être complète de façon à ce que si on la donne à un autre robot, il sera capable de faire rapidement l'exploration de cette pièce. Les auteurs ont présenté un algorithme qui permet de construire une carte pour n'importe quelle pièce sous forme de polygone et dont les obstacles ont les contours alignés avec les murs de la pièce. Cet algorithme est compétitif puisque son ratio est borné par une constante. Dans cet article, les auteurs montrent aussi que le calcul du ratio compétitif est très compliqué. Ceci découle, comme les auteurs l'ont prouvé, que la construction d'une carte pour un polygone qui contient des obstacles est un problème NP-difficile bien qu'on puisse construire la carte du même polygone en temps polynomial lorsqu'il ne contient aucun obstacle. On ne doit pas oublier de mentionner que cet algorithme ne fonctionne que dans les cas où le robot connaît une borne supérieure sur le nombre d'obstacles dans le polygone.

L'exploration peut servir aussi à trouver un endroit précis. Ce problème est connu dans la littérature sous le nom de « Recherche d'un point dans le plan ». Gal [62] a mentionné que la meilleure stratégie à utiliser pour trouver un point dans un plan était de faire une exploration sous forme de spirale mais aucune preuve n'a été fournie pour démontrer que cette stratégie est optimale. Langetepe [79] reprend ce problème pour montrer effectivement que la stratégie de la spirale est optimale. L'idée de cette stratégie est la suivante : le début de l'exploration se fait à partir d'une position initiale nommée O et le point recherché t se trouve dans une position inconnue dans le plan. Soit Π le chemin qui définit la spirale. En pire cas, le point t ne se trouve pas dans Π. Considérons maintenant le point p_t qui se trouve dans Π tel que t appartient au segment Op_t et le segment tp_t n'intersecte la spirale qu'en p_t. Langetepe a prouvé que la stratégie de la spirale pour la recherche d'un point dans le plan est optimale et se fait avec un ratio compétitif de $17,289....$ Ce ratio est calculé en prenant la plus grande valeur de la division de la taille du chemin sur la spirale allant de O à p_t par la taille du segment Ot. Le livre d'Alpern et Gal [5] offre plus de détails sur cette stratégie.

Dans un autre article [80], Langetepe aborde le même problème d'exploration mais cette fois-ci pour trouver une ligne dans le plan. La ligne est trouvée lorsque l'exploration trouve un point qui appartient à cette ligne. Dans cet article, l'auteur s'est intéressé au cas où la ligne recherchée est parallèle à l'un des deux axes de l'espace euclidien. Il prouve qu'en utilisant un parcours sous la forme d'une spirale polygonale (c'est le même principe de la spirale normale mais son chemin contient des angles de moins de 180 degré), l'exploration peut se faire avec un ratio compétitif de $12,5385....$ Ce ratio est calculé en prenant la plus grande valeur de la division de la taille du chemin sur la spirale allant du point de départ jusqu'à la première intersection de la spirale avec la ligne recherchée par la taille du segment débutant au point de départ et perpendiculaire à cette ligne recherchée. Langetepe démontre aussi que sa stratégie est optimale. La

question qui reste ouverte est s'il peut exister une autre stratégie que la spirale et qui permet de faire ce genre d'exploration avec un ratio compétitif meilleur que $12,5385...$. La réponse à cette question semble être négative mais ça reste une conjecture qui n'est pas encore prouvée.

L'exploration dans le plan restera toujours un domaine intéressant, surtout qu'elle touche à la vie quotidienne des humains : exploration des villes visitées, de quartiers où on habite et des mers et océans. Ceci étant dit, il existe un autre domaine très vaste et très riche et à qui s'intéressent plusieurs chercheurs, c'est l'exploration des réseaux informatiques. C'est ce qu'on va voir dans la section qui suit.

2.2.2 Exploration dans les réseaux

Avec l'évolution informatique et Internet qui relie maintenant tous les coins du monde, les réseaux informatiques deviennent de plus en plus vastes et leur exploration s'avère une tâche primordiale pour plusieurs raisons : connaissance de topologie des réseaux, recherche des pannes et des intrusions, construction des cartes des réseaux, etc. Dans cette section, on va s'arrêter sur quelques articles de recherche qui ont traité l'exploration des réseaux.

Dans la littérature, les réseaux sont modélisés par des graphes. Selon chaque article, on expliquera le type du graphe utilisé et ses caractéristiques. Dans la section précédente, on parlait des robots mobiles lorsque l'exploration se fait dans le plan, dans les réseaux (ou les graphes), on parle des agents mobiles. En application, ces agents mobiles sont des agents logiciels, c'est à dire des morceaux de codes qui peuvent parcourir les nœuds du réseau et y exécuter diverses tâches.

On commence par l'article de Dessmark et Pelc[42] dans lequel les auteurs s'intéressent à l'exploration d'un graphe connexe et non-orienté. Le but est de parcourir ce

graphe en utilisant le minimum possible de traversées des arêtes. Les auteurs cherchent à présenter des algorithmes qui permettent de faire cette exploration avec un ratio compétitif intéressant. Dans le cas des graphes, le ratio compétitif est la borne supérieure sur le quotient du nombre d'arêtes traversées par l'agent mobile en exécutant un algorithme et de ce nombre optimal dans le cas où l'agent connaît toutes les caractéristiques du graphe (coût optimal). Dessmark et Pelc étudient l'exploration dans trois types de graphes : les lignes, les arbres et les graphes arbitraires, tous de taille n. Pour chacun de ses types, les auteurs utilisent trois scénarios différents pour calculer le ratio : le premier scénario est celui où l'agent ne connaît ni la topologie ni la taille du graphe. Le deuxième est celui où l'agent possède une carte du graphe sans connaître sa propre position initiale et dans le troisième scénario, l'agent possède une carte du réseau sur laquelle est indiquée sa position initiale.

Dans le cas sans aucune information, le DFS (Algorithme de parcours en profondeur) est le choix évident pour parcourir ce genre de graphes. Les auteurs montrent que le DFS est une solution optimale avec un ratio compétitif égal à 2 pour l'exploration des trois types de graphes cités ci-dessus lorsque l'agent ne possède aucune information sur le réseau.

Pour les lignes, les auteurs présentent des algorithmes optimaux et prouvent que le ratio compétitif optimal dans le cas où l'agent possède une carte du réseau sans connaître sa position initiale est $\sqrt{3}$ et que lorsque la carte lui indique sa position initiale, ce ratio est égal à $\frac{7}{5}$.

Pour ce qui est des arbres maintenant, le DFS n'est pas une solution optimale pour le deuxième et troisième scénarios cités ci-dessus. Les auteurs prouvent que le ratio optimal est inférieur ou égale à 1.99 dans le cas du deuxième scénario et inférieur strictement à 1.5 dans le cas du troisième scénario. Les auteurs prouvent aussi que $\sqrt{3}$ est une borne

inférieure dans le cas où l'agent possède une carte du réseau sans connaître sa position initiale.

On termine l'étude par les graphes arbitraires, connexes et non-orientés. Les auteurs prouvent que le DFS reste la solution optimale pour tous les scénarios et que le ratio compétitif optimal est égal à 2.

L'exploration ne se limite pas seulement à parcourir des graphes dans un temps optimal, elle permet aussi de créer une carte d'un graphe. On peut dire que cette carte représente une image fidèle du graphe. Un des articles qui traite ce sujet est celui de Chalopin, Das et Kosowski [16]. Les auteurs s'intéressent à la construction d'une carte d'un graphe anonyme, c'est à dire que ses nœuds ne sont pas étiquetés. Par contre, chaque nœud de degré d possède des numéros de ports de 1 à $d-1$ qu'un agent peut voir une fois qu'il est dans ce nœud. Le temps de l'exploration est défini comme le nombre total de traversées d'arêtes par un agent pour faire l'exploration du graphe en entier. Cet article présente aussi un compromis entre le temps de l'exploration et la taille de mémoire nécessaire pour un agent mobile pour faire l'exploration d'un tel graphe. Chalopin, Das et Kosowski ont utilisé la notion du UXS qu'on peut traduire par « Séquences d'exploration universelle » introduite par Koucky [70] et reprise par Reingold [91] qui a démontré que l'utilisation de l'UXS permet de faire l'exploration d'un graphe en temps polynomial. L'UXS est une suite numérique qui permet à un agent de visiter tous les nœuds du graphe à partir de n'importe quelle position initiale. Les auteurs ont démontré les deux résultats suivants : Le premier est que si l'agent connaît uniquement une borne supérieure sur le nombre des nœuds n du graphe et sur son degré maximal d sans connaître sa propre position initiale, alors il peut construire la carte du graphe avec une exploration en temps $O(n^6 d^2 \log n)$ s'il est équipé d'une mémoire de taille $O(n^6 d^2 \log n)$ bits et en un temps légèrement pire $O(n^6 d^3 \log n \log d)$, mais en mémoire de taille seulement $O(n^3 d^2 \log n \log d)$ bits. L'autre résultat est que si

l'agent peut marquer sa position initiale par un jeton mais il n'a aucune information sur la taille du graphe ou sur son degré maximal alors il peut construire la carte du graphe en un temps polynomial avec une mémoire de taille optimale $\Theta(\log n)$ bits ; en revanche, si la taille de la mémoire est $O(nd \log n)$ alors la construction de la carte peut se faire en un temps $O(n^3 d)$.

Gorain et Pelc [65] traitent l'exploration d'une autre manière en répondant à la question suivante : quelle serait la taille nécessaire de l'information fournie à un agent pour qu'il puisse faire l'exploration en un temps donné ? Ce genre de problème est connu dans la littérature sous le nom de « Exploration avec conseil ». Le *conseil* est une information sous forme de chaîne binaire qui permet à un agent de performer son exploration et de la faire dans un meilleur temps si c'est possible. Pour ce genre d'exploration, les auteurs se sont intéressés à des réseaux sous forme de graphe anonyme, le même utilisé par Chalopin, Das et Kosowski [16]. Le temps de l'exploration est défini comme dans [16]. Le conseil est fourni à l'agent par un « Oracle » ; on peut définir ce dernier comme une entité qui possède des connaissances sur le réseau. Dans cet article, les auteurs s'intéressent à deux types d'oracle : le premier appelé « oracle avec carte » qui connaît tout le graphe avec ses numéros de port pour chaque nœud et le deuxième qui est appelé « oracle avec instance » qui diffère du précédent par l'ajout de la connaissance de la position initiale de l'agent. Voici les résultats de leur recherche :

- Un conseil de taille $\log \log \log n$ - c est suffisant pour faire l'exploration en un temps polynomial où n est la taille du graphe et c est une constante. Ce résultat est valable pour les deux types d'oracles mentionnés ci-dessus. Les auteurs montrent aussi que l'exploration en temps polynomial est impossible si la taille du conseil est $\log \log \log n$ - $\phi(n)$ tel que ϕ est une fonction divergente vers l'infini.

- Un conseil de taille $O(n \log n)$ qui provient d'un « oracle avec carte » permet à l'agent de faire l'exploration en temps $O(n^2)$, par contre s'il provient d'un « oracle avec instance », l'agent fera l'exploration en temps $O(n)$.

- Toute exploration dont le conseil provient d'un « oracle avec carte » ne peut se faire en un temps meilleur que $\Omega(n^2)$ et ceci peu importe la taille de ce conseil. Les auteurs prouvent aussi que le plus petit conseil de l'oracle avec carte permettant de faire l'exploration en temps $\Theta(n^2)$ est plus grand que n^δ pour n'importe quelle constante $\delta < 1/3$.

- Pour « l'oracle avec instance », le conseil de taille $O(n \log n)$ est suffisant pour atteindre le temps $O(n)$. Les auteurs montrent qu'un tel conseil est nécessaire pour l'exploration en temps linéaire. Ils prouvent un résultat plus fort : avec un conseil de taille $o(n \log n)$, le temps d'exploration doit être au moins n^ϵ pour chaque constant $\epsilon < 2$.

- Un dernier résultat concernant les graphes de taille n est si la taille du conseil est $O(n)$ alors le temps de l'exploration ne peut pas être meilleur que $\Omega(n^2)$.

Dans le même article [65], Gorain et Pelc s'intéressent à ce type d'exploration dans les graphes hamiltoniens de taille n (un graphe hamiltonien est un graphe qui possède au moins un cycle qui passe par tous les nœuds exactement une seule fois). Dans le cas où le conseil provient d'un « oracle avec instance », l'agent peut accomplir l'exploration dans un temps optimal de $n - 1$ si la taille du conseil est $O(n \log n)$. Les auteurs montrent aussi que si ce type d'oracle fournit un conseil de taille $o(n \log n)$ alors l'exploration doit se faire en un temps minimum de $n + \Omega(n^\epsilon)$ pour n'importe quel ϵ inférieur strictement à 1. Dans le cas où le conseil provient d'un « oracle avec carte », le temps de l'exploration des graphes hamiltoniens ne peut pas être meilleur que $\Omega(n^2)$ et ceci peu importe la taille du conseil fourni.

Nous terminons cette partie en parlant de l'exploration des graphes en présence des pannes de liens. Le lien c'est l'arête d'un graphe et il est en panne lorsque l'agent mobile

ne peut pas l'utiliser pour se déplacer vers un nœud adjacent. Caissy et Pelc [15] ont étudié ce problème pour deux types de graphes : les anneaux et les graphes hamiltoniens de tailles supérieures ou égales à 3. L'agent connaît la topologie du graphe et sa position initiale mais il n'a aucune information sur les arêtes qui sont en panne, c'est à dire qu'il ne connaît ni leur emplacement ni leur nombre au départ. Les nœuds du graphe ne sont pas étiquetés mais l'agent peut voir les numéros de port d'un nœud une fois qu'il le visite et il peut aussi à ce moment voir les arêtes incidentes qui sont en panne. Les auteurs ont utilisé la notion du *surplus* pour créer des algorithmes *parfaitement compétitifs*. Le *surplus* est le ratio maximum du coût des algorithmes d'exploration en présence des liens en panne sur le coût optimsl d'une telle exploration en présence de ces pannes. Un agent exécute un algorithme optimal lorsqu'il connaît dès le début l'emplacement des pannes dans le graphe : ceci va lui permettre de bien choisir son parcours afin de faire le moins de traversées possibles. Un algorithme est dit *parfaitement compétitif* si son surplus est le plus petit parmi tous ceux des algorithmes de l'exploration qui ne connaissent pas l'emplacement des pannes. Les auteurs ont commencé par résoudre le problème de l'anneau non orienté et ils ont fourni un algorithme parfaitement compétitif ; le surplus de cet algorithme ne dépend que de la taille de l'anneau et son temps d'exploration est linéaire dans le nombre des arêtes du graphe qui ne sont pas en panne. Caissy et Pelc ont ensuite prouvé dans le même article [15] que n'importe quel algorithme de fouille en profondeur qui permet l'exploration d'un graphe hamiltonien aura un surplus au plus 10/9 fois plus grand que celui d'un algorithme parfaitement compétitif. De plus, si un tel graphe hamiltonien possède plus de 23 nœuds, alors son ratio ne peut jamais dépasser la valeur 1.0583. Pour conclure leur article [15], les auteurs ont prouvé qu'un algorithme de fouille en profondeur qui fait l'exploration d'un *graphe complet* a exactement un surplus égal à $\frac{2n-4}{n-1}$.

2.2.3 Le patrouillage

Pendant plusieurs années, le patrouillage ou le problème de la patrouille était considéré par les chercheurs qui travaillent dans le domaine de l'ingénierie et de la robotique. Depuis une quinzaine d'années, il a connu un grand intérêt de la part des chercheurs qui s'intéressent aux agents mobiles. Le patrouillage est défini comme un parcours d'un environnement par un ou plusieurs agents mobiles afin de visiter toutes les parties de cet environnement le plus souvent possible. On peut dire que le problème de la patrouille est une sorte d'exploration d'un environnement de manière périodique. L'environnement peut être géométrique tel qu'un terrain ou un réseau tel qu'un arbre. Le patrouillage a été introduit la première fois par Machado, Ramalho, Zucker et Drogoul [81] dans un système multi-agents. Les auteurs ont soulevé le problème des soldats dans les jeux vidéo qui ont pour mission de patrouiller une zone de guerre. Ils ont présenté une analyse approfondie du problème de la patrouille effectuée par plusieurs agents. Pour appuyer leur étude, les auteurs ont implémenté un simulateur sur une plate-forme de jeux vidéo. Cependant cette analyse [81] est expérimentale et non théorique. La première analyse théorique de ce sujet a été conduite par Chevaleyre [21] qui a présenté une classification de stratégies de patrouillage dans un graphe géométrique à n nœuds. Chaque arête possède un poids qui peut représenter soit la distance entre ses deux nœuds soit le temps de son parcours. Ces stratégies ont pour but d'offrir à k agents la possibilité de visiter chaque nœud le plus souvent possible. L'auteur a défini deux classes de stratégies de patrouillage : la classe des stratégies cycliques qui consistent à trouver un chemin fermé (qui commence et se termine dans le même nœud) qui couvre tout le graphe et la classe des stratégies régionalisées qui a pour principe de partitionner le graphe en k régions disjointes telles que chaque région est patrouillée par un seul agent. L'auteur a aussi fait

une comparaison des deux classes et a mentionné que le choix d'une stratégie dépend surtout de la topologie du graphe.

L'efficacité d'une stratégie de patrouillage est mesurée à l'aide de l'oisiveté (du mot anglais *idleness* [81]). L'oisiveté d'un point d'une arête est définie comme la durée du temps écoulé entre deux visites successives de ce point par un agent. En pire cas, l'oisiveté d'un graphe est la plus grande valeur de l'oisiveté d'un des points de ses arêtes.

La protection est parmi les objectifs du patrouillage. Pensons à une maison clôturée avec des gardiens qui doivent faire leurs tournées pour la protéger des voleurs et réfléchissons comment des soldats peuvent protéger la frontière de leurs pays des malfaiteurs qui veulent s'y infiltrer. Czyzowicz, Gasieniec, Kosowski et Kranakis [27] se sont intéressés à ce genre de problèmes qui est connu dans la littérature sous le nom du *patrouillage de frontière* ou *patrouillage de clôture*. L'objectif est de vérifier si la frontière est vulnérable, c'est à dire qu'un intrus peut réussir à la franchir, ou fournir une solution qui permet aux agents de se déplacer d'une certaine manière le long de la frontière pour ne laisser aucun de ses points sans surveillance pendant un intervalle de temps donné. Ces auteurs ont été les premiers à étudier ce problème en utilisant k agents qui possèdent des vitesses maximales distinctes. Ils ont commencé par traiter le cas du patrouillage d'un segment d'une droite. Les auteurs ont fourni un algorithme qui utilise une stratégie régionalisée et qui permet à chaque agent de parcourir sa région en zigzag. Ils ont prouvé que cet algorithme est optimal lorsque $k = 2$. Les auteurs ont ensuite étudié le cas d'une clôture circulaire que les agents peuvent patrouiller en utilisant un sens unique (cercle orienté). Ils ont fourni un algorithme qui n'utilise que r agents ($r < k$) qui sont placés à distance égale autour de la clôture et qui bougent à la vitesse maximale de l'agent le plus lent parmi les r. Les auteurs ont prouvé que cet algorithme est optimal lorsque $k < 5$. Pour $k = 2$, ils ont aussi prouvé que si les agents peuvent patrouiller la clôture en utilisant les deux sens (cercle non-orienté) alors il n'existe aucun algorithme de patrouillage utilisant

une stratégie cyclique qui peut faire mieux que celui qu'ils ont fourni. Dans cet article [27], les auteurs ont énoncé deux conjectures : la première est que l'algorithme qu'ils ont fourni pour traiter le cas du patrouillage dans un segment est aussi optimal pour tout $k > 2$. La deuxième est que l'algorithme qu'ils ont élaboré dans le cas du patrouillage d'un cercle orienté est aussi optimal pour tout $k > 2$. Pour la première conjecture, Kawamura et Kobayashi [69] ont prouvé qu'elle est correcte pour $k = 3$ mais qu'elle est fausse si $k \geq 4$. Pour ce qui est de la deuxième conjecture, Dumitrescu, Ghosh et Tóth [52] ont démontré qu'elle est fausse dans le cas où $k = 32$.

Un autre modèle étudié par Collins, Czyzowicz, Gasieniec, Kosowski, Kranakis, Krizanc, Martin et Morales Ponce [23] est celui du patrouillage d'une courbe par k agents qui possèdent la même vitesse maximale. Les auteurs se sont intéressés à des courbes composées de deux types de segments : des segments *vitaux* qui doivent être patrouiller et des segments *neutres* que les agents peuvent ignorer. Pour résoudre ce problème, les auteurs ont proposé une stratégie régionale pour patrouiller les segments vitaux. Ensuite, ils ont prouvé que l'oisiveté optimale peut être atteinte en utilisant une stratégie cyclique ou régionale pour le patrouillage des courbes fermés. Les auteurs ont montré que le choix de la stratégie dépend de la configuration des segments vitaux dans la courbe. Il faut aussi mentionné que pour résoudre ce problème, les agents ont été positionnés à distances égales et qu'ils ne pouvaient se déplacer que dans une seule direction.

De leur part, Czyzowicz, Kranakis, Pajak et Taleb [33] ont étudié le même modèle que l'article précédent mais avec des agents qui ont le pouvoir de changer leurs directions et de voir leur entourage à une distance donnée dans les deux directions. Chaque agent possède un champ de vision qui peut être différent de celui des autres. Une région de la courbe est donc protégée si elle se trouve dans le champ de vision d'au moins un agent. Les auteurs ont construit des algorithmes optimaux qui permettent de patrouiller une courbe (fermée ou non) par un nombre arbitraire k d'agents qui ont la même vitesse

maximale et qui peuvent posséder de différents champs de vision. Pour $k = 2$, ils ont fourni des algorithmes optimaux qui résolvent ce problème lorsque les agents possèdent des vitesses maximales différentes. Les auteurs ont clôturé cette recherche en prouvant que pour certaines classes de graphes, la résolution du problème de la patrouille avec des agents qui possèdent différents champs de vision est $NP - difficile$, et ceci même dans le cas où les agents possèdent la même vitesse maximale.

Nous terminons cette section par l'étude effectuée par Czyzowicz, Kosowski, Kranakis et Taleb [30] sur le patrouillage des arbres par k agents qui possèdent la même vitesse de déplacement. Les auteurs ont considéré les deux environnements suivants. Le premier est l'arbre géométrique dans lequel les agents peuvent changer la direction de leurs déplacements dans les nœuds et à l'intérieur des arêtes à n'importe quel moment. Le deuxième est l'arbre dans le contexte de la théorie des graphes et dans lequel les agents peuvent changer la direction de leurs déplacements uniquement dans les nœuds, c'est à dire si un agent quitte un nœud u vers un nœud voisin v alors il ne peut retourner à u qu'après avoir visité v. La longueur d'une arête dans l'arbre géométrique est représentée par un nombre réel tandis que dans l'autre modèle d'arbre c'est un nombre entier positif. Les auteurs ont prouvé que le patrouillage d'un arbre T (géométrique ou non) peut se faire en une oisiveté $\frac{2|T|}{k}$, où $|T|$ est la somme de la longueur de tous les arêtes de T. Leur importante contribution dans cette étude est qu'ils ont prouvé qu'il n'existe aucun algorithme de patrouillage d'arbre qui permet de réaliser une oisiveté meilleure que $\frac{2|T|}{k}$, ce qui prouve l'optimalité de leur solution.

2.3 Le rendez-vous

À travers cette section, nous allons parcourir quelques articles de recherche qui ont traité le problème du rendez-vous des agents mobiles. Comme son nom l'indique, le

rendez-vous est une rencontre entre deux ou plusieurs agents dans un certain environnement. Dans la littérature, lorsque le nombre d'agents qui doivent se rencontrer dépasse deux, on parle d'un rassemblement d'agents mobiles. Nous avons divisé cette section en deux parties : le rendez-vous synchrone et le rendez-vous asynchrone.

2.3.1 Le rendez-vous synchrone

Nous considérons le rendez-vous dans les réseaux modélisés par des graphes connexes et non-orientés. Un rendez-vous synchrone est une rencontre qui se fait de la manière suivante. Les agents se déplacent en rondes synchrones. À chaque ronde, un agent doit rester immobile sur son nœud actuel ou se déplacer vers un nœud voisin. On suppose que le rendez-vous synchrone ne peut se réaliser que dans un nœud. La rencontre dans une arête du graphe est impossible. Un des rendez-vous difficile à réaliser est le rendez-vous synchrone déterministe à cause de la symétrie. Pour mieux comprendre ce problème, regardons la situation suivante : Dans un graphe composé de deux nœuds liés par une arête, on met un agent sur un nœud et un autre agent identique sur l'autre nœud. Les deux agents commencent leurs déplacements à la même ronde. Nous pouvons voir sans aucun doute que le rendez-vous ne va jamais se faire parce que les agents font la même chose à chaque ronde et ils vont toujours se trouver dans des positions différentes. Donc pour résoudre ce problème, il faut briser la symétrie. Dans la littérature, on trouve différentes manières pour casser la symétrie [72] :

- Attribuer à chaque agent une étiquette différente. Cette étiquette joue le rôle d'un identifiant qui permet de distinguer les agents. Un agent connaît seulement sa propre étiquette.
- Permettre aux agents de marquer des nœuds à l'aide des jetons ou écrire de l'information dans des tableaux blancs. Les jetons peuvent soit être stationnaires

et il est impossible de les déplacer une fois déposés dans des nœuds soit ils peuvent être mobiles et les agents ont la possibilité de les enlever et de les mettre ailleurs. Un tableau blanc est un espace initialement vide qui se trouve sur un nœud et sur lequel un agent peut écrire de l'information, la lire ou l'effacer.

– Trouver une caractéristique singulière de l'environnement comme un nœud central dans un arbre qui peut être utilisé pour faire le rendez-vous même dans le cas où cet arbre est symétrique[98].

Après cette introduction, nous commençons par un article de référence pour le rendez-vous synchrone et déterministe dans un graphe. Dans cet article, Dessmark, Fraigniaud, Kowalski et Pelc [41] ont présenté la première solution complète pour ce genre de problème. Dans un réseau modélisé par un graphe anonyme et connexe, deux agents mobiles veulent se rencontrer. Un graphe anonyme est un graphe dans lequel les nœuds ne sont pas identifiables, c'est à dire si un agent visite deux nœuds différents qui ont le même degré, alors pour lui ces nœuds sont identiques. Chaque agent possède un identifiant, qu'on appelle étiquette, unique et se déplace dans le réseau d'une manière synchrone. Chacun des agents connaît sa propre étiquette sans avoir aucune information sur celle de l'autre agent. Les nœuds possèdent des ports numérotés de $\{0, 1, ..., d-1\}$, où d est le degré du nœud et il n'existe aucune relation entre les numéros de ports associés à deux nœuds différents. Ces auteurs [41] ont fourni un algorithme déterministe qui permet de de résoudre le problème du rendez-vous pour n'importe quel graphe en temps $O(n^5\sqrt{\tau \log l}\log n + n^{10}\log^2 n \log l)$ où n est le nombre des nœuds du graphe, D est la distance entre les positions initiales des agents, l est la plus petite étiquette et τ est la différence de temps de départ des deux agents. Dans ce même article, les auteurs ont posé la question s'il y aurait une possibilité de trouver un algorithme déterministe qui permettrait de réaliser le rendez-vous en un temps indépendant de τ.Une réponse positive est venue deux ans plus tard par Kowalski et Malinowski [71]. Ces deux cher-

cheurs ont réalisé un algorithme qui fonctionne en temps $O(\log^3 l + n^{15} \log^{12} n \log l)$. Une amélioration importante de ce temps a été faite par Ta-Shma et Zwick [94]. Ces auteurs ont utilisé le résultat de Reingold[91] qui a prouvé que le calcul de la *séquence d'exploration universelle* proposée par Kouckỳ[70] pour n'importe quel graphe connexe d'ordre n se fait en temps polynomial en n. À l'aide de ce résultat, ils ont réussi à fournir un algorithme déterministe qui permet à deux agents de faire le rendez-vous en temps $O(n^5 \log l.f(\log n) + g(\log n))$ où f et g sont des polynômes. Jusqu'à date, ce temps est resté imbattable.

Dieudonné et Pelc [44] ont présenté une solution pour le rendez-vous synchrone et déterministe pour un nombre d'agents supérieur à 2. Ce problème est connu dans la littérature sous le nom du « rassemblement ». Plusieurs agents se trouvent initialement dans des nœuds différents dans un graphe anonyme. Les auteurs ont gardé le même modèle que celui cité dans l'article [41] à l'exception que les agents sont anonymes, c'est à dire qu'ils n'ont pas d'étiquettes. Dans ce cas, ils ne peuvent pas briser la symétrie en utilisant leurs propres étiquettes. Cette contrainte rend la réalisation du rendez-vous très difficile. Dans leur article [44], Dieudonné et Pelc ont traité le problème du rassemblement dans deux cas : le premier cas est celui où les agents connaissent une borne supérieure N sur la taille du graphe. Les auteurs ont formulés une condition suffisante et nécessaire sur la configuration initiale des agents qui permet le rendez-vous. Cette condition est formulée à l'aide de la notion de la vue de l'agent [96]. La vue d'un agent à partir d'un nœud v est l'arbre infini enraciné en v, qui représente tous les chemins possibles qui peuvent être parcouru dans le graphe à partir du nœud v. Avec connaissance de N, l'algorithme fourni par les auteurs permet de faire le rassemblement avec détection en temps polynomial en N. Les algorithmes qui traitent le rendez-vous avec détection permettent aux agents d'affirmer que la rencontre de tous les agents a eu lieu. Le deuxième cas que les auteurs ont traité dans leur article [44] est celui où les

agents ne connaissent aucune borne supérieure sur la grandeur du graphe. Ils ont prouvé que si les agents se trouvent dans une configuration initiale qui permet le rassemblement alors ils peuvent se rencontrer sans détection en temps polynomial en la taille du graphe.

Le temps du rendez-vous peut être amélioré lorsqu'on donne aux agents plus d'informations. Collins, Czyzowicz, Gasieniec, Kosowski et Martin[24] se sont intéressés au même problème du rendez-vous cité ci-dessus [41] mais avec quelques modifications. Les deux agents sont anonymes, c'est à dire qu'ils n'ont aucune information sur leurs propres étiquettes, par contre ils connaissent la topologie du graphe et chacun d'eux connaît seulement sa propre position initiale. Ces auteurs ont étudié ce problème lorsque les deux agents font un départ simultané et ils ont fourni un algorithme qui permet de faire le rendez-vous en temps $O(D)$ dans le cas où le graphe est une ligne infinie ou un arbre, où D est la distance initiale qui sépare les deux agents. Pour les graphes arbitraires, ils ont montré que le rendez-vous est réalisable en temps $O(D \log^2 n)$ où n est la taille du graphe. On remarque bien qu'avec ses quelques informations fournies aux agents, le temps du rendez-vous s'est beaucoup amélioré.

Dans l'article [24], le temps du rendez-vous dépend de la distance initiale D entre les agents et de la taille de graphe. Est-il possible d'avoir un temps en fonction de D seulement ? Miller et Pelc [83] ont répondu positivement à cette question en utilisant l'approche du « Rendez-vous avec conseil ». Le *conseil* est une information sous forme de chaîne binaire qui permet aux agents de faire le rendez-vous dans un meilleur temps si c'est possible. Ces auteurs ont réussi à trouver la taille minimale d'informations qu'on peut fournir aux agents pour réaliser le rendez-vous en temps $\Theta(D)$. Ils ont utilisé le même modèle étudié dans [41] en imposant la contrainte que les étiquettes des agents doivent appartenir à un ensemble déterminé $\{1, ..., L\}$. Dans cet article, Miller et Pelc ont réalisé un algorithme qui permet de faire le rendez-vous en temps $\Theta(D)$ en fournissant aux agents une quantité d'informations de taille $O(D \log \frac{n}{D} + \log \log L)$. Ils ont aussi

prouvé que cette taille d'information est optimale pour faire un rendez-vous en temps $\Theta(D)$.

Les chercheurs s'intéressent parfois à des graphes particuliers. Elouasbi et Pelc [53] ont étudié le problème du rendez-vous dans les arbres anonymes en utilisant le même modèle que celui de l'article [44] : les agents sont anonymes. Contrairement à [44], ils ont considéré seulement le rendez-vous de deux agents. Les auteurs ont réussi à fournir un algorithme qui exploite les particularités des arbres et permet de faire le rendez-vous déterministe en temps $O(n)$. Ils ont aussi prouvé qu'il n'existe aucun algorithme qui peut faire mieux que $\Omega(n)$ même si les agents se trouvent initialement à une distance égale à 1.

Kranakis, Krizanc et Markou se sont intéressés au problème du rendez-vous dans les anneaux et leurs études de synthèse [74] à ce sujet s'avère très intéressante. Un des modèles que ces auteurs ont utilisé est le suivant : deux agents anonymes veulent se rencontrer dans un anneau anonyme qui peut être orienté ou non et la distance D qui sépare leurs positions initiales est inférieure strictement à $n/2$ où n est le nombre des nœuds de l'anneau. Nous sommes dans la même situation que celle de l'article [53], les agents ne peuvent pas utilisés leurs étiquettes pour briser la symétrie de l'anneau. C'est pour cela que les auteurs de [74] permettent à chaque agent d'utiliser un jeton stationnaire pour marquer sa position initiale. Ces jetons sont identiques. Voici des résultats sur le temps du rendez-vous selon quelques situations : si les agents connaissent n et D alors le rendez-vous se fait en temps optimal $3n/4$ si l'anneau est orienté. Si l'anneau n'est pas orienté alors le rendez-vous peut se faire en temps $5n/6$ avec une bonne inférieure de $3n/4$. Si les agents ne connaissent ni n ni D alors le rendez-vous se fait en temps optimal $5n/4$. Les auteurs ont aussi prouvé que le rendez-vous est impossible si $D = n/2$ même si les agents utilisent leurs jetons stationnaires (les auteurs ont supposé que n est pair).

Les mêmes auteurs ont étudié le problème du rendez-vous dans un autre graphe particulier qui est le tore orienté. Dans leur article [75], deux agents anonymes et équipés d'un ou plusieurs jetons sont placés dans un tore orienté et anonyme de dimension $n \times m$. Les auteurs n'ont pas seulement cherché à résoudre le problème du rendez-vous mais ils ont regardé si le rendez-vous pouvait se faire avec ou sans détection. Une autre chose qui les a intéressés était de calculer la taille de mémoire nécessaire pour faire le rendez-vous. Différents résultats ont été présentés : si chaque agent possède un jeton stationnaire et une mémoire de taille $O(\log n + \log m)$ alors le rendez-vous avec détection est faisable en temps $O(nm)$. Dans le cas où chaque agent possède 2 jetons mobiles et une mémoire de taille constante, le rendez-vous sans détection peut se faire en temps $O(n^2 + m^2)$. Si chaque agents possède 3 jetons mobiles et une mémoire constante alors les agents peuvent faire le rendez-vous avec détection en temps $O(n^2 + m^2)$. Les auteurs de [75] ont aussi prouvé que si deux agents possèdent un nombre constant de jetons stationnaires alors le rendez-vous ne serait possible que si chacun d'eux possède une mémoire de taille au moins $\Omega(\log n)$ bits. Les agents ont aussi besoin de la même taille de mémoire pour faire le rendez-vous si chacun d'eux possède un seul jeton mobile.

Plusieurs chercheurs ont étudié l'impact de la taille de la mémoire des agents sur le temps de réalisation du rendez-vous. Kranakis, Santoro, Sawchuk et Krizanc [78] se sont intéressés à ce problème dans le modèle suivant : soient deux agents anonymes placés initialement à une distance D dans un anneau anonyme à n nœuds. Chaque agent est équipé d'un jeton stationnaire, sait que $D \leq n/3$ et que le réseau dans lequel il se trouve est un anneau. Par contre, les agents ne connaissent ni n, ni D, ni l'orientation de l'anneau. Ces auteurs ont prouvé que si les agents sont équipés d'une mémoire de taille $O(1)$ bits alors le rendez-vous déterministe peut se faire en temps $\frac{n+D}{2}$. Pour le même modèle, ils ont fourni un algorithme qui permet à deux agents équipés d'une mémoire de taille $\Theta(\log \log n)$ bits de détecter si $D = n/2$ et dans ce cas les agents savent que

le rendez-vous est impossible ; s'ils détectent que $D < n/2$ alors le rendez-vous peut se faire en temps $O(\frac{n \log n}{\log \log n})$.

Flocchini, Kranakis, Krizanc, Santoro et Sawchuk [59] ont étudié le même problème que [78] mais pour un nombre d'agents $k \geq 2$. Ils ont commencé par prouver que le rassemblement est impossible si les agents ne connaissent ni k ni la taille de l'anneau n. Ils ont ensuite prouvé que si les agents connaissent k alors le rassemblement est possible seulement si la séquence des distances entre les jetons placés dans les positions initiales des agents n'est pas périodique. Dans ce cas, si les agents sont équipés d'une mémoire de taille $O(k \log n)$ bits alors le rassemblement se fait en temps $O(n)$. Lorsque la taille de leurs mémoires est diminuée à $O(\log n)$ bits, les agents peuvent faire le rassemblement en temps $O(kn)$ et si elle est diminuée à $O(k \log \log n)$ bits, le rassemblement peut se faire en temps $O(\frac{n \log n}{\log \log n})$. Les auteurs ont aussi démontré qu'il n'existe aucun algorithme déterministe qui permet de faire le rassemblement dans un tel modèle pour k agents tel que $k > 2$ avec des agents équipés d'une mémoire de taille inférieure à $\Theta(\log \log n + \log k)$ bits.

Tandis que les auteurs de [59] se sont intéressés seulement au cas où la configuration des agents dans leurs positions initiales n'est pas périodique, Gasieniec, Kranakis, Krizanc et Zhang [63] ont étudié le même modèle pour définir la taille de mémoire nécessaire pour chaque agent, non seulement pour faire le rassemblement mais pour aussi permettre aux agents de savoir si le rassemblement est faisable ou non. Pour cela, les auteurs ont défini deux autres configurations initiales possibles des agents : une configuration est dite symétrique si la distance entre les positions initiales des agents est uniforme et elle est dite périodique si la séquence des distances entre les positions initiales des agents est cyclique. Dans ces deux configurations, le rassemblement est impossible. Les auteurs de [63] ont fourni en premier lieu un algorithme déterministe qui permet à chaque agent équipé d'une mémoire de taille $O(\log k)$ bits de savoir si les agents se trouvent dans

une configuration initiale périodique ou non (cet algorithme ne permet pas de savoir si la configuration est symétrique et dans un tel cas l'exécution de cet algorithme ne va jamais s'arrêter) ; si la configuration n'est pas périodique alors le rassemblement peut se faire en temps $O(kn^2)$. Ils ont ensuite fourni un autre algorithme déterministe qui permet à chaque agent équipé d'une mémoire de taille $\Theta(\log\log n + \log k)$ bits de savoir si les agents se trouvent dans une configuration initiale périodique ou symétrique ; le cas échéant, le rassemblement peut se faire en temps $O(\frac{n \log n}{\log\log n})$.

Fraigniaud et Pelc [61] ont traité la taille de mémoire nécessaire pour deux agents anonymes pour faire le rendez-vous dans un arbre anonyme. Chaque nœud possède des ports numérotés de $\{0, 1, ..., d-1\}$ où d est le degré du nœud. Les auteurs ont montré que pour faire le rendez-vous déterministe, les agents doivent être équipés d'une mémoire de taille d'au moins $\Omega(\log n)$ bits pour un départ arbitraire où n est la grandeur de l'arbre. Ils ont aussi prouvé que cette taille de mémoire est aussi nécessaire pour faire le rendez-vous dans une ligne de longueur n. Dans le cas d'un départ simultané, les auteurs ont fourni une solution qui permet aux agents de faire le rendez-vous déterministe s'ils possèdent une mémoire de taille $\Theta(\log l + \log\log n)$ bits où l est le nombre de feuilles de l'arbre.

Pour les graphes arbitraires, Czyzowicz, Kosowski et Pelc [31] ont prouvé le résultat suivant : Soient deux agents anonymes placés dans un graphe anonyme de grandeur n. Chaque nœud possède des ports numérotés de $\{0, 1, ..., d-1\}$ où d est le degré du nœud. Les agents peuvent faire le rendez-vous déterministe avec une mémoire de taille $O(\log n)$ bits quel que soit leur départ (simultané ou arbitraire). Une mémoire de taille $\Omega(\log n)$ est nécessaire même pour un départ simultané.

Czyzowicz, Kosowski et Pelc [32] ont repris le même modèle que [61] pour étudier l'impact de la taille de la mémoire sur le temps du rendez-vous déterministe. Ils ont prouvé que si les agents possèdent une mémoire de taille k alors le rendez-vous peut-être

fait, quand c'est possible, en temps $\Theta(n+n^2/k)$ même dans le cas dun départ arbitraire. Ils ont aussi montré que les agents ont besoin d'une mémoire de taille logarithmique en n, même s'ils se trouvent sur une ligne de longueur n dans le cas d'un départ simultané. Dans cet article [32], les auteurs ont réussi à trouver un « compromis »entre la taille de la mémoire des agents et le temps du rendez-vous.

Un autre type de compromis est celui qui a été étudié par Miller et Pelc [85] et qui touche la relation entre le temps et le coût du rendez-vous. Le temps est calculé par le nombre de rondes synchrones utilisé par chaque agent jusqu'à la réalisation du rendez-vous tandis que le coût est calculé par le nombre de fois que les arêtes du graphe ont été traversées pour réaliser le rendez-vous. Les auteurs ont utilisé le même modèle que [32] mais cette fois-ci avec des des agents qui possèdent des étiquettes différentes de l'ensemble $\{1, ..., L\}$. Ils ont fourni deux solutions : La première solution est un algorithme déterministe qui fonctionne en temps $O(EL)$ et dont le coût appartient à $O(E)$ où E est une borne supérieure sur le temps d'exploration du graphe qui est connue par les deux agents. La deuxième solution est un algorithme déterministe qui fonctionne en temps $O(E \log L)$ et dont le coût appartient à $O(E \log L)$. Les auteurs ont aussi prouvé que les deux solutions qu'ils ont proposées sont optimales, c'est à dire que n'importe quel algorithme déterministe qui permet de faire le rendez-vous en temps $O(E \log L)$ coûte au moins $\Omega(E \log L)$ et n'importe quel algorithme déterministe qui permet de faire le rendez-vous avec un coût en $E + o(E)$ doit le faire en un temps $\Omega(EL)$.

Les mêmes auteurs se sont intéressés à trouver le compromis entre le coût pour réaliser le rendez-vous et la taille minimale du conseil fourni aux agents pour le faire. Dans l'article qui traite ce problème [84], Miller et Pelc utilise un autre modèle : Deux agents anonymes se trouvent dans un graphe connexe qui possède des nœuds identifiés avec des étiquettes distinctes. Pour les arbres, les auteurs ont prouvé que les agents peuvent se rencontrer avec un conseil de taille $O(D \log(\frac{e}{C}) + \log \log e)$, où C est le coût

de la réalisation du rendez-vous, D est la distance initiale qui sépare les agents et e le nombre d'arêtes de l'arbre. Pour les graphes arbitraires, les auteurs ont prouvé que les agents peuvent se rencontrer avec un conseil de taille $O(D\log(D.\frac{e}{C}) + \log\log e)$. Les auteurs ont aussi montré que pour ce genre de problème, les agents ont besoin d'un conseil de taille au moins $\Omega(D\log(\frac{e}{C}))$ pour réaliser le rendez-vous à un coût C même dans la classe des arbres.

À la fin de cette partie, nous allons parler un peu d'un autre aspect important pour résoudre le problème du rendez-vous synchrone ; c'est l'utilisation des algorithmes aléatoires. Dans la littérature, on parle du rendez-vous aléatoire lorsque ce genre d'algorithmes est utilisé. L'ingrédient aléatoire aide beaucoup à résoudre des problémes qui bloquent parfois les chercheurs dans le cas déterministe. Prenons l'exemple suivant : deux agents anonymes se trouvent dans un graphe anonyme à deux nœuds et ils désirent se rencontrer. Dans le cas déterministe, le rendez-vous à départ simultané est impossible parce que les agents vont toujours faire la même chose et ils ne se retrouveront jamais dans le même nœud. Supposons maintenant que chaque agent possède une pièce de monnaie qu'il utilise pour bouger ; si c'est *pile* alors il reste immobile sur sa position actuelle et si c'est *face* alors il se déplace vers l'autre nœud. Avec une probabilité 1/2, un agent va avoir *pile* et l'autre aura *face* et le rendez-vous sera réalisable.

L'exemple qu'on vient de voir explique comment le lancer de la monnaie, introduit par Alpern [4] dans le contexte de rendez-vous, aide à débloquer des situations complexes. Un autre principe qui est très utilisé pour résoudre le problème du rendez-vous aléatoire est celui de la marche aléatoire (du mot anglais *Random Walk*). C'est le mathématicien Karl Pearson [87] qui a introduit ce principe en statistique. Quelques années après, le mathématicien George Pólya [3] lui a donné le nom de la marche aléatoire pour la première fois. Le livre de Doyle et Snell [51] s'avère une référence complète sur l'utilisation de cette marche dans la théorie des graphes.

La marche aléatoire [14] dans un graphe G non orienté et connexe consiste à parcourir G en utilisant une distribution de probabilité qui se calcule de la manière suivante. L'agent qui se trouve dans un nœud u de degré $d(u)$ reste dans ce nœud avec une probabilité $\frac{1}{d(u)+1}$ et se déplace vers chaque nœud voisin de u avec une probabilité $\frac{1}{d(u)+1}$.

Les modèles que nous allons présenter sont des graphes anonymes mais les agents peuvent lire les numéros de ports associés à chaque nœud. Lorsqu'on utilise des algorithmes aléatoires, on ne parle pas du temps d'exécution mais plutôt du « temps espéré »d'exécution mais dans ce qui suit nous allons utiliser seulement le terme « temps »pour ne pas alourdir le texte.

Kranakis and Krizanc [73] ont étudié le problème du rendez-vous aléatoire dans un anneau anonyme de taille n. Deux agents anonymes sont placés dans deux positions initiales à une distance d et ils sont équipés d'une mémoire de taille constante. Les agents doivent se rencontrer dans un même nœud. Les auteurs ont construit un algorithme qui utilise la marche aléatoire et qui permet aux agents de faire le rendez-vous en temps $\frac{d}{2}(n-d)$ lorsque $d \leq \frac{n}{2}$ et d et n sont paires. Si $d = \Theta(n)$ alors le rendez-vous se fera en temps quadratique en n. Afin d'améliorer le temps du rendez-vous, les auteurs ont fourni un autre algorithme aléatoire qui utilise le lancer de la monnaie. Ils ont montré qu'avec une mémoire de taille $O(\log n)$, les agents font le rendez-vous en temps $\Theta(n)$. Une amélioration de la taille de la mémoire a été réalisée par Kranakis, Krizanc et Morin [76] dans la classe des anneaux orientés. Les auteurs ont montré que si les agents possèdent une mémoire de taille $O(\log \log n)$ alors le rendez-vous aléatoire se fait en temps $\Theta(n)$.

Nous terminons cette partie par une étude comparative faite par Elouasbi et Pelc [53] sur le rendez-vous déterministe et aléatoire. Ils ont considéré deux agents anonymes situés dans deux positions initiales différentes d'un arbre anonyme non-orienté. Les agents n'ont aucune connaissance sur la taille de l'arbre, ils peuvent voir les numéros de ports

de chaque nœud qu'ils visitent et ils se déplacent en rondes synchrones. Nous avons déjà vu que ces auteurs ont prouvé que le rendez-vous déterministe se fait en temps $\Theta(n)$, où n est la taille de l'arbre. Il faut mentionner que dans ce modèle, le rendez-vous n'est réalisable que si les deux agents se trouvent dans des positions initiales qui ne sont pas symétriques ou qu'elles sont symétriques mais les agents commencent l'exécution de l'algorithme déterministe à deux instants différents. Dans le cas aléatoire, les auteurs ont prouvé que le rendez-vous se fait en temps $\Theta(n)$ avec une probabilité de succès d'au moins $1 - \frac{1}{n}$ et ceci pour n'importe quelle configuration des positions initiales des agents. De plus, ils ont montré que si les agents connaissent à l'avance n, la distance d qui sépare leurs positions initiales et le degré maximum de l'arbre Δ alors le rendez-vous aléatoire peut se faire en temps $O(\log n)$ avec une probabilité de succès de $1 - \frac{1}{n}$.

2.3.2 Le rendez-vous asynchrone

Dans cette partie, nous allons voir une autre manière du mouvement des agents pour faire le rendez-vous ou le rassemblement. C'est le déplacement asynchrone. Dans la littérature, il existe plusieurs modèles asynchrones mais nous allons nous contenter de présenter seulement deux. Il faut aussi mentionner que le rendez-vous asynchrone est plus difficile à résoudre mais c'est le modèle qui est le plus proche de la réalité. Nous allons présenter le premier modèle asynchrone pour le rendez-vous dans le plan et le deuxième modèle pour le rendez-vous dans le réseau.

Cieliebak, Flocchini, Prencipe et Santoro [22] se sont intéressés à résoudre le problème du rassemblement asynchrone dans le plan. Ils ont utilisé le modèle suivant. Au moins 5 robots anonymes veulent se rencontrer dans un point du plan cartésien. Ils peuvent se déplacer dans n'importe quelle direction et ils ne possèdent aucun système d'orientation commun (par exemple une boussole). Ces robots ne peuvent pas communiquer entre eux

et leurs mémoires servent seulement à faire des opérations instantanées, c'est à dire que les robots ne se rappellent pas des anciennes opérations. Les robots performent leurs déplacements en cycles *Regarde-Calcule-Bouge*. Durant la phase *Regarde*, un robot a une vue globale de tout le plan. Il peut voir les points du plan qui contiennent un ou plusieurs robots et ceux qui ne contiennent aucun. Ce principe est connu dans la littérature sous le nom de « détection avec multiplicité »et dans certains modèles, les robots ne possèdent pas cette capacité. À la fin de cette phase, le robot aura toutes les informations nécessaires pour prendre une décision de déplacement qu'il doit faire durant la phase *Calcule*. Cette dernière permet au robot de faire des calculs en utilisant son algorithme pour décider vers quel point il va se diriger ou s'il va rester immobile sur sa position actuelle. Enfin durant la phase *Bouge*, le robot se dirige vers sa destination. Les robots exécutent leurs cycles et se déplacent de manière *asynchrone*, c'est à dire que leurs calculs et leurs mouvements prennent un temps fini, mais imprévisible contrairement au modèle des rondes synchrones. Par exemple, si deux robots veulent rejoindre au même moment un point du plan qui se trouve à la même distance de leurs positions actuelles alors il n'y a aucune garantie qu'ils vont arriver à leur destination au même instant à cause de leur déplacement asynchrone. Les auteurs ont fourni un algorithme déterministe qui permet de faire le rassemblement asynchrone dans le plan pour k robots anonymes tel que $k > 2$.

Flocchini, Prencipe, Santoro et Widmayer [60] ont repris le même modèle que celui des auteurs précédents mais avec deux modifications. La première est que tous les robots possèdent le même sens d'orientation et la deuxième c'est qu'ils sont capables de voir à une distance uniforme D. Les auteurs ont fourni un algorithme déterministe qui permet aux robots de faire le rassemblement asynchrone dans un temps fini.

Chatterjee, Chaudhuri et Mukhopadhyaya [20] se sont intéressés au problème étudié par les auteurs de l'article [60] mais en supposant que le champ de vision n'est pas

le même pour tous les robots. Ils ont prouvé que si les robots connaissent une borne inférieure sur leurs champs de vision alors le rassemblement est possible en un temps fini.

Nous reprenons le même modèle de l'article [22] mais cette fois-ci avec seulement deux robots qui possèdent le même sens d'orientation et qui utilisent des boussoles pour choisir leurs directions. Le problème est que, à cause des champs magnétique, il y a une probabilité que les boussoles ne seront pas fiables à cent pour cent. C'est le problème que Izumi, Souissi, Katayama, Inuzuka, Défago, Wada et Yamashita [68] ont essayé de résoudre. Ils ont défini deux types de boussoles : une boussole *statique* qui garde une marge d'erreur constante durant l'exécution de l'algorithme et une autre dite *dynamique* dont la marge d'erreur change avant et après chaque exécution des cycles *Regarde-Calcule-Bouge*. La marge d'erreur est calculée par la taille de l'angle Φ formé par l'axe des x de la boussole et la direction de référence du système de coordonnées global. Les auteurs ont montré que le rendez-vous déterministe est réalisable lorsque $\Phi < \pi/2$ pour les boussoles statiques et $\Phi < \pi/6$ pour celles qui sont dynamiques.

Dans les modèles que nous avons vus jusqu'à maintenant, les robots sont considérés comme des points dans le plan. Une autre considération qui n'est pas très réalisable est que dans le principe du *Regarde-Calcule-Bouge*, les chercheurs supposent aussi dans ces modèles qu'un robot peut voir à travers un autre même s'il se trouve sur sa ligne de vision. Czyzowicz, Gasieniec et Pelc [28] étaient les premiers à étudier le problème du rassemblement déterministe asynchrone pour des robots qui ne sont pas considérés comme des points dans le plan mais plutôt comme des disques identiques de rayon 1. Dans ce cas, les robots ne sont pas transparents et ne possèdent pas de GPS pour s'orienter. Les auteurs ont fourni un algorithme déterministe qui permet de faire le rassemblement asynchrone pour au plus 4 robots et ils ont posé un problème ouvert si ce rassemblement est possible dans le cas d'un nombre arbitraire de robots.

Agathangelou, Georgiou et Mavronicolas [2] ont répondu positivement à cette question. Ils ont pris le même modèle mais en supposant que chaque robot connaît le nombre total des robots, possède une vision totale de tout le plan et connaît lorientation des axes du plan cartésien. Les auteurs ont réussi à construire un algorithme déterministe qui permet de faire le rassemblement asynchrone pour un nombre arbitraire k de robots.

Nous allons maintenant présenter le deuxième modèle asynchrone dans les réseaux. De Marco, Gargano, Kranakis, Krizanc, Pelc et Vaccaro [82] étaient les premiers à définir ce modèle. Deux agents se trouvent dans un graphe connexe et non-orienté. Ils possèdent deux étiquettes différentes et chaque agent connaît uniquement sa propre étiquette. Le graphe est anonyme mais les agents peuvent percevoir les numéros de ports associés à chaque nœud. Les agents se déplacent de façon asynchrone. Un agent choisit un numéro de port sur sa position actuelle pour traverser l'arête vers un nœud voisin. Ce parcours ce fait à une vitesse arbitraire pas nécessairement constante qui est contrôlée par un adversaire. Ce dernier a le pouvoir de forcer les agents à se déplacer rapidement ou lentement afin de compliquer la réalisation du rendez-vous. Suite à cette contrainte, ces chercheurs ont considéré que le rendez-vous peut se faire aussi à l'intérieur d'une arête contrairement au rendez-vous synchrone qui ne peut se faire que dans un nœud. Le rendez-vous asynchrone sera accompli lorsque les agents vont se rencontrer au même instant à la même place dans un nœud ou à l'intérieur d'une arête. Puisque les agents n'ont pas d'horloge pour synchroniser leurs mouvements, le coût du rendez-vous est le nombre total d'arêtes parcourues par les agents.

Les auteurs de [82] ont commencé par résoudre le problème du rendez-vous asynchrone dans une ligne infinie.$|L|$ dénote la longueur de l'étiquette L. Si L_{max} et L_{min} sont les étiquettes des deux agents tels que $|L_{max}| < |L_{min}|$ et si D est la distance entre leurs positions initiales alors le rendez-vous peut se faire à un coût en $O(D|L_{min}|^2)$ si les agents connaissent D et en $O((D + |L_{max}|)^3)$ dans le cas contraire. Ce dernier

coût a été amélioré par Stachowiak [93] qui a fourni un algorithme déterministe qui permet de faire le rendez-vous à un coût en $O(D \log^2 D + D \log D|L_{max}| + D|L_{min}|^2 + |L_{max}||L_{min}| \log |L_{min}|)$. Ensuite, les auteurs ont traité le cas de l'anneau. Ils ont proposé un algorithme déterministe qui permet de faire le rendez-vous à un coût en $O(n|L_{max}|)$ quand les agents ne connaissent pas la taille n de l'anneau. Dans le cas où les agents connaissent n, les auteurs ont prouvé que le rendez-vous peut se faire à un coût en $\Theta(n|L_{min}|)$. Enfin, les auteurs ont proposé un algorithme déterministe qui permet de faire le rendez-vous à un coût en $\Theta(D|L_{min}|)$ lorsque les agents connaissent n et D.

Quelques années plus tard, Czyzowicz, Labourel et Pelc [34] ont fourni une solution pour résoudre le problème du rendez-vous déterministe asynchrone dans n'importe quel graphe connexe sans la connaissance d'une borne supérieure sur sa taille. Malgré que ce résultat est très important, les auteurs n'ont pas donné le coût de la réalisation du rendez-vous. Ils ont mentionné seulement que ce coût est exponentiel.

Dieudonné, Pelc et Villain [46] ont réussi de leur part de prouver que le rendez-vous déterministe asynchrone dans les graphes arbitraires peut se faire en temps polynomial en la taille du graphe et en la longueur de l'étiquette la plus petite des deux agents mais le coût optimal du rendez-vous asynchrone reste toujours un problème ouvert.

Guilbault et Pelc [66] ont repris le modèle précédent mais cette fois-ci avec des agents anonymes, ce qui complique la situation parce que les étiquettes sont très importantes pour briser la symétrie. Les auteurs ont prouvé que le rendez-vous déterministe est faisable exactement dans les deux situations suivantes. Soit que les vues [96] des deux agents à partir de leurs positions initiales sont différentes, soit que les positions initiales sont connectées par un chemin dont les numéros de ports forme un palindrome.

Jusqu'à maintenant nous avons vu la résolution du rendez-vous asynchrone dans le cas déterministe. L'ingrédient aléatoire peut-il apporté des améliorations comme dans le cas du synchrone ? La réponse est positive. Nous avons vu que la solution déterministe de

Guilbault et Pelc [66] était limitée par la configuration des positions initiales des agents mais ces auteurs ont réussi à éliminer cette contrainte en fournissant un algorithme aléatoire qui permet de faire le rendez-vous asynchrone avec une probabilité 1 dans n'importe quel graphe connexe, pour n'importe quelles positions initiales.

Ooshita, Kawai, Kakugawa et Masuzawa [86] se sont intéressés au rassemblement aléatoire asynchrone de $k > 2$ agents dans les anneaux orientés. Les agents ne possèdent aucune information sur la taille de l'anneau et ne connaissent pas k. L'anneau est anonyme mais chaque nœud possède un tableau blanc que les agents peuvent utiliser pour lire et écrire leurs messages. Les auteurs ont étudié la possibilité du rassemblement aléatoire asynchrone avec ou sans détection avec une certaine probabilité p. Ils ont prouvé que le rassemblement aléatoire avec détection est impossible pour une probabilité $0 < p \leq 1$. Pour le rassemblement aléatoire sans détection, il est impossible avec une probabilité $p = 1$ mais il est possible avec une probabilité $0 < p < 1$. Les auteurs ont prouvé que lorsque le rassemblement est possible, le temps du rassemblement est en $O(n)$ si les agents sont équipés d'une mémoire de taille $O(n)$ bits et si la taille de chaque tableau blanc est $O(\log n)$ bits.

Nous terminons cette partie par un modèle asynchrone qui est proche de celui que nous venons de voir mais avec une petite variante. C'est le modèle de l'*approche* dans le plan. La situation est la suivante. Deux agents veulent se rencontrer dans un plan cartésien, mais cette rencontre a une particularité : les agents ne sont pas obligés de se rencontrer dans un même point du plan mais ils peuvent se trouver à une distance constante qui dépend de leur champ de vision. Chaque agent connaît seulement sa propre étiquette. Le déplacement des agents se fait de manière asynchrone comme celle que nous venons de voir pour les réseaux. Bouchard, Bournat, Dieudonné, Dubois et Petit [11] se sont intéressés à ce problème. L'idée qu'ils ont utilisée était de réduire le problème sur le plan à celui de la grille. Les auteurs ont prouvé que l'approche asynchrone peut se

faire en coût polynomial en la distance qui sépare les position initiales des agents et en la taille de la représentation binaire de l'étiquette la plus petite.

2.4 Le rendez-vous et la tolérance aux pannes

Nous avons vu dans la section précédente différents modèles étudiés pour résoudre le problème du rendez-vous. Nous avons aussi parlé de l'impact de l'information fournie aux agents et de la taille de leur mémoire (la taille zéro à la taille illimitée) sur la réalisation et l'optimisation du temps du rendez-vous. Puisqu'on ne vit pas dans un monde parfait, il n'existe aucun système qui est infaillible à 100%. Parmi les choses qui rendent un système vulnérable sont les pannes et nous allons voir dans cette section l'impact de différents types de pannes sur le rendez-vous.

Dobrev, Flocchini, Prencipe et Santoro [48] se sont intéressés à la réalisation du rendez-vous en présence des trous noirs dans les réseaux. Un trou noir est un nœud défectueux qui détruit tout agent qui le visite sans laisser de trace [50]. Souvent, avant de faire le rendez-vous, il faut trouver l'emplacement du trou noir qui peut empêcher la rencontre. Ces mêmes auteurs ont traité dans d'autres articles [47, 49, 50] le nombre nécessaire d'agents pour localiser un trou noir dans différentes topologies de réseaux et particulièrement dans les anneaux [50]. Dans l'article [48], les auteurs ont étudié la situation suivante : k agents anonymes ($k > 1$) se trouvent dans un anneau à n nœuds ; ils se déplacent de manière asynchrone et ils peuvent lire et écrire sur le tableau blanc (*whiteboard*) de chaque nœud. Un des nœuds représente un trou noir qui est inconnu pour les agents. Les auteurs ont prouvé en premier lieu que le rassemblement de k agents est impossible, de plus si l'anneau n'est pas orienté alors le rassemblement de $k-1$ agents n'est pas réalisable. Les auteurs ont aussi démontré que la connaissance de k ou de n (pas nécessairement les deux) par les agents est nécessaire pour réaliser le rassemblement. Si

l'anneau est orienté et les agents connaissent seulement k alors le rassemblement peut se faire avec $k-1$ agents en un temps au plus $3(n-2)$; si l'anneau n'est pas orienté alors le rassemblement peut se faire avec $k-2$ agents si k est impair et avec $\frac{k-2}{2}$ agents si k est pair et en un temps au plus $5(n-2)$. Si l'anneau est orienté et les agents connaissent seulement n alors le rassemblement peut se faire avec $k-2$ agents pour $k>3$ en un temps au plus $8(n-2)$; si l'anneau n'est pas orienté alors le rassemblement peut se faire avec $k-2$ agents si k est impair ou si n est pair et avec $\frac{k-2}{2}$ agents si k est pair et n est impair ; le temps du rassemblement sera au plus $5(n-2)$. Ces auteurs se sont aussi intéressés au rassemblement à une distance égale à 1 dans les anneaux non-orientés, c'est à dire que les agents n'ont pas besoin d'être dans le même nœud pour se rencontrer mais le rassemblement est aussi réalisé s'ils se trouvent à une distance égale à 1 chacun par rapport aux autres. Pour ce genre de rassemblement, les auteurs ont montré que si les agents connaissent seulement k alors le rassemblement peut se faire avec $k-2$ agents en un temps au plus $5(n-2)$; dans le cas où ils connaissent seulement n alors le rassemblement peut se faire avec $k-2$ agents mais en un temps au plus $8(n-2)$.

Un autre type de pannes est celui de la perte des jetons, c'est à dire que ces derniers ne sont plus visibles pour les agents. Flocchini, Kranakis, Krizanc, Luccio, Santoro et Sawchuk [58] ont étudié le problème du rassemblement pour k agents ($k \geq 2$) équipés chacun d'un seul jeton stationnaire. les agents se trouvent dans des positions différentes d'un anneau non-orienté à n nœuds. Les auteurs ont prouvé que le rassemblement est impossible si les agents ne connaissent ni n ni k ; il est aussi impossible si tous les jetons tombent en panne ou si le $pgcd(m,n) \neq 1$ pour $m \leq k$. Si au plus $k-1$ jetons tombent en panne, les agents connaissent soit k soit n et le $pgcd(m,n) \neq 1$ pour $m \leq k$ alors le rassemblement de k agents est réalisable.

Chalopin, Das et Santoro [17] ont traité les pannes de liens dans les graphes arbitraires et non-orientés. On parle des pannes de liens lorsqu'il existe quelques arêtes du

graphe qui sont défectueuses. Si un agent utilise une arête défectueuse alors il sera détruit et aucune trace ne sera laissée. Le but de cet article est de trouver le nombre minimum d'agents qui peuvent être détruits et permettre au reste des agents qui sont toujours présents de faire le rassemblement. Les auteurs ont considéré k agents anonymes placés dans des nœuds différents d'un graphe anonyme. Chaque nœud possède un tableau blanc et des numéros de ports de 1 à d où d est le degré du nœud. Les agents se déplacent de manière asynchrone et ils peuvent lire et écrire sur les tableaux blancs. Les auteurs ont aussi imposé que l'accès à un tableau blanc ne peut pas se faire par deux agents au même moment. Dans un premier lieu, les auteurs ont prouvé que si le graphe contient τ arêtes défectueuses, alors le rassemblement est impossible pour $(k - \tau + 1)$ agents et plus. Si le nombre d'agents est $k - \tau$, le rassemblement reste toujours impossible si les agents ne connaissent pas la taille du graphe ou au moins une borne supérieure sur cette taille. Les auteurs ont aussi fourni un algorithme qui permet de faire le rassemblement pour $(k - \tau)$ agents dans un graphe de taille n dans le cas où les agents connaissent une borne supérieure B sur la taille du graphe et telle que $n \leq B < 2n$. Enfin, ils ont montré que l'exécution de leur algorithme se fait en temps $\Theta(m(m + k))$ où m est le nombre total des arêtes du graphe (les bonnes et les défectueuses).

Nous avons vu jusqu'à maintenant des pannes qui touchent le réseau, des nœuds qui sont des trous noirs, des jetons qui se perdent ou des arêtes qui sont défectueuses. Une question qu'on peut aussi se poser est la suivante : est-ce que le rendez-vous est possible en présence des agents qui peuvent tomber en pannes ? Les premiers qui ont répondu à cette question sont Dieudonné, Pelc et Peleg [45]. Voici la situation : Plusieurs agents veulent se rencontrer dans un graphe anonyme, arbitraire et non-orienté. Leurs déplacements se font en rondes synchrones. Chaque nœud possède des ports numérotés de 0 à $d - 1$ où d est le degré du nœud. Chaque agent possède une étiquette unique et il ne possède aucune information sur les étiquettes des autres agents. Si plusieurs

agents se trouvent dans le même nœud à la même ronde, ils peuvent communiquer entre eux pour échanger de l'information ; durant cette ronde, les agents présents dans le même nœud possèdent la même information. Parmi les agents qui veulent faire le rassemblement, il se trouve des agents qui sont en panne ou ce que les auteurs ont nommé des *agents byzantins*. On peut considérer les agents byzantins comme des entités défectueuses qui n'aident pas à faire le rassemblement ou des entités malveillantes qui fournissent de l'information erronée pour faire échouer ou compliquer le rassemblement. Dans leur étude, ces auteurs se sont intéressés à deux types de rassemblement : Le rassemblement en présence d'agents byzantins « forts »qui peuvent n'importe quelles opérations malveillantes pour nuire à la rencontre et ils peuvent aussi changer leurs propres étiquettes et le rassemblement en présence d'agents byzantins « faibles »qui sont capables de faire la même chose que les forts à l'exception de changer leurs propres étiquettes. Les auteurs considèrent aussi qu'il existe au plus f agents byzantins et que ce nombre est connu à l'avance par tous les agents. Ceux qui ne sont pas byzantins sont nommés les « bons agents ». Dans leur article [45], ces auteurs ont répondu à la question suivante : *Quel est le nombre minimum de bons agents qui peuvent faire le rassemblement en présence de f agents byzantins ?* Dans le cas où les agents byzantins sont faibles, si tous les agents connaissent la taille n du graphe alors le rassemblement peut se faire avec n'importe quel nombre de bons agents ; si tous les agents ne connaissent pas n alors il est possible de faire le rassemblement avec au moins $f + 2$ bons agents et il est impossible de le faire avec au plus $f + 1$ bons agents. Dans le cas où les agents byzantins sont forts, si tous les agents connaissent la taille n du graphe alors il est possible de faire le rassemblement avec au moins $2f + 1$ bons agents et il est impossible de le faire avec au plus f bons agents ; si tous les agents ne connaissent pas n alors il est possible de faire le rassemblement avec au moins $4f + 2$ bons agents et il est impossible de le faire avec au plus $f + 1$ bons agents.

Dans l'article [45], les auteurs ont réussi à prouver que dans le cas des agents byzantins faibles, le rassemblement est réalisable par n'importe quel nombre de bons agents si tous les agents connaissent la taille du graphe n et qu'il faut exactement $f + 2$ bons agents pour faire le rassemblement si tous les agents ne connaissent pas n. Dans le cas des agents byzantins forts, ces auteurs ont seulement fourni des bornes supérieures et inférieures sur le nombre de bons agents pour faire le rassemblement. Bouchard, Dieudonné et Ducourthial [12] sont venus ensuite pour résoudre complètement le problème du rassemblement en présence des agents byzantins forts. Il ont prouvé que si tous les agents connaissent n alors il faut exactement $f + 1$ bons agents pour faire le rassemblement. Dans le cas où tous les agents ne connaissent pas n, le nombre exact de $f + 2$ bons agents permet de faire le rassemblement. Il est très intéressant de remarquer que lorsque les agents connaissent n, le type des agents byzantins influence la réalisation du rassemblement selon si les agents byzantins peuvent changer leurs étiquettes ou non. Par contre, cet impact disparaît lorsque n est inconnu.

Dans leur article [45], Dieudonné, Pelc et Peleg ont montré que le temps du rassemblement en présence des agents byzantins faibles est polynomial en la taille du graphe et en la valeur de l'étiquette la plus grande des bons agents, par contre ce temps est exponentiel lorsque les agents byzantins sont forts. À la fin de cet article [45], les auteurs ont posé la question sur la possibilité de trouver un algorithme déterministe qui fonctionne en temps polynomial et qui permet de faire le rassemblement en présence des agents byzantins forts. Bouchard, Dieudonné et Lamani [13] ont répondu positivement à cette question. Dans leur étude, ces auteurs ont apporté quelques modifications au modèle de [45] : Le nombre d'agents peut être supérieure à la taille du graphe, c'est à dire que plusieurs agents peuvent avoir la même position initiale. Les auteurs considèrent l'utilisation de ce qu'ils ont nommé *la connaissance générale* qui représente l'information fournie aux agents pour faire le rassemblement et ils supposent aussi que le nombre des

bons agents est au moins $5f^2 + 6f + 2$. Avec ces contraintes, les auteurs ont construit un algorithme déterministe qui permet de faire le rassemblement en présence des agents byzantins forts pour la classe des graphes arbitraires de taille au plus n en temps polynomial en n et en la taille de l, où l est la plus petite étiquette que possède un bon agent. Ils ont montré que ce temps est réalisable lorsque la connaissance générale fournie aux agents est de taille $O(\log\log\log n)$. Ils ont aussi prouvé que lorsque la connaissance générale fournie aux agents est de taille $o(\log\log\log n)$, il n'existe aucun algorithme déterministe qui permet de faire le rassemblement en présence des agents byzantins forts pour la classe des graphes arbitraires de taille au plus n en temps polynomial en n et en la taille de l.

Un autre type de pannes est celui étudié par Chalopin, Dieudonné, Labourel et Pelc [18, 19] et qui porte le nom des *pannes de retardement*. Si un agent subit une panne de retardement durant une certaine ronde alors il reste immobile dans sa position actuelle même s'il prévoyait se déplacer vers un autre nœud. Les auteurs ont étudié ce genre de pannes dans le modèle suivant. Deux agents mobiles se trouvent dans un un graphe anonyme, arbitraire et non-orienté. Chaque nœud possède des numéros de ports de 0 à $d-1$, où d est le degré du nœud. Les agents ont des étiquettes différentes, ne connaissent pas la taille du graphe et se déplacent de manière synchrone. Chaque agent connaît uniquement sa propre étiquette et possède une mémoire de taille illimitée. Le but est de réaliser un algorithme qui permet aux agents de faire le rendez-vous déterministe en présence des pannes de retardement et de calculer son coût qui est le nombre total de traversées des arêtes. Les auteurs ont commencé leur étude par traiter les pannes de retardement *aléatoires* ; ce sont des pannes que chaque agent peut subir à chaque ronde avec une probabilité constante p telle que $0 < p < 1$. Il faut aussi mentionner que ces pannes de retardement aléatoires sont indépendantes pour chaque ronde et pour chaque agent. Dans cette situation, les auteurs ont fourni un algorithme

déterministe qui permet aux agents de faire le rendez-vous dans n'importe quel graphe de taille n à un coût $c = O(P(n, \log L))$ avec une probabilité d'au moins $1 - e^{-O(c)}$ où P est un polynôme et L est l'étiquette la plus grande des deux agents. L'autre type de retardement que les auteurs ont étudié est celui causé par les pannes *non-bornées* ; ce sont des pannes qui gardent un agent immobile pendant un nombre fini mais arbitraire de rondes consécutives. Les auteurs ont prouvé que dans cette situation, le rendez-vous en présence des pannes de retardement est impossible dans la classe des anneaux et donc aussi impossible dans la classe des graphes arbitraires qui contiennent des cycles (il ne faut pas oublier que les agents ne connaissent pas la taille du graphe). En revanche, ils ont montré que ce genre de rendez-vous est faisable dans la classe des arbres arbitraires avec un coût dans $O(nl)$ où n est la taille de l'arbre et l est la plus petite étiquette des deux agents. Les auteurs ont aussi prouvé que dans le cas des pannes de retardement non-bornées, il n'existe aucun algorithme déterministe qui permet de faire le rendez-vous dans la classe des arbres arbitraires en temps meilleur que $\Omega(l)$, même dans le cas où la taille de l'arbre est 2. Le dernier type de retardement que les auteurs ont traité dans cet article [19] est celui causé par les pannes *bornées* ; ce sont des pannes qui gardent un agent immobile pendant au plus k rondes consécutives ; les agents ne connaissent pas la valeur de k. Dans cette situation de pannes, les auteurs ont construit un algorithme déterministe qui permet de faire le rendez-vous dans n'importe quel graphe de taille n avec un coût en $O((P(n) \cdot k \cdot L)^2)$, où P est un polynôme et L est l'étiquette la plus grande des deux agents.

Das, Luccio et Markou [39] se sont intéressés au problème du rassemblement en présence des *agents malicieux* dans les anneaux. Une simple définition d'un agent malicieux est qu'il est un agent byzantin avec le pouvoir de bloquer un autre agent. Regardons maintenant plus en détail le modèle étudié par ces auteurs [39]. k agents anonymes veulent se rencontrer dans un anneau anonyme. Chaque nœud possède des ports numé-

rotés et il ne peut accueillir qu'au plus un agent à l'état initial. Ces k agents possèdent une mémoire constante et se déplacent de manière asynchrone. Les auteurs les ont appelés les agents *honnêtes*. Un agent honnête exécute les instructions avec toute fidélité ; il peut voir les autres agents qui se trouvent avec lui dans le même nœud mais il ne peut pas communiquer avec eux en leur envoyant des messages ou en les écrivant dans les nœuds. Ces agents ne connaissent ni la taille de l'anneau n ni le nombre k des agents honnêtes. Le but est de faire le rassemblement de tous ces agents dans un nœud du graphe. Mais le problème est qu'ils ne sont pas seuls dans l'anneau. Un autre agent, appelé malicieux, se trouve avec eux dans le réseau et son but est de les empêcher de faire le rassemblement. Cet agent puissant à le pouvoir de se déplacer rapidement dans l'anneau et de connaître à chaque instant les positions de tous les agents honnêtes. Il connaît n, k et comment chaque agent honnête va faire ses déplacements. Un autre pouvoir que l'agent malicieux possède est celui de bloquer les agents honnêtes, c'est à dire si l'agent malicieux se trouve sur un nœud u et un agent honnête A veut se déplacer vers u alors A ne peut pas visiter u. Par contre, l'agent malicieux ne peut ni visiter un nœud occupé par un agent honnête ni dépasser ce dernier lorsqu'il traverse une arête du graphe. L'agent malicieux se déplace aussi de manière asynchrone. Afin d'éviter le cas où l'anneau est fortement symétrique, une situation qui va rendre le rassemblement impossible, les auteurs ont ajouté la contrainte suivante : L'anneau possède un nœud spécial qui est connu par tous les agents (bien sûr l'agent malicieux connaît aussi ce nœud grâce à son pouvoir). Les auteurs ont montré en premier lieu les situations pour lesquelles le rassemblement en présence d'un agent malicieux est impossible. Ces situations sont toutes les configurations qui permettent à un agent malicieux de garder en tout temps au moins deux agents séparés. Ensuite, les auteurs ont étudié la problématique dans un anneau orienté ; cette orientation est connue par tous les agents. Ils ont construit un algorithme déterministe qui permet à k agents honnêtes de faire le rassem-

blement en présence d'un agent malicieux. Dans le cas où l'anneau est non-orienté, les auteurs ont prouvé que ce genre de rassemblement n'est possible que si le nombre k des agents honnêtes est impair. Lorsque le rassemblement en présence d'un agent malicieux est réalisable, il se fait à un coût en $O(kn)$ traversées d'arêtes.

Nous terminons cette section par l'étude réalisée par Das, Focardi, Luccio, Markou, Moro et Squarcina [38] concernant les pannes en présence d'agents malicieux. Ces auteurs ont repris le modèle présenté par les auteurs de l'article [39] mais cette fois-ci le déplacement des agents se fait de manière synchrone. Les auteurs ont ajouté deux contraintes par rapport au modèle asynchrone : La première est que tous les agents honnêtes commencent leurs déplacements à la même ronde et la deuxième est qu'ils peuvent communiquer entre eux lorsqu'ils se trouvent dans le même nœud. Les auteurs ont montré en premier lieu que le rassemblement en présence d'un agent malicieux est impossible dans un anneau non-orienté lorsque n est impair et k est pair. Ils ont ensuite prouvé que si $k > 2$ et si n et k sont impairs alors le rassemblement en présence d'un agent malicieux est réalisable. Il faut remarquer que la synchronie a aidé à résoudre le problème de ce genre de rassemblement pour les anneaux non-orientés contrairement au cas asynchrone où ce rassemblement est impossible [39]. L'algorithme que les auteurs ont fourni permet de faire le rassemblement de k agents honnêtes en présence d'un agent malicieux dans un anneau non-orienté de grandeur n en temps $O(n)$ et avec un coût en $O(kn)$ traversées d'arêtes.

2.5 Le rendez-vous et la communication

Dans cette dernière partie de la revue de la littérature, nous allons parler du rôle que peut jouer la communication dans la réalisation du rendez-vous et dans l'optimisation de

sa performance. Nous nous intéressons exactement à l'information que les agents mobiles peuvent échanger entre eux à l'aide de certains mécanismes de communication.

Un des mécanismes que nous avons déjà cité dans cette revue de la littérature est l'utilisation des jetons. Afin de comprendre comment les jetons peuvent être considérés comme un important moyen de communication, regardons l'exemple cité par Pelc [88] dans son étude de synthèse sur le rendez-vous déterministe dans les réseaux. Soient deux agents anonymes qui se trouvent dans deux positions initiales différentes d'un anneau symétrique et qui ne peuvent ni écrire ni laisser de traces sur les nœuds. Ils commencent à exécuter le même algorithme au même moment. À cause de la symétrie, les agents vont faire la même chose tout au long de leur parcours et ils ne vont jamais se rencontrer. L'utilisation des jetons pour marquer la position initiale d'un agent a été introduite par Baston et Gal [10] dans le contexte de la théorie des jeux et ensuite utilisée dans le problème du rendez-vous. Dans leur article [77], Kranakis, Krizanc et Rajsbaum ont étudié le problème du rendez-vous dans l'anneau pour deux agents anonymes équipés de jetons stationnaires identiques. Chaque agent possède un seul jeton qui peut mettre dans sa position initiale. Si les agents connaissent la taille de l'anneau alors chacun d'eux peut identifier la position initiale de l'autre, une information que les agents ont réussi à échanger grâce aux jetons. Nous pouvons aussi penser à plusieurs scénarios de communication en utilisant les jetons comme par exemple équiper deux agents d'un nombre illimité de jetons qui vont leur permettre d'échanger les valeurs de leurs étiquettes ; si un agent possède une étiquette de valeur 100 alors il serait capable de mettre 100 jetons dans sa position initiale pour la communiquer à l'autre agent. Un autre résultat qui met en valeur l'importance de l'utilisation des jetons pour communiquer entre les agents est celui présenté par Czyzowicz, Dobrev, Kranakis et Krizanc [26] et dans lequel ils montrent que si chacun des deux agents est équipé d'un jeton mobile alors le rendez-vous avec détection est impossible. Par contre, s'ils utilisent deux jetons mobiles alors

le rendez-vous avec détection est possible en temps $\Theta(n^2)$. Les auteurs ont prouvé aussi que si un agent possède $t \geq 3$ jetons mobiles alors le rendez-vous avec détection peut se faire en temps $O(t \cdot n^{\frac{t-1}{t-2}})$. Nous remarquons ici que l'utilisation de plusieurs jetons permet d'avoir plus d'informations telles que les positions initiales et les déplacements des agents, ce qui permet non seulement de réaliser le rendez-vous mais aussi de savoir s'il est fait ou non et d'améliorer son temps.

L'autre mécanisme de communication que les agents peuvent utiliser quand c'est possible est les tableaux blancs (du mot anglais *whiteboards*). C'est un moyen très puissant et plus flexible comparé aux jetons ; chaque nœud du graphe possède son propre tableau blanc que tout agent visitant ce nœud peut utiliser pour écrire, lire ou effacer de l'information. Deux contraintes s'imposent. La première est qu'un tableau blanc a une taille qui peut être très grande ou assez petite, ce qui limitera la taille d'information à écrire. La deuxième est que deux agents ne peuvent pas utiliser le tableau blanc d'un même nœud au même moment, c'est à dire que si un agent A écrit un message sur le tableau blanc d'un nœud alors un autre agent ne peut ni lire ni effacer ce message ou écrire une autre information sur ce tableau avant que l'agent A quitte ce nœud. Barrière, Flocchini, Fraigniaud et Santoro [8, 9] ont utilisé les tableaux blancs pour permettre à des agents anonymes d'échanger des informations qui vont les aider à accomplir le rassemblement dans un graphe anonyme. Chaque agent qui écrit un message sur un tableau blanc doit le signer avec son nom, ce qui va lui permettre de différencier ses messages de ceux écrits par les autres agents ; ceci lui permet aussi de connaître la signature d'un autre agent et être capable d'identifier les messages de ce dernier dans les autres nœuds. De leur part, Balamohan, Flocchini, Miri et Santoro [6] ont utilisé les tableaux blancs pour permettre à deux agents de se rencontrer en un temps optimal dans un anneaux en présence d'un trou noir. Les agents écrivent des messages sur les tableaux pour se partager les tâches. chacun d'eux indique à l'autre dans le message les régions qu'il va

explorer. Tandis que les auteurs précédents ont utilisé les tableaux blancs pour résoudre le problème du trou noir, Tsuchida, Ooshita et Inoue [95] les ont utilisés pour optimiser le temps du rassemblement en présence d'agents byzantins. Les auteurs ont utilisé un type spécial de tableaux blancs pour contourner la situation dans laquelle un agent byzantin peut effacer toute l'information écrite. Chaque agent possède son propre espace sur le tableau pour écrire ses messages et les effacer ; il ne peut pas effacer les messages des autres agents mais il peut lire tous les messages du tableau blanc. Dans sa partie dédiée, un agent peut écrire la valeur de son étiquette et informer les autres de son nœud de départ. Avec ces informations, les auteurs ont réussi à faire le rassemblement en temps $O(f|E|)$, où f est une borne supérieure sur le nombre des agents byzantins et $|E|$ représente le nombre des arêtes du graphe. Ce temps est meilleur que $\tilde{\mathrm{O}}(n^9 L)$ [1], celui trouvé par les auteurs de l'article [45] et dans lequel les agents n'utilisent pas de tableaux blancs.

De leur part, Das, Flocchini, Prencipe, Santoro et Yamashita [37] se sont intéressés à un autre mécanisme de communication : la lumière. Plusieurs robots se trouvent dans un plan et veulent faire le rassemblement. Ils se déplacent selon le principe du *Regarde-Calcule-Bouge*. Les agents sont anonymes et au début de chaque cycle ils ne se rappellent de rien, c'est à dire que leur mémoire est instantanée et elle s'efface une fois que les agents s'arrêtent. En revanche, chaque robot possède une lumière qui est toujours allumée mais qui change de couleurs selon l'état dans lequel le robot se trouve. Ce mécanisme de communication a été introduit par Peleg [89] et il permet à chaque robot de connaître l'emplacement des autres grâce à leurs lumières permanentes. Si tous les robots possèdent une lumière de couleur unique alors cela va correspondre au modèle classique des robots anonymes. Les auteurs ont fourni une solution qui se base sur le principe que chaque agent possède 5 couleurs différentes de lumière : T, M, S, F, W. À l'état initial, tous

1. *$\tilde{O}$ signifie un O à un facteur logarithmique près.*

les robots allument leurs lumières T ; c'est la couleur qui indique aussi qu'un robot va commencer un nouveau cycle du *Regarde-Calcule-Bouge*. Lorsqu'il termine son cycle, le robot allume la lumière F. Durant son mouvement, il doit allumer la lumière M et si des robots observent cette couleur durant leur phase de *Regarde*, ils n'ont pas le droit de bouger et ils doivent allumer leurs lumières de couleur W pour indiquer qu'ils attendent leurs tours. Un robot utilise la lumière S lorsqu'il termine sa phase *Bouge* et qu'il voit qu'aucun robot n'a la lumière T.

Le dernier mécanisme dont nous allons parler dans cette section est celui de la communication *face à face*, c'est à dire que les agents peuvent échanger des informations directement entre eux lorsqu'ils se rencontrent. C'est le mécanisme que Dieudonné, Pelc et Peleg [45] ont utilisé pour traiter le problème du rassemblement en présence d'agents byzantins. Si plusieurs agents se trouvent dans le même nœud à la même ronde, ils peuvent échanger toutes les informations qu'ils possèdent y compris les valeurs de leurs étiquettes et la liste des bons agents.

Chapitre 3

Rendez-vous déterministe avec détection utilisant des bips

Dans ce chapitre, nous nous intéressons au problème du rendez-vous entre deux agents mobiles qui sont capables de communiquer entre eux en utilisant les bips. Le *bip* est un signal sonore très court qui ressemble à celui émis par un répondeur téléphonique pour indiquer le moment de laisser le message. Ce concept a été introduit la première fois par Cornejo et Kuhn [25] pour résoudre le problème du coloriage des nœuds dans un réseau de transmission. C'est un problème très connu dans la théorie du graphe et qui consiste à utiliser le minimum de couleurs possible pour colorier les nœuds d'un graphe en respectant la règle que deux nœuds adjacents ne peuvent pas avoir la même couleur. Les bips ont été utilisés par plusieurs chercheurs [1, 64, 67, 97] pour résoudre différents problèmes mais le point commun entre leurs recherches était que le signal sonore est émis par un nœud et reçu par ses nœuds voisins. Dans notre étude, ce sont les agents qui envoient et écoutent les bips.

Ce chapitre est structuré de la manière suivante. Nous commençons par la description des différents modèles que nous avons étudiés et par l'énoncé du problème. Nous

présentons ensuite notre contribution en décrivant tous les résultats obtenus. Le reste du chapitre est consacré à présenter quelques définitions et outils, les algorithmes et les preuves d'exactitude ainsi que la preuve d'optimalité des résultats.

3.1 Modèles et problèmes

Deux agents mobiles se trouvent à deux endroits différents dans un réseau pour réaliser le rendez-vous. Ils possèdent des étiquettes différentes sous forme d'entiers positifs. Chaque agent connaît sa propre étiquette mais pas celle de l'autre. Les agents commencent leurs déplacements à des moments arbitraires et possiblement différents de manière synchrone, c'est à dire qu'à chaque ronde, un agent peut soit rester immobile dans sa position actuelle soit se déplacer vers un nœud voisin. Chaque agent ne connaît pas le moment de départ de l'autre, ne possède aucune information ni sur la topologie du réseau ni sur une borne sur sa taille et ne connaît pas la position de départ de l'autre agent. Les agents sont équipés d'une mémoire de taille illimitée et ils sont activés par un adversaire dont le rôle est de compliquer le rendez-vous. Le réseau est modélisé par un graphe connexe, non orienté et dont les nœuds sont anonymes. Les agents ne peuvent en aucun cas marquer les nœuds visités mais par contre ils peuvent voir les numéros de ports associés à chaque nœud.

Dans le cas général lorsqu'on parle du rendez-vous synchrone, les chercheurs font référence à une rencontre faite par les agents dans un même nœud à la même ronde. Dans notre étude, le rendez-vous est plus exigeant. Nous voulons que, lorsque la rencontre est réalisée, les agents le sachent en déclarant simultanément le rendez-vous et en arrêtant l'exécution de l'algorithme. Ce genre de rencontre est connu dans la littérature (voir chapitre 2) sous le nom du rendez-vous *avec détection*. En revanche, pour réaliser ce type de rendez-vous, les agents ont besoin d'un moyen de communication pour déclarer

leur présence dans un nœud. Pour répondre à ce besoin, les agents utilisent les bips pour communiquer : à chaque ronde, un agent peut soit **biper** c'est à dire émettre un signal sonore, soit **écouter** c'est-à-dire rester silencieux. Un agent ne peut rien entendre lorsqu'il est en train de biper ce qui veut dire que si deux agents émettent des bips à la même ronde alors ils ne vont rien entendre.

Nous utilisons deux variantes de modèle de communication par bips : *local* et *global*. Dans le modèle de bips local, un agent qui se trouve dans un nœud u n'entend un bip à la ronde r que s'il écoute et qu'à cette ronde, l'autre agent qui a émis le bip se trouve dans u. Pour ce qui est du modèle de bips global, nous distinguons deux types de bips. On dit qu'un agent entend un bip *fort* à une ronde s'il écoute et que l'autre agent qui, se trouve dans le même nœud, a émis le bip à cette même ronde. On dit qu'un agent entend un bip *faible* à une ronde s'il écoute et que l'autre agent qui a émis le bip à cette même ronde, ne se trouve pas avec lui dans le même nœud. Dans le cas du bip faible, un agent ne peut pas déduire si l'autre est plus proche ou plus loin de lui ; la seule déduction dans ce cas c'est qu'il ne sont pas dans le même nœud. Le choix de ces deux modèles est dû au fait que dans le cas où les agents ont de très faibles capacités de transmission qui limitent la réception d'un bip, nous pouvons leur appliquer le modèle local ; par contre si les agents sont plus puissants dans le contexte où ils peuvent émettre des bips suffisamment forts pour être entendus dans tout le réseau et qu'ils ont une capacité d'écoute leur permettant de distinguer les bips émis dans un même nœud de ceux émis à des nœuds différents, on peut alors utiliser le modèle global.

Nous nous permettons de dire que notre modèle de bips local est possiblement le plus faible moyen de communication que les agents peuvent utiliser : le bip est le message le plus simple à envoyer, les agents ne peuvent rien entendre s'ils bipent au même moment et ils ne sont capables de communiquer que lorsqu'ils se trouvent simultanément dans le même nœud. En effet, les auteurs de [25] ont mentionné que les bips locaux constituent

une manière plus faible de communiquer que d'utiliser des messages d'un bit ; un bip local permet seulement de savoir s'il y a un signal ou non, par contre un message d'un bit permet de distinguer trois situations différentes : aucun message, le message 0 ou le message 1.

Un premier résultat qui est évident est que le rendez-vous avec détection est impossible si les agents ne peuvent pas communiquer entre eux. Il est aussi clair que dans le modèle global, si les agents ne peuvent pas faire la différence entre les bips forts et les bips faibles alors le rendez-vous avec détection est aussi impossible puisque c'est le seul moyen pour les agents de savoir s'ils sont dans le même nœud ou non. Une remarque importante est que tout algorithme qui résout le problème du rendez-vous avec détection dans le modèle local est capable de résoudre le même problème dans le modèle global, il suffit juste d'ignorer les bips faibles. Nous verrons plus tard que l'inverse n'est pas vrai.

Nous nous intéressons aussi au temps de réalisation du rendez-vous avec détection pour deux agents qui possèdent des étiquettes différentes et qui se trouvent dans des positions initiales différentes dans un graphe arbitraire. Ce temps représente le nombre total de rondes nécessaires pour exécuter en pire cas l'algorithme du rendez-vous avec détection à partir de l'activation du deuxième agent et jusqu'à la déclaration de la rencontre par les deux agents.

3.2 Notre contribution

Nous présentons en premier lieu un algorithme déterministe qui permet à deux agents de faire le rendez-vous avec détection dans un réseau arbitraire et anonyme. Cet algorithme est valable pour le modèle de bips global et local. Pour ce qui est du temps, notre algorithme réalise le rendez-vous avec détection en temps polynomial en n et $\log \ell$, où n est la taille du réseau et ℓ est la plus petite étiquette des deux agents. Ce temps est

le même que celui réalisé par l'algorithme [94], le plus rapide jusqu'à date, qui permet de faire le rendez-vous déterministe *sans détection* entre deux agents dans un graphe arbitraire et anonyme.

Notre deuxième résultat est une réflexion sur le fonctionnement de notre premier algorithme. Nous avons remarqué que les agents dépensent beaucoup d'énergie dans la solution que nous avons proposée. Plus précisément, le nombre total de parcours d'arêtes qu'un agent doit faire est proportionnel au temps du rendez-vous. Toutefois, dans de nombreuses applications, les agents peuvent être des robots mobiles qui utilisent des batteries ou des dispositifs d'alimentation qui s'épuisent avec le temps, et donc l'énergie qu'un agent peut dépenser pour se déplacer est limitée. Nous nous sommes donc posés la question suivante : si les agents ont une énergie limitée, c'est-à-dire qu'ils sont capables de faire au plus c mouvements pour un certain entier c, seront-ils toujours capables d'accomplir le rendez-vous avec détection ? Une réponse évidente est que c'est impossible dans certains réseaux de très grande taille : il suffit de prendre une ligne de taille $n > 2c+3$ dans laquelle les agents sont placés sur ses deux extrémités et ils commencent l'exécution de l'algorithme à la même ronde. Nous avons donc reformulé notre question : Est-ce que des agents qui possèdent une énergie limitée peuvent toujours accomplir le rendez-vous avec détection dans des réseaux de taille bornée ? Nous répondons positivement à cette question en fournissant un algorithme déterministe qui permet de faire le rendez-vous avec détection dans le modèle de bips local (ça fonctionne aussi pour le modèle de bips global). Nous avons prouvé, un résultat qui peut paraître surprenant, que le temps de ce rendez-vous est exponentiellement supérieur à celui des agents à grande énergie.

Le dernier résultat que nous présentons dans ce chapitre est un algorithme déterministe qui résout le problème du rendez-vous avec détection dans le modèle de bips global

pour des agents à énergie limitée en temps aussi rapide que celui réalisé par des agents à énergie illimitée dans des réseaux de taille bornée.

3.3 Préliminaires et définitions

Dans le reste de ce chapitre, le mot « graphe »signifie un graphe anonyme, connexe et non orienté modélisant un réseau. La *taille* d'un graphe est le nombre total de ses nœuds. Dans cette section, nous allons rappeler deux procédures connues dans la littérature et qui vont nous être utiles pour construire nos algorithmes.

La première procédure concerne l'exploration de graphe, c'est-à-dire la visite de tous les nœuds du graphe par un seul agent. Cette procédure, qu'on note $EXP(m)$, est basée sur la technique des séquences universelles d'exploration (UXS) introduite par Koucký [70] et découle du résultat établi par Reingold [91]. D'une manière très simple, si un agent se trouve sur n'importe quel nœud d'un graphe de taille au plus m alors la procédure $EXP(m)$ va lui permettre de visiter tous les nœuds de ce graphe en utilisant $R(m)$ traversées d'arêtes, où R est un polynôme.

Un « UXS »est une séquence infinie $x_1, x_2, \ldots$ d'entiers positifs. Compte tenu de cette séquence, dont les détails de sa construction se trouvent dans les articles [70, 91], la procédure $EXP(m)$ peut être décrite comme suit : à la ronde 1, l'agent quitte sa position initiale par le port 0. Pour $i \geq 1$, si l'agent entre dans un nœud u de degré d par le port p à la ronde i alors il doit quitter u à la ronde $i+1$ par le port $q = (p + x_i) \mod d$. Le résultat de Reingold implique que si un agent exécute la procédure $EXP(m)$ à partir de n'importe quel nœud v d'un graphe de taille au plus m alors il va visiter tous les nœuds de ce graphe après $R(m)$ rondes, ce qui veut dire tout simplement que le parcours de n'importe quel graphe se fait en temps polynomial en sa taille.

La seconde procédure concerne la réalisation du rendez-vous (sans détection) dans un graphe par deux agents mobiles. Elle est due à Ta-Shma et Zwick [94] et elle s'avère jusqu'à nos jours la procédure la plus rapide pour réaliser le rendez-vous déterministe dans les graphes arbitraires. Voici une brève description de cette procédure que nous allons utiliser dans notre algorithme de rendez-vous avec détection pour les agents à grande énergie.

Soient $\mathbb{Z}^+$ l'ensemble des entiers positifs et $\mathbb{Z}^* = \mathbb{Z}^+ \cup \{-1\}$. Pour tout entier positif L, les auteurs de [94] ont défini une fonction $\Phi_L : \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}^* \longrightarrow \mathbb{Z}^*$ que chaque agent va utiliser pour se déplacer dans le graphe : lorsqu'un agent se trouve dans un nœud v, la fonction Φ va lui indiquer à une ronde t par quel port il va quitter v pour rejoindre un nœud voisin ou s'il doit rester immobile sur v à cette ronde. Donc, les trois paramètres de la fonction Φ sont : la ronde t, le degré d du nœud v et le numéro de port p par lequel l'agent est entré dans le nœud v à la ronde $t-1$; dans le cas où l'agent était immobile à la ronde $t-1$, alors le numéro de port est remplacé par la valeur -1. La fonction Φ va retourner soit le numéro de port par lequel l'agent doit quitter le nœud v à la ronde t ou -1 s'il doit rester immobile sur v à cette ronde.

De manière plus formelle, la fonction Φ_L est utilisée par un agent qui possède une étiquette L et qui se trouve dans un nœud v d'un graphe G comme suit. Soit $v_0 = v$ et soit v_1 le nœud adjacent à v_0 lorsque l'agent quitte v par le port 0. Supposons qu'à la ronde $t-1$ l'agent a parcouru le chemin $v_0, v_1, \ldots, v_{t-1}$ alors pour $0 \leq i \leq t-1$, v_{i+1} est soit égal à v_i soit c'est un nœud adjacent à v_i. À la ronde t, le nœud v_t va être choisi de la manière suivante. Dans le cas où $v_{t-1} = v_{t-2}$ tel que d est le degré de v_{t-1} alors $v_t = v_{t-1}$ si $\Phi_L(t, d, -1) = -1$; si $\Phi_L(t, d, -1) = q \geq 0$ alors v_t est le nœud adjacent à v_{t-1} lorsque l'agent quitte ce dernier par le port q. Dans le cas où $v_{t-1} \neq v_{t-2}$, c'est à dire qu'à la ronde $t-1$, l'agent a accédé à v_{t-1} dont le degré est d par un certain port q en provenance de v_{t-2}, alors $v_t = v_{t-1}$ si $\Phi_L(t, d, p) = -1$; si $\Phi_L(t, d, p) = q \geq 0$ alors v_t est

le nœud adjacent à v_{t-1} lorsque l'agent quitte ce dernier par le port q. Par conséquent, l'application de la fonction Φ_L à partir d'un nœud v va générer un chemin infini dans G qui commence à partir de v pour l'agent qui possède l'étiquette L. Tout au long de ce chemin et à chaque ronde t, soit l'agent reste immobile sur sa position actuelle soit il visite un nœud adjacent par un port calculé par la fonction Φ_L. On dit qu'une ronde t est *active* pour un agent si $v_t \neq v_{t-1}$ et elle est *passive* si $v_t = v_{t-1}$.

Le résultat suivant, prouvé dans l'article [94], garantit le rendez-vous sans détection en temps polynomial entre deux agents qui possèdent respectivement les étiquettes ℓ et L et qui appliquent les fonctions Φ_ℓ et Φ_L dans un graphe arbitraire et anonyme.

Théorème 3.3.1 *Soit G un graphe de taille n et considérons deux agents avec des étiquettes différentes L_1 et L_2 respectivement. À la ronde t_1 un agent se trouve dans sa position initiale v et à la ronde t_2 l'autre agent se trouve dans sa position initiale w telle que $t_2 \leq t_1$. Pour $i \in \{1, 2\}$, si un agent qui possède l'étiquette L_i applique la fonction Φ_{L_i} à partir de sa position initiale alors le rendez-vous est réalisable en temps $O(P(n, \log \ell))$, où P est un polynôme et ℓ est la plus petite des deux étiquettes. De plus, cette rencontre se fait à une ronde $t \geq t_1$ qui est active pour un des agents et passive pour l'autre. Le rendez-vous est aussi réalisable si un agent reste définitivement immobile dans sa position initiale et l'autre agent applique sa fonction Φ_{L_i}.*

3.4 Rendez-vous des agents à énergie illimitée

Dans cette section, nous présentons et analysons un algorithme de rendez-vous avec détection qui fonctionne pour les agents à énergie illimitée, c'est-à-dire pour des agents qui peuvent dépenser une quantité arbitraire d'énergie pour se déplacer dans le réseau. Notre algorithme fonctionne même pour le modèle de bips local qui se présente comme le plus faible des deux modèles que nous avons définis ci-dessus.

Nous allons décrire une procédure qui va être utilisée par notre algorithme. Cette procédure, basée sur l'utilisation de la fonction Φ_L, décrit un parcours infini qui peut être réalisé par un agent qui possède l'étiquette L.

Procédure `Marche-bips`

Considérons un agent qui possède l'étiquette L et qui a pour position initiale le nœud v dans un graphe G. Soit W le parcours résultant de l'application de la fonction Φ_L dans le graphe G à partir du nœud v.

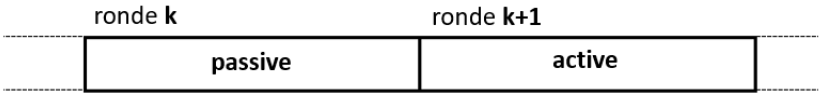

FIGURE 3.1 – Les états que peuvent avoir deux rondes k et $k+1$ de W.

Chaque ronde de W est remplacée par deux rondes consécutives qu'on définit comme suit. Si la ronde t de W est passive, c'est-à-dire que $v_t = v_{t-1}$, alors cette ronde est remplacée par deux rondes durant lesquelles l'agent reste dans v_t et écoute. Si la ronde t de W est active, c'est-à-dire que $v_t \neq v_{t-1}$, alors cette ronde est remplacée par les deux rondes t_1 et t_2 suivantes : à la ronde t_1, l'agent se rend au nœud v_t et émet un bip et à la ronde t_2, l'agent reste immobile dans v_t et écoute (voir figure 3.2).

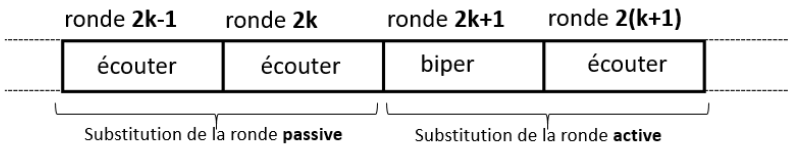

FIGURE 3.2 – Substitution des rondes k et $k+1$ de W.

Nous allons maintenant présenter notre algorithme pour le rendez-vous avec détection que chaque agent va exécuter. Il est important de remarquer que l'exécution de la procédure `Marche-bips`, lorsqu'elle est invoquée par cet algorithme, dépend seulement de l'étiquette de l'agent.

Algorithme `RV-avec-détection`

Exécuter la procédure `Marche-bips` **jusqu'à** entendre un bip

Soit s le numéro de la ronde actuelle

Rester immobile pour toujours

Émettre un bip à la ronde $s + 1$

Écouter à la ronde $s + 2$

Si aucun bip n'est entendu à la ronde $s + 2$ **alors**

    Déclarer le rendez-vous à la ronde $s + 3$ et arrêter

sinon

    Écouter à la ronde $s + 3$

    Déclarer le rendez-vous à la ronde $s + 4$ et arrêter.

L'idée derrière cet algorithme est que dans le cas du modèle de bips local, si un agent entend un bip alors il sera sûr que l'autre agent se trouve avec lui dans le même nœud. Il faut dans ce cas définir le mécanisme qui va leur permettre de déclarer le rendez-vous au même moment. Dans le cas du modèle de bips global, les agents vont considérer seulement les bips *forts* et ignorer les bips *faibles* pour réaliser leur rendez-vous avec détection à l'aide de cet algorithme.

Nous allons montrer maintenant que l'algorithme `RV-avec-détection` permet à deux agents qui possèdent deux étiquettes différentes de faire le rendez-vous avec détection. Nous allons aussi prouver que le temps d'exécution de cet algorithme est polynomial en la taille du graphe et en le logarithme de la plus petite étiquette des deux agents (la taille de l'étiquette). Dans le reste de cette section, si les agents commencent à exécuter l'algorithme à des rondes différentes alors nous appellerons l'agent qui commence l'exécution en premier le *premier* agent et celui qui commence l'exécution plus tard le

dernier agent. Dans le cas du départ simultané, ces deux caractéristiques sont attribuées arbitrairement aux agents.

Théorème 3.4.1 *Considérons deux agents avec des étiquettes différentes L_1 et L_2 respectivement et qui sont activés, possiblement à deux rondes différentes, dans deux positions initiales d'un graphe de taille n. Soit ℓ la plus petite étiquette des deux agents. Si les agents exécutent l'algorithme* `RV-avec-détection` *alors ils vont se rencontrer et déclarer simultanément le rendez-vous en temps $O(P(n, \log \ell))$, où P est un polynôme. Ce temps est calculé à partir de l'activation du dernier agent.*

Preuve : La première instruction que chaque agent va exécuter dans l'algorithme `RV-avec-détection` est la procédure `Marche-bips`. Nous allons prouver en premier lieu que lorsque les deux agents exécutent cette procédure, il existe une ronde à laquelle les deux agents se trouvent dans le même nœud et l'un des deux émet un bip et l'autre écoute. Selon le théorème 3.3.1, lorsque l'agent qui possède une étiquette L_1 applique à partir de sa position initiale la fonction Φ_{L_1} et l'autre agent applique à partir de sa position initiale la fonction Φ_{L_2} alors il existe une ronde à laquelle les deux agents vont se trouver dans le même nœud (rendez-vous sans détection) et telle que cette ronde va être *active* pour un agent et *passive* pour l'autre. Ce résultat est toujours vrai indépendemment des rondes d'activation des agents et de la configuration de leurs positions initiales. Par définition, la procédure `Marche-bips` est une simulation de l'application de la fonction Φ_{L_i} dans laquelle nous remplaçons chaque ronde par un segment de deux rondes. On dit qu'un segment est *actif* lorsque la ronde simulée par ce segment est *active* et qu'un segment est *passif* lorsque la ronde qu'il simule est *passive*. Plus précisément, dans un segment *actif*, un agent peut émettre un bip à la première ronde de ce segment et écouter à sa deuxième ronde tandis que dans un segment *passif*, un agent peut seulement écouter à ses deux rondes.

Soit r la ronde à laquelle les deux agents se sont rencontrés et supposons qu'elle est simulée par le ρ-ième segment du premier agent. Si les segments des deux agents sont alignés, c'est à dire que la première ronde du segment du dernier agent concorde avec la première ronde du segment du premier agent, alors à la première ronde du ρ-ième segment du premier agent, un agent émet un bip et l'autre écoute parce que ce segment est actif pour un agent et passif pour l'autre. Par conséquent, nous allons supposer que les segments ne sont pas alignés. Supposons que le σ-ième segment du dernier agent débute au cours du ρ-ième segment du premier agent, c'est à dire que la première ronde du σ-ième segment du dernier agent concorde avec la deuxième ronde du ρ-ième segment du premier agent. Selon le théorème 3.3.1, il existe deux cas possibles : soit le ρ-ième segment du premier agent est passif et donc le σ-ième segment du dernier agent est actif, soit le ρ-ième segment du premier agent est actif et le σ-ième segment du dernier agent est passif. Commençons par traiter le premier cas et soit r' la première ronde du σ-ième segment du dernier agent. À cette ronde, les deux agents se trouvent dans le même nœud, le dernier agent émet un bip et le premier agent écoute et par conséquent il entend le bip. Considérons maintenant le deuxième cas. Supposons que les deux agents se trouvent dans un nœud u à la ronde r. Puisque le σ-ième segment du dernier agent est passif alors durant le $(\sigma - 1)$-ième segment, le dernier agent se trouvait forcément dans u. Soit r'' la deuxième ronde du $(\sigma - 1)$-ième segment du dernier agent. À cette ronde, le dernier agent écoute peu importe si le segment $(\sigma - 1)$ est actif ou passif. D'autre part, puisque le ρ-ième segment du premier agent est actif alors r'' représente la première ronde de ce segment à laquelle le premier agent émet un bip et par conséquent le dernier agent va l'entendre. Ce qui montre que dans tous les cas, il existe une ronde à laquelle l'un des agents entend le bip que l'autre a émis.

Soit t la ronde à laquelle un agent entend pour la première fois un bip lors de l'exécution de l'algorithme `RV-avec-détection` par deux agents A_1 et A_2. Sans perte

de généralité, supposons que A_1 est l'agent qui a entendu le bip à la ronde t et A_2 l'autre agent. Selon l'algorithme, l'agent A_1 va rester immobile définitivement sur sa position actuelle à partir de la ronde t et émet un bip à la ronde $t+1$. L'agent A_2 a émis bien sûr un bip à la ronde t (le bip entendu par A_1) et écoute à la ronde $t+1$ puisqu'il est en train d'exécuter la procédure `Marche-bips`. Par conséquent, A_2 entend son premier bip à la ronde $t+1$, reste immobile définitivement sur sa position actuelle à partir de cette ronde et émet un bip à la ronde $t+2$. Puisque l'agent A_1 écoute à la ronde $t+2$, il va entendre le bip émis par A_2 et selon l'algorithme, il va rester en écoute à la ronde $t+3$ pour déclarer le rendez-vous à la ronde $t+4$ et arrêter l'exécution de l'algorithme. De sa part, l'agent A_2 écoute à la ronde $t+3$ mais il n'entend aucun bip à cette ronde. Selon l'algorithme, A_2 déclare le rendez-vous à la ronde $t+4$ et arrête l'exécution. Ce qui prouve l'exactitude de l'algorithme `RV-avec-détection`.

Pour ce qui est du temps d'exécution, par définition de la procédure `Marche-bips` et selon le théorème 3.3.1, les agents seront à la ronde t après un parcours d'une durée d'au plus $2P(n, \log \ell)$ rondes à partir du moment de l'activation du dernier agent. Puisque la déclaration du rendez-vous se fait simultanément par les agents à la ronde $t+4$, l'exécution de l'algorithme `RV-avec-détection` se fait en temps $O(P(n, \log \ell))$.

□

3.5 Rendez-vous des agents à énergie limitée dans le modèle local

Dans cette section, nous étudions le rendez-vous avec détection qui peut être réalisé par des agents à énergie limitée, c'est à dire qu'ils ne peuvent effectuer qu'un nombre

limité de déplacements dans le réseau. Lorsqu'on parle d'un déplacement dans le réseau, nous faisons référence à un parcours d'une arête par un agent. Nous allons commencer par expliquer les caractéristiques d'un agent à énergie limité. Soit c un entier positif. On dit qu'un agent est *c-limité* lorsqu'il peut effectuer au plus c déplacements, c'est à dire que le nombre total de parcours d'arêtes ne peut pas dépasser le nombre c. Il faut noter que nous ne limitons pas le nombre de bips qu'un agent peut émettre parce que la quantité d'énergie requise pour réaliser un déplacement est généralement beaucoup plus élevée que la quantité d'énergie nécessaire pour émettre des bips. Par conséquent, nous ne limitons pas le nombre de bips émis par les agents.

Lorsque nous avons présenté ce modèle dans une section précédente, nous avons montré que les agents *c-limités* ne seront pas en mesure de réaliser le rendez-vous avec ou sans détection dans des graphes arbitraires à l'aide d'un exemple dans lequel les agents se trouvent à une distance supérieure à $2c$. Même dans le cas où la distance initiale entre les agents est 1, il existe des graphes dans lesquels les agents *c-limités* ne peuvent pas accomplir le rendez-vous. Prenons par exemple deux graphes en étoile de taille n chacun et dont les centres sont liés par une arête et supposons que deux agents *c-limités* se trouvent initialement dans les centres des deux graphes en étoile. En pire cas, au moins un des agents doit effectuer au moins $2(n-1)$ parcours d'arêtes pour trouver l'arête qui lie les deux centres pour faire le rendez-vous, une chose qui est impossible pour des agents *c-limités* lorsque n est assez grand.

C'est pour cela que nous posons la question suivante : est-ce que les agents *c-limités* peuvent réaliser le rendez-vous avec détection dans des graphes de taille bornée ? Autrement dit, pour tout entier n, existe-t-il un entier c tel que les agents *c-limités* seront capables d'accomplir le rendez-vous avec détection dans tous les graphes de taille au plus n ? Nous répondons positivement à cette question dans le cas du modèle de bips local. Nous allons traiter le même problème dans le modèle de bips global dans la pro-

chaine section de ce chapitre. Il est aussi intéressant de mentionner que même si le graphe est borné, nous ne pouvons pas utiliser l'algorithme `RV-avec-détection` parce que le nombre de déplacements effectués par un agent qui possède une étiquette L est proportionnel à $P(n, \log L)$, ce qui peut être arbitrairement grand.

Avant de décrire notre nouvel algorithme qui permet de résoudre ce problème, nous allons définir une procédure qui prend en paramètre la taille du graphe n et qui va être utilisée par cet algorithme.

Procédure `Exploration-bips` (n)

Soit $EXP(n)$ la procédure décrite dans la section 3.3 qui permet de faire l'exploration de n'importe quel graphe connexe de taille au plus n. Remplacer chaque ronde r de $EXP(n)$ par trois rondes consécutives r_1, r_2 et r_3 comme suit. Si l'agent prend le port p pour se déplacer vers le nœud w à la ronde r de $EXP(n)$ alors il va rejoindre le nœud w par le port p et émettre un bip à la ronde r_1. Au rondes r_2 et r_3, l'agent reste dans le nœud w et écoute (voir figure 3.3).

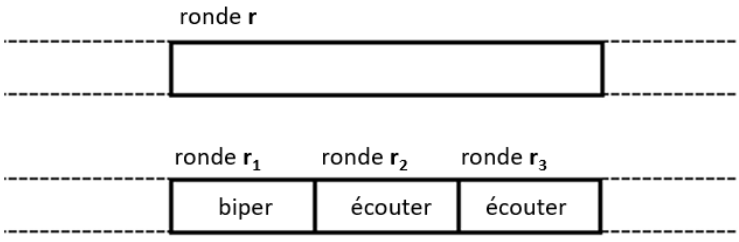

FIGURE 3.3 – Substitution de la ronde r de $EXP(n)$ par les rondes r_1, r_2 et r_3.

Il faut retenir que dans chacune des trois rondes qui remplacent la ronde r de $EXP(n)$, lorsqu'il exécute la procédure `Exploration-bips` (n), l'agent se trouve dans le même nœud que celui dans lequel il se trouvera à la ronde r s'il exécute la procédure $EXP(n)$.

Nous allons maintenant décrire notre algorithme qui permet de réaliser le rendez-vous avec détection pour des agents à énergie limitée. Cet algorithme est exécuté par un agent qui possède une étiquette L dans un graphe de taille au plus n. Rappelons que $R(n)$ est le temps d'exécution de la procédure $EXP(n)$. L'idée générale de notre algorithme est la suivante. La partie principale de l'algorithme que nous avons nommée *bloc* consiste à exécuter deux fois la procédure `Exploration-bips` (n) ; ces deux exécutions sont séparées par une longue *période d'attente* durant laquelle l'agent reste immobile sur sa position actuelle et écoute. La durée de cette *période d'attente* dépend seulement de l'étiquette de l'agent. Nous prouverons que quel que soit le délai entre les temps de départ des deux agents, une exécution complète de la procédure `Exploration-bips` (n) par un des agents doit soit coïncider avec la période d'attente de l'autre agent, soit va être réalisée après que l'autre agent ait exécuté deux fois cette procédure. Le bloc de l'algorithme exécuté par un agent donné est interrompu dans l'un des deux cas suivants : le premier est celui où l'agent entend un bip pendant sa période d'attente ou après avoir terminé son bloc. Pour le deuxième cas, c'est lorsque l'agent entend des bips dans deux rondes consécutives durant l'une des exécutions de la procédure `Exploration-bips` (n). Dans le premier cas, l'agent émet des bips durant deux rondes consécutives, déclare le rendez-vous à la ronde suivante et arrête l'exécution de l'algorithme. Dans le deuxième cas, il déclare le rendez-vous à la ronde suivante et arrête l'exécution de l'algorithme.

Après cette brève description, voici le pseudo-code de notre algorithme qui permet de faire le rendez-vous avec détection dans un graphe de taille au plus n et qui est exécuté par un agent qui possède une étiquette L. Nous utilisons une variable booléenne *attente* qui prend la valeur *faux* chaque fois que l'agent exécute la procédure `Exploration-bips` (n). La variable *attente* est mise à *vrai* lorsque l'agent se trouve dans sa période d'attente et après la deuxième exécution de la procédure `Exploration-bips` (n). Nous utilisons aussi une fonction *condition* qui retourne une valeur booléenne et qui prend en paramètre

la variable *attente*. Après chaque ronde, *condition* retourne l'évaluation de l'expression (*attente* **ET** entendre un bip) **OU** (¬*attente* **ET** entendre des bips durant deux rondes consécutives).

Algorithme `RV-avec-détection-énergie-limitée`

attente := faux
Effectuer la séquence d'actions suivante durant des rondes consécutives
et vérifier la valeur de `condition` à chaque ronde
jusqu'à `condition` soit vrai
    Exécuter la procédure `Exploration-bips` (n)
    attente := vrai
    Rester immobile et écouter pendant $6L \cdot R(n)$ rondes
    attente := faux
    Exécuter la procédure `Exploration-bips` (n)
    attente := vrai
    Rester immobile définitivement et écouter
s := le numéro de ronde à laquelle `condition` devient vrai
si *attente* **alors**
    Émettre un bip aux rondes $s+1$ et $s+2$
    Déclarer le rendez-vous à la ronde $s+3$ et arrêter
sinon
    Déclarer le rendez-vous à la ronde $s+1$ et arrêter.

Théorème 3.5.1 *Pour toute constante entière positive n, il existe un entier positif c tel que l'algorithme* `RV-avec-détection-énergie-limitée` *peut être exécutée par des agents c-limités dans n'importe quel graphe de taille au plus n. Si deux agents de ce*

type qui possèdent des étiquettes différentes exécutent cet algorithme dans un tel graphe, alors ils se rencontrent et déclarent simultanément le rendez-vous en temps $O(\ell^*)$ après l'activation du dernier agent, où ℓ^* est la plus grande étiquette des deux agents.

Preuve : Soit n un entier positif fixé à une valeur constante. Afin de montrer l'existence de l'entier c, il suffit de prouver que le nombre de déplacements réalisés par un agent lorsqu'il exécute l'algorithme dépend uniquement de n et non pas de l'étiquette L de l'agent. L'agent se déplace seulement pendant les deux exécutions de la procédure `Exploration-bips` (n) et au plus une fois chaque trois rondes par définition de la procédure. Puisqu'une exécution complète de la procédure `Exploration-bips` (n) se fait en $E = 3R(n)$ rondes alors l'agent fait au plus $2E/3 = 2R(n)$ déplacements. Il suffit donc de prendre $c = 2R(n)$.

Nous allons maintenant prouver l'exactitude de l'algorithme. Soit $B(L)$ la séquence des actions suivantes.

> Exécuter la procédure `Exploration-bips` (n) ;
>
> Rester immobile et écouter pendant $6L \cdot R(n)$ rondes ; (période d'attente)
>
> Exécuter la procédure `Exploration-bips` (n) ;

Pour n'importe quelle étiquette L, la séquence d'actions $B(L)$ sera appelée le *bloc* de l'agent qui possède l'étiquette L. Nous utiliserons l'affirmation suivante.

Affirmation. Considérons les blocs $B(L_1)$ et $B(L_2)$, pour $L_1 > L_2$, arbitrairement décalés dans le temps l'un par rapport à l'autre. Alors une des deux propriétés suivantes doit se réaliser. Soit une exécution entière de la procédure `Exploration-bips` (n) dans un des deux blocs coïncide avec la période d'attente dans l'autre bloc, soit une exécution entière de cette procédure dans un des blocs est réalisée à la fin de l'autre bloc.

Afin de prouver cette affirmation, considérons deux blocs $B(L_1)$ et $B(L_2)$, pour $L_1 > L_2$. Nommons le bloc $B(L_1)$ le plus grand bloc et $B(L_2)$ le plus petit. La période d'attente

dans le plus petit bloc a une taille de $Y = 2EL_2 \geq 2E$. Puisque $L_1 \geq L_2 + 1$, la période d'attente dans le plus grand bloc est de taille $2EL_1 \geq 2E(L_2+1) = 2EL_2+2E = Y+2E$. Attribuons le numéro 0 à la première ronde du bloc qui commence le premier.

Si le plus petit bloc commence avant le plus grand bloc, alors la deuxième exécution de la procédure `Exploration-bips` (n) dans le plus grand bloc est effectuée à la fin du plus petit bloc, et l'affirmation est prouvée. On peut donc supposer que le plus grand bloc commence avant le plus petit ou que les deux blocs commencent simultanément. Soit p le numéro de la première ronde du plus petit bloc. Il y a trois cas à traiter.

Premier cas : $0 \leq p \leq E$. Dans cette situation, la deuxième exécution de la procédure `Exploration-bips` (n) dans le plus petit bloc se trouve entièrement pendant la période d'attente du plus grand bloc.

Deuxième cas : $E < p \leq 2L_1E$. Dans ce cas, la première exécution de la procédure `Exploration-bips` (n) dans le plus petit bloc se trouve entièrement pendant la période d'attente du plus grand bloc.

Troisième cas : $p > 2L_1E$. Dans ce dernier cas, la deuxième exécution de la procédure `Exploration-bips` (n) dans le plus petit bloc est réalisée entièrement après la fin du plus grand bloc. Ce qui termine la preuve de l'affirmation.

Nous allons maintenant considérer deux agents qui possèdent des étiquettes différentes L_1 et L_2 telles que $L_1 > L_2$. Nous appelons l'agent qui possède l'étiquette L_1 *l'agent le plus grand* et celui qui possède l'étiquette L_2 *l'agent le plus petit*. L'affirmation que nous venons juste de prouver implique qu'il doit exister une ronde r à laquelle la fonction booléenne *condition* retourne la valeur « vrai » pour un des deux agents. En effet, une telle ronde doit se produire durant l'exécution de la procédure `Exploration-bips` (n) par un des agents qu'on va nommer A_1 ; cette exécution se fait entièrement soit dans la période d'attente de l'autre agent qu'on va appeler A_2, soit après que l'agent A_2 ait terminé la *deuxième* exécution de la procédure `Exploration-bips`

(n). Dans les deux cas, la variable *attente* pour l'agent A_2 a la valeur « vrai », il est immobile dans sa position actuelle et écoute durant une exécution complète de la procédure `Exploration-bips` (n) par l'agent A_1. Au cours de cette exécution, l'agent A_1 parcourt tout le graphe et émet un bip dans chaque nœud qu'il visite. Par conséquent, l'agent A_2 entend un bip dans la situation où sa variable *attente* a la valeur « vrai », ce qui signifie que la fonction booléenne *condition* retourne la valeur « vrai » pour l'agent A_2.

Soit r_0 la première ronde à laquelle la fonction *condition* retourne « vrai » pour un agent A. Nous allons prouver que la condition (*attente* **ET** entendre un bip) doit être satisfaite pour cet agent à la ronde r_0. En effet, supposons que la condition ($\neg$*attente* **ET** entendre des bips durant deux rondes consécutives) est satisfaite à la ronde r_0 ; ce qui veut dire que les deux bips consécutifs que A a entendus ont été émis à la ronde $r_0 - 1$ et r_0 par l'autre agent. Ce dernier n'a pas pu émettre ces deux bips consécutifs lors de l'exécution de son bloc B puisque la procédure `Exploration-bips` (n) ne permet à un agent de biper qu'une seule fois durant trois rondes consécutives. Par conséquent, les bips ont été émis après l'interruption du bloc de l'autre agent. Ceci n'est possible que lorsque la fonction *condition* retourne « vrai » pour l'autre agent, ce qui doit avoir eu lieu avant la ronde r_0 ; ce qui contredit la définition de cette ronde.

Nous allons maintenant considérer une ronde r_0 et l'agent qu'on nomme A_3 pour qui la condition (*attente* **ET** entendre un bip) est satisfaite à cette ronde. À r_0, cet agent se trouve dans un nœud v. À cette même ronde, l'autre agent qu'on nomme A_4 doit être encore en train d'exécuter son bloc et plus précisément, il doit exécuter la procédure `Exploration-bips` (n). À la ronde r_0, A_4 se trouve dans v, émet son bip et reste immobile pour écouter aux rondes consécutives $r_0 + 1$ et $r_0 + 2$. Durant ces deux rondes, l'agent A_3 reste immobile définitivement dans v, émet deux bips consécutifs, déclare le rendez-vous à la ronde $r_0 + 3$ et arrête son exécution. Quant à l'agent A_4,

après avoir entendu des bips aux rondes $r_0 + 1$ et $r_0 + 2$, la condition ($\neg$*attente* **ET** entendre des bips durant deux rondes consécutives) devient satisfaite pour lui à la ronde $s = r_0 + 2$. Par conséquent, l'agent A_4 qui se trouve toujours dans le nœud v déclare le rendez-vous à la ronde $s + 1 = r_0 + 3$ et arrête l'exécution de l'algorithme. Ce qui conclut la preuve d'exactitude de l'algorithme `RV-avec-détection-énergie-limitée`.

Il reste maintenant à estimer le temps du rendez-vous avec détection à partir de l'activation du dernier agent. Puisque n est constant, $R(n)$ est également constant, et donc le temps d'exécution du bloc $B(L)$ par un agent qui possède l'étiquette L est $(2L + 2) \cdot 3R(n)$ qui appartient à $O(L)$. Soit L_i tel que $i = 1$ ou $i = 2$, l'étiquette du dernier agent. Par conséquent, le temps entre le départ de cet agent et la déclaration du rendez-vous est au plus $(2L_i + 2) \cdot 3R(n) + 3$ qui appartient à $O(L_i)$. Puisque $L_i \leq \ell^*$ alors le rendez-vous avec détection se fait en temps $O(\ell^*)$.

□

Il est intéressant de comparer le temps requis pour compléter la tâche du rendez-vous avec détection établi par l'algorithme `RV-avec-détection` pour les agents à grande énergie avec le temps réalisé par l'algorithme `RV-avec-détection-énergie-limitée` pour les agents à énergie limitée. Cette comparaison est significative sur la classe des graphes pour lesquels les deux types d'agents peuvent accomplir le rendez-vous avec détection, c'est à dire pour les graphes bornés. Soit C_n la classe des graphes arbitraires de taille au plus n, où n est une constante. Considérons des agents *c-limités* tel que c est un nombre entier assez grand pour permettre de réaliser le rendez-vous avec détection pour la classe C_n en utilisant l'algorithme `RV-avec-détection-énergie-limitée`. Selon le théorème 3.4.1, les agents à grande énergie peuvent accomplir le rendez-vous avec détection en temps $O(P(n, \log \ell))$, c'est-à-dire en temps *polylogarithmique en la plus petite étiquette*, puisque n est constant. En revanche, nous avons prouvé dans le théorème 3.5.1 que les agents à énergie limitée peuvent accomplir le rendez-vous avec détection en

temps $O(\ell^*)$, c'est-à-dire un temps linéaire en la plus grande étiquette. Dans une telle situation, il est naturel de se questionner si cet écart exponentiel dans le temps, dû à la restriction énergétique, est inévitable. Le théorème suivant montre une borne inférieure qui répond positivement à cette question. Nous prouvons que cette borne inférieure sur le temps est nécessaire même dans le cas d'un graphe à deux nœuds, même avec un départ simultané des agents et même pour résoudre le problème du rendez-vous sans détection.

Théorème 3.5.2 *Soit c une constante positive. Dans le modèle de bips local, le temps du rendez-vous dans le graphe à deux nœuds pour des agents c-limités qui possèdent des étiquettes dans l'ensemble $\{1, \ldots, M\}$ est $\Omega(\sqrt[c]{M})$.*

Preuve : Soit $\mathcal{A}$ un algorithme qui permet à deux agents *c-limités* de faire le rendez-vous dans un graphe à deux nœuds. Supposons que les étiquettes de ces agents appartiennent à l'ensemble $\{1, \ldots, M\}$ et que les agents commencent l'exécution de $\mathcal{A}$ au même moment. Soit T le pire temps du rendez-vous sur toutes les paires d'étiquettes de l'ensemble $\{1, \ldots, M\}$. Pour n'importe quelle étiquette L dans $\{1, \ldots, M\}$, on note $\Phi_L : \{1, \ldots, T\} \longrightarrow \{0, 1\}$ la fonction qui représente une séquence binaire de longueur T définie de la manière suivante : Si l'agent avec l'étiquette L exécute seul l'algorithme $\mathcal{A}$ dans le graphe à deux nœuds alors $\Phi_L(i) = 1$ s'il se déplace à une ronde i et $\Phi_L(i) = 1$ s'il est immobile à cette ronde. Pour n'importe quelle étiquette L, la fonction Φ_L est bien définie parce que dans un graphe à deux nœuds, la séquence binaire permet à l'agent de savoir dans quelles situations il se trouvait dans les rondes précédentes. Donc l'exécution de l'algorithme $\mathcal{A}$ dépend uniquement de l'étiquette de l'agent.

Puisque les agents sont *c-limités* alors le nombre total des 1 dans chaque fonction Φ_L est au plus c. Donc le nombre de fonctions Φ_L possible pour un agent qui possède une

étiquette L et qui réalise le rendez-vous en temps T est

$$\binom{T}{0} + \binom{T}{1} + \cdots + \binom{T}{c} \leq 1 + T + T^2 + \cdots + T^c \leq 2T^c.$$

Si $2T^c < M$ alors il existe deux étiquettes L_1 et L_2 dans $\{1, \ldots, M\}$ telles que $\Phi_{L_1} = \Phi_{L_2}$. Les agents qui possèdent ces deux étiquettes ne pourront pas se rencontrer à la ronde T, car ils se déplacent exactement durant les mêmes rondes jusqu'à la ronde T. Donc les agents seront toujours dans différents nœuds à chaque ronde, d'où $2T^c \geq M$. Ce qui montre que $T \in \Omega(\sqrt[c]{M})$.

□

Le théorème 3.5.2 implique que dans le modèle de bips local, les agents à énergie limitée ont besoin d'un temps en $\Omega(\sqrt[c]{\ell^*})$ pour réaliser le rendez-vous, où ℓ^* est la plus grande étiquette et c est une constante. Le corollaire suivant découle des théorèmes 3.4.1, 3.5.1 et 3.5.2.

Corollaire 3.5.1 *Le rendez-vous avec détection des agents à énergie limitée est réalisable dans la classe des graphes de taille bornée dans le modèle de bips local, mais son temps de réalisation doit être exponentiellement plus grand que le meilleur temps de rendez-vous avec détection d'agents à grande énergie dans cette classe de graphes.*

3.6 Rendez-vous des agents à énergie limitée dans le modèle global

Dans cette section, nous allons présenter notre résultat final concernant le rendez-vous avec détection dans le modèle de bips global. Tel qu'indiqué dans la section 3.1, ce modèle est plus fort que le modèle local puisqu'un agent peut aussi entendre les bips d'un

autre agent qui ne se trouve pas avec lui dans le même nœud. Dans le cas du modèle global, la borne inférieure prouvée dans le théorème 3.5.2 n'est plus valide. En fait dans ce modèle, nous montrons que les agents à énergie limitée peuvent réaliser le rendez-vous avec détection dans la classe des graphes de taille bornée en temps logarithmique dans la plus petite étiquette. Nous prouvons aussi que ce temps est optimal, même dans les graphes à deux nœuds.

L'idée principale de notre algorithme est de briser en premier lieu la symétrie entre les agents en temps logarithmique en la plus petite étiquette sans que les agents effectuent un seul déplacement, une chose qui est possible puisqu'un agent a la capacité d'entendre les bips émis par un autre agent qui se trouve à n'importe quel endroit dans le graphe. Pour réaliser cette tâche, nous allons définir la procédure `Briser-symétrie` utilisée par notre algorithme et qui permet à un agent qui l'exécute d'émettre des bips ou écouter selon une stratégie bien définie. Ce qui est intéressant, c'est qu'à la fin de cette procédure, les agents seront synchronisés pour exécuter le reste des instructions de l'algorithme.

Après le bris de la symétrie, un des agents reste immobile dans sa position actuelle et l'autre agent essaye de le trouver en utilisant un nombre limité de déplacements. Ceci est réalisé grâce à la procédure `Exploration-bips-modifiée` (une version modifiée de la procédure `Exploration-bips`) qui permet à l'agent qui se déplace d'effectuer l'exploration du graphe tout en émettant des bips dans chaque deuxième ronde. Une déclaration correcte du rendez-vous est possible non seulement parce que les agents peuvent faire la distinction entre les bips forts et les bips faibles, mais aussi en raison de la synchronisation des agents qui permet à chacun d'eux d'entendre les bips de l'autre.

Nous procédons maintenant à la description détaillée de l'algorithme. Nous commençons tout d'abord par définir des transformations de l'étiquette L d'un agent qui vont servir à organiser son déplacement. Soit $(c_1 \dots c_k)$ la représentation binaire de l'étiquette L. Cette représentation va subir deux transformations. La première transformation T_1

consiste à doubler chaque bit et d'ajouter la séquence 01 au début et à la fin de la nouvelle séquence, c'est à dire que $T_1(L) = (01c_1c_1c_2c_2 \dots c_kc_k01)$. La deuxième transformation $T_2(L)$ remplace chaque bit 0 de $T_1(L)$ par la séquence 00 et chaque bit 1 par 10. On remarque bien que la taille de $T_2(L)$ est $2(2k+4)$ qui appartient à $O(\log L)$.

La procédure suivante, qui va être utilisée par notre algorithme, est exécutée par chaque agent à son activation. Durant l'exécution de cette procédure, les agents restent immobiles dans leurs positions initiales et réussissent à briser la symétrie grâce à leurs différentes étiquettes.

Procédure `Briser-symétrie`

Soit $T_2(L) = (d_1 \dots d_s)$

$i := 1$

Répéter à des rondes consécutives **jusqu'à** entendre un bip

    Si ($i \leq s$ **ET** $d_i = 1$) **alors** émettre un bip

    sinon écouter

    $i := i + 1$

Soit r la ronde à laquelle l'agent entend son premier bip

Si (un bip est émis à la ronde $r-1$) **alors**

    déclarer la ronde $r+1$ *rouge*

    $role := attente$

sinon

    émettre un bip à la ronde $r+1$

    déclarer la ronde $r+2$ *rouge*

    $role := marche$;

Si le bip entendu est *fort* **alors**

    déclarer rendez-vous à la ronde *rouge* et arrêter.

Lemme 3.6.1 *La procédure* `Briser-symétrie` *permet à deux agents qui possèdent des étiquettes différentes de déclarer une même ronde comme rouge. Pour un des agents, la ronde rouge est la ronde qui vient juste après avoir entendu un bip pour la première fois ; cet agent met la variable role à attente. Pour l'autre agent, le ronde rouge est la deuxième ronde qui vient juste après avoir entendu un bip pour la première fois ; cet agent met la variable role à marche. Une ronde rouge est déclarée par les deux agents en temps $O(\log \ell)$ après l'activation du dernier agent, où ℓ est la plus petite étiquette.*

Preuve : Nous allons montrer en premier lieu qu'il existe une ronde ξ à laquelle les deux agents sont présents dans le graphe et que l'un des deux entend un bip. Dans la définition de cette procédure, un agent ne doit émettre un bip que si la valeur du bit de son étiquette transformée est 1 ; lorsque la valeur est 0, il doit seulement écouter. Soit L_1 l'étiquette de l'agent A_1 et L_2 l'étiquette de l'agent A_2. Nous allons considérer trois cas dans lesquels nous allons montrer l'existence de la ronde ξ comme elle est décrite ci-dessus. Dans ces trois cas, nous supposons qu'aucun agent n'a entendu un bip durant une des rondes précédentes.

Premier cas : Les deux agents sont activés à la même ronde et possèdent des étiquettes avec des représentations binaires de longueur égale.
Soit i la première position des bits pour lesquels les séquences $T_2(L_1)$ et $T_2(L_2)$ diffèrent. Puisque les agents sont activés à la même ronde, cette dernière doit correspondre à la i-ème ronde pour les deux agents. À cette ronde, un des agents émet un bip et l'autre écoute, donc ce dernier va entendre un bip.

Deuxième cas : Les deux agents sont activés à la même ronde mais ils ont des étiquettes avec des représentations binaires de longueurs différentes.
Sans perte de généralité, soit L_1 l'étiquette qui possède la taille la plus petite de sa représentation binaire. Soit $T_2(L_1) = (d_1 \dots d_s)$. Soit α la ronde à laquelle les deux

agents sont activés et soit $\gamma = \alpha + s - 1$. L'agent A_1 émet un bip à la ronde $\gamma - 1$ et écoute aux rondes $\gamma - 3$, $\gamma - 2$ et γ. L'agent A_2 soit il écoute aux rondes $\gamma - 3$, $\gamma - 2$, $\gamma - 1$ et γ, soit il émet des bips aux rondes $\gamma - 3$ et $\gamma - 1$ et écoute aux rondes $\gamma - 2$ et γ. Dans les deux cas, un des agents entend un bip soit à la ronde $\gamma - 3$ soit à la ronde $\gamma - 1$.

Troisième cas : Les agents sont activés à des rondes différentes.

Sans perte de généralité, soient A_1 le premier agent activé et α sa ronde d'activation. Soit $\beta > \alpha$ la ronde d'activation de A_2 et soit s la longueur de la séquence binaire $T_2(L_1)$. Si $\beta = \alpha + 1$ ou $\beta = \alpha + 2$ alors A_2 entend un bip à la ronde $\alpha + 2$. Si $\beta = \alpha + 3$ alors A_2 entend un bip à la ronde $\alpha + 4$ parce que le premier bit de la représentation binaire de chaque étiquette L est 1 et par conséquent le cinquième bit de $T_2(L_1)$ est 1.

Supposons maintenant que $\alpha + 4 \leq \beta \leq \alpha + s - 8$. Nous considérons les trois possibilités suivantes. Si $\beta - \alpha$ est impair alors l'agent A_2 émet un bip à la ronde $\beta + 2$ à laquelle l'agent A_1 écoute, donc ce dernier va entendre le bip à cette ronde. Si $\beta - \alpha$ est divisible par 4 alors l'agent A_2 écoute à la ronde β et émet un bip à la ronde $\beta + 2$. L'agent A_1 soit il écoute aux rondes β et $\beta + 2$ soit il émet des bips à ces rondes et par conséquent un des agents entend un bip à une de ces rondes. Le cas de $\beta - \alpha = 4i + 2$, pour un certain entier i, est un peu plus compliqué. Dans ce cas, nous divisons toutes les rondes qui commencent à partir de la ronde α en segments de taille 2. Supposons que le début d'un tel segment correspond à une ronde β. les segments correspondent aux bits des séquences $T_1(L_1)$ et $T_1(L_2)$. Soit $T_1(L_1) = 01c_1c_1c_2c_2 \ldots c_kc_k01$ et considérons les deux derniers bits 01 de la séquence $T_1(L_2)$. Soit I le segment qui correspond au bit 0 et J le segment qui correspond au bit 1. Il existe quatre situations possibles :

1. Le segment I correspond à un certain c_j dans $T_1(L_1)$ et le segment J correspond à c_{j+1} dans $T_1(L_1)$.

Dans ce cas, la seconde copie de c_{j+1} correspond à un segment dans lequel l'agent A_2 a déjà terminé et il écoute. Si $c_{j+1} = 0$ alors J correspond à 0 dans $T_1(L_1)$ et à 1 dans $T_1(L_2)$. Donc A_1 entend un bip à la première ronde de ce segment. Si $c_{j+1} = 1$ alors, à la première ronde du segment qui vient tout juste après le segment J, l'agent A_1 émet un bip et l'agent A_2 écoute, et par conséquent A_2 entend un bip.

2. Le segment I correspond à c_k dans $T_1(L_1)$.

   Dans ce cas I correspond à la deuxième copie de c_k dans $T_1(L_1)$, et donc J correspond à 0 dans $T_1(L_1)$ et à 1 dans $T_1(L_2)$. Par conséquent, l'agent A_1 entend un bip à la première ronde de ce segment.

3. Le segment I correspond au dernier bit 1 dans $T_1(L_1)$.

   Dans ce cas, l'agent A_2 entend un bip à la première ronde du segment I.

4. Dans le segment I, l'agent A_1 a déjà terminé et il écoute.

   Dans ce cas, l'agent A_1 a lui aussi terminé et il écoute dans le segment J correspondant à 1 dans $T_1(L_2)$. Par conséquent, il entend un bip à la première ronde de ce segment.

Il s'ensuit que lorsque $\alpha + 4 \leq \beta \leq \alpha + s - 8$, un des agents entend un bip, au plus tard, à la ronde qui suit immédiatement la dernière ronde qui correspond au bit de $T_2(L_2)$ (ou de façon équivalente, au plus tard à la ronde qui suit immédiatement la dernière ronde qui correspond au bit de $T_1(L_2)$).

Si $\beta = \alpha + s - 7$ alors l'agent A_2 écoute à la ronde $\beta + 1$ et émet un bip à la ronde $\beta + 2$, alors que l'agent A_1 soit il écoute aux rondes $\beta + 1$ et $\beta + 2$ soit il émet un bips à la ronde $\beta + 1$ et écoute à la ronde $\beta + 2$. Donc un des agents entend un bip dans une de ces deux rondes. Si $\beta = \alpha + s - 6$ alors l'agent A_2 émet un bip à la ronde $\beta + 2$ et l'agent A_1 écoute à cette ronde, et par conséquent A_1 entend un bip. Si $\beta = \alpha + s - 5$

alors l'agent A_2 émet un bip à la ronde $\beta+2$ et l'agent A_1 écoute à cette ronde, et donc A_1 entend un bip. Enfin, si $\beta > \alpha + s - 5$ alors l'agent A_2 émet un bip à la ronde $\beta+4$, étant donné que le premier bit de la représentation binaire de chaque étiquette L est 1 et donc le cinquième bit de $T_2(L_2)$ vaut 1, et l'agent A_1 écoute à cette ronde puisqu'il a déjà terminé le traitement des bits de $T_2(L_1)$), donc A_1 entend un bip.
Ce qui termine le traitement du **troisième cas**.

Ces arguments montrent qu'un des deux agents doit forcément entendre un bip dans une certaine ronde ξ. De plus, notre analyse des trois cas ci-dessus montre clairement que la ronde ξ est atteinte en un temps au plus $O(\log \ell)$ rondes après l'activation du dernier agent, où ℓ est la plus petite étiquette des deux agents.

Regardons maintenant comment les agents vont faire la déclaration de la ronde *rouge*. Soit ρ la première ronde à laquelle un des deux agents entend pour la première fois un bip. Appelons cet agent A_3 et l'autre agent A_4. Il faut noter que l'agent A_3 ne pouvait pas émettre un bip à la ronde $\rho-1$ parce que si c'était le cas, l'agent A_4 qui était déjà actif à la ronde $\rho-1$, et à laquelle il écoutait, aurait déjà entendu un bip à cette ronde (les agents ne peuvent pas émettre dans deux rondes consécutives) et ceci entre en contradiction avec la définition de la ronde ρ. Par conséquent, selon la procédure `Briser-symétrie`, l'agent A_3 émet un bip à la ronde $\rho+1$ et déclare la ronde $\rho+2$ comme *rouge*. L'agent A_4 qui écoute à la ronde $\rho+1$ entend un bip à cette ronde pour la première fois. Puisqu'il a émis un bip à la ronde $(\rho+1)-1$, l'agent A_4 déclare la ronde $(\rho+1)+1$ comme *rouge*. Donc les deux agents déclarent la même ronde $(\rho+1)+1$ comme *rouge*.

Pour l'agent A_4, la ronde *rouge* est celle qui vient immédiatement après la ronde à laquelle il entendu un bip pour la première fois, donc cet agent met la variable *role* à *attente*. Pour l'agent A_3, la ronde *rouge* est la deuxième qui vient immédiatement après avoir entendu un bip pour la première fois et cet agent met la variable *role* à *marche*.

Puisque la ronde ρ peut être atteinte en temps au plus $O(\log \ell)$ après l'activation du dernier agent, alors ce temps reste aussi valable pour la ronde déclarée *rouge*, ce qui termine la preuve du lemme.

□

Pour construire notre algorithme, nous allons aussi utiliser une version modifiée de la procédure `Exploration-bips` (n) décrite à la section 3.5.

Procédure `Exploration-bips-modifiée` (n)

Soit $EXP(n)$ la procédure décrite dans la section 3.3 qui permet de faire l'exploration de n'importe quel graphe connexe de taille au plus n. Remplacer chaque ronde r de $EXP(n)$ par deux rondes consécutives r_1 et r_2 comme suit. Si l'agent prend le port p pour se déplacer vers le nœud w à la ronde r de $EXP(n)$ alors il va rejoindre le nœud w par le port p et émettre un bip à la ronde r_1. À la ronde r_2, l'agent reste dans le nœud w et écoute (voir figure 3.4).

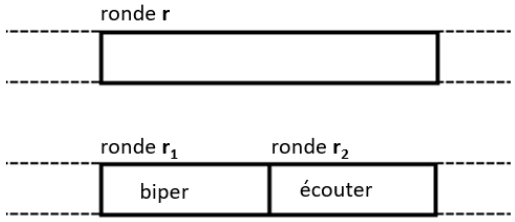

FIGURE 3.4 – Substitution de la ronde r de $EXP(n)$ par les rondes r_1 et r_2.

Il faut retenir que dans chacune des deux rondes qui remplacent la ronde r de $EXP(n)$, lorsque l'agent exécute la procédure `Exploration-bips-modifiée` (n), il se trouve dans le même nœud que celui dans lequel il se trouvera à la ronde r s'il exécute la procédure $EXP(n)$.

Nous allons maintenant présenter le pseudo-code de notre algorithme qui permet de réaliser le rendez-vous avec détection pour des agents à énergie limitée dans le modèle

de bips global. Cet algorithme est exécuté par un agent qui possède une étiquette L dans un graphe de taille au plus n.

Algorithme `Rapide-RV-avec-détection-énergie-limitée`

Exécuter la procédure `Briser-symétrie`

Si *role* = *attente* **alors**

Rester immobile et écouter **jusqu'à** entendre un bip *fort*

Soit t la ronde à laquelle l'agent entend pour la première fois un bip *fort*

Émettre un bip à la ronde $t + 1$

Déclarer le rendez-vous à la ronde $t + 2$ et arrêter

sinon

Exécuter la procédure `Exploration-bips-modifiée` (n) à partir de la ronde *rouge* **jusqu'à** entendre un bip *fort*

Soit t la ronde à laquelle l'agent entend pour la première fois un bip *fort*

Déclarer le rendez-vous à la ronde $t + 1$ et arrêter.

Théorème 3.6.1 *Pour toute constante entière positive n, il existe un entier positif c tel que l'algorithme* `Rapide-RV-avec-détection-énergie-limitée` *peut être exécutée par des agents c-limités dans n'importe quel graphe de taille au plus n, dans le modèle de bips global. Si deux agents de ce type qui possèdent des étiquettes différentes exécutent cet algorithme dans un tel graphe, alors ils se rencontrent et déclarent simultanément le rendez-vous en temps $O(\log \ell)$ après l'activation du dernier agent, où ℓ est la plus grande étiquette des deux agents. Ce temps est optimal même dans le cas du graphe à deux nœuds.*

Preuve : Soit n un entier positif fixé à une valeur constante. Afin de montrer l'existence de l'entier c, il suffit de prouver que le nombre de déplacements réalisés par un agent lorsqu'il exécute l'algorithme dépend uniquement de n et non pas de l'étiquette L de l'agent. L'agent se déplace seulement pendant l'exécution de la procédure `Exploration-bips-modifiée` (n) et une fois chaque deux rondes par définition de la procédure. Puisqu'une exécution complète de la procédure `Exploration-bips-modifiée` (n) se fait en $2R(n)$ rondes alors il suffit de prendre $c = R(n)$.

Nous allons maintenant prouver l'exactitude de l'algorithme. Selon le Lemme 3.6.1, les deux agents déclarent la même ronde comme *rouge*. Lors de l'exécution de la procédure `Briser-symétrie`, les deux agents entendent soit un bip *fort* soit un bip *faible*, car les agents ne se déplacent pas pendant l'exécution de cette procédure. S'ils entendent un bip *fort* alors ils sont dans le même nœud à la ronde *rouge* et ils peuvent déclarer correctement le rendez-vous à cette ronde et s'arrêter. Supposons maintenant qu'ils entendent un bip *faible*. Selon le lemme 3.6.1 à la ronde *rouge*, un des agents qu'on va nommer A_1 a la variable $role = attente$, et l'autre agent qu'on nomme A_2 a la variable $role = marche$. L'agent A_1 reste donc immobile dans son nœud de départ v et il écoute. Soit σ la première ronde à laquelle l'agent A_2 arrive au nœud v. Cette ronde existe parce qu'en exécutant la procédure `Exploration-bips-modifiée`, A_2 sera capable de parcourir tout le graphe. Une fois arrivé dans v, A_2 émet un bip à la ronde σ. L'agent A_1 entend pour la première fois un bip *fort* et par conséquent il émet un bip à la ronde $r = \sigma + 1$ et déclare le rendez-vous à la ronde $r + 1 = \sigma + 2$. L'agent A_2 entend un bip *fort* pour la première fois à la ronde $\sigma + 1$ et donc déclare le rendez-vous à la ronde $\sigma + 2$. Ce qui prouve l'exactitude de cet algorithme.

Il reste à estimer le temps du rendez-vous avec détection à partir de l'activation du dernier agent. Selon le lemme 3.6.1, le temps écoulé à partir de l'activation du dernier agent jusqu'à la déclaration de la ronde *rouge* appartient à $O(\log \ell)$ lorsque cet agent

exécute la procédure `Briser-symétrie`, où ℓ est l'étiquette la plus petite des deux agents. Puisque n est constant alors $R(n)$ est également constant et donc le temps d'exécution de la procédure `Exploration-bips-modifiée` (n) est aussi constant. Il s'ensuit que le temps écoulé entre l'activation du dernier agent et la déclaration du rendez-vous est $O(\log \ell)$.

Nous allons maintenant prouver que le temps $O(\log l)$ du rendez-vous avec détection dans le modèle global est optimal. Considérons K_2, le graphe à deux nœuds. Pour chaque entier $x > 2$ et chaque algorithme $\mathcal{A}$ de rendez-vous avec détection dans le modèle global, travaillant en temps $t < (x-1)/2$, nous montrons qu'il existe deux étiquettes L_1 et L_2 avec des représentations binaires de taille x tel que l'algorithme $\mathcal{A}$ échoue si les agents sont placés dans les extrémités de K_2 et activés simultanément.

À chaque ronde, on peut décrire le comportement d'un agent à l'aide d'un couple de bits (m_i, b_i) tel qu'à une ronde i, m_i détermine si un agent est en mouvement ou il est immobile et b_i indique s'il émet un bip ou écoute. Par exemple si $(m_3, b_3) = (0, 1)$ alors à la ronde 3, cet agent est immobile et émet un bip. Par conséquent un agent qui exécute $\mathcal{A}$ en temps t peut avoir 4^t comportements possibles. Dans toute exécution où un agent n'entend aucun bip, son comportement dépend uniquement de son étiquette. D'autre part, il existe 2^{x-1} étiquettes dont les représentations binaires sont de taille x (chaque représentation binaire commence par 1). Donc $t < (x-1)/2$ implique que $2t < x-1$ et par conséquent $4^t < 2^{x-1}$; donc il existe au moins deux étiquettes qui possèdent des représentations binaires de la même longueur x et qui induisent le même comportement pour deux agents.

Aucun de ces agents n'entendra de bip car ils les émettent exactement aux mêmes rondes ; de plus, les agents seront toujours dans des nœuds différents durant toute l'exécution de $\mathcal{A}$. Donc ces agents ne peuvent pas réaliser le rendez-vous avec détection ce qui contredit la définition de $\mathcal{A}$. Par conséquent, le rendez-vous avec détection dans le

modèle de bips global nécessite un temps en $\Omega(\log \ell)$, où ℓ est la plus petite étiquette des deux agents.

□

3.7 Conclusion

Dans ce chapitre, nous avons présenté trois algorithmes qui permettent à deux agents mobiles qui possèdent deux étiquettes différentes de faire le rendez-vous avec détection dans un graphe non orienté et connexe. Les deux premiers fonctionnent dans le modèle local ainsi que le modèle global : un pour les agents à grande énergie dans des graphes arbitraires, et l'autre pour les agents à énergie limitée dans des graphes de taille bornée. Nous avons montré que dans ce deuxième cas, le temps du rendez-vous avec détection pour des agents à énergie limitée est exponentiellement plus grand que le meilleur temps du rendez-vous effectué par des agents à grande énergie. Plus précisément, pour réaliser le rendez-vous dans des graphes de taille bornée, les agents à énergie limitée doivent utiliser un temps polynômial en la plus grande étiquette, tandis que les agents à grande énergie peuvent le faire en temps polylogarithmique en la plus petite étiquette. Le troisième algorithme fonctionne seulement pour les agents à énergie limitée dans le modèle global, mais il est beaucoup plus rapide : il permet à deux agents d'effectuer le rendez-vous avec détection dans la classe des graphes de taille bornée en temps logarithmique en l'étiquette la plus petite. Nous avons prouvé que ce temps est optimal.

Le rendez-vous avec détection peut être considéré comme une procédure de pré-traitement pour la résolution d'autres problèmes importants dans des graphes. Parmi ces problèmes les plus connus, on cite celui de la construction d'une carte d'un graphe anonyme par un agent. Il est bien connu dans la littérature qu'un seul agent ne peut pas réaliser une telle carte s'il n'a pas la possibilité de marquer les nœuds du graphe (par

exemple, un seul agent ne peut pas connaître la taille d'un anneau orienté s'il ne peut pas marquer sa position initiale). Ce problème ne peut pas être résolu même si on utilise deux agents qui ne peuvent pas communiquer entre eux ; dans ce cas, chaque agent ne serait pas au courant de l'existence de l'autre et par conséquent se comporterait comme s'il était le seul agent présent dans le graphe.

En revanche, nos algorithmes de rendez-vous avec détection qui permettent à des agents d'utiliser les bips comme moyen de communication, peuvent servir, avec une simple modification, à construire la carte d'un graphe par deux agents. L'algorithme réalisé pour des agents à grande énergie peut être utilisé pour accomplir cette tâche dans des graphes arbitraires, et les deux algorithmes élaborés pour les agents à énergie limitée peuvent être utilisés pour réaliser cette tâche dans des graphes de taille bornée. Il faut mentionner que dans nos trois algorithmes, la symétrie est brisée au moment où les agents déclarent le rendez-vous. Dans le cas des algorithmes dans le modèle local, un des agents entend deux bips dans le nœud de la rencontre et l'autre entend un seul bip dans ce nœud. Dans le cas de l'algorithme dans le modèle global, un des agents a la variable $role = attente$ tandis que l'autre a $role = marche$.

Voici comment on peut modifier nos algorithmes. À la ronde qui suit la déclaration du rendez-vous, les agents peuvent démarrer simultanément la procédure suivante. Un des agents reste immobile et se comporte comme un jeton stationnaire ; cet agent émet des bips à chaque deux rondes. En revanche, l'autre agent exécute silencieusement l'exploration du graphe avec un jeton stationnaire décrite par exemple dans l'article [16] mais en remplaçant chaque ronde de l'exploration par deux rondes consécutives ; dans la première ronde, l'agent se déplace comme indiqué dans l'exploration de [16] et dans la seconde ronde, il reste immobile dans le nœud actuel. Les bips de l'agent immobile permettent à l'agent qui se déplace de reconnaître l'emplacement du jeton (agent immobile) à chaque visite, de terminer l'exploration et de construire la carte du graphe.

À la fin de l'exploration, l'agent qui possède la carte va se trouver dans le même nœud avec l'agent immobile et il peut donc l'informer de la fin de l'exploration en émettant un bip à la dernière ronde à laquelle l'agent immobile est toujours en situation d'écouter. Une fois que l'agent immobile entend le bip, les agents changent les rôles : celui qui était immobile va faire l'exploration pour avoir la carte du graphe et celui qui a fait la carte va jouer le rôle du jeton stationnaire. Il faut noter qu'un agent n'a pas la possibilité de communiquer la carte déjà acquise à l'autre agent en raison du modèle de communication très restrictif.

Chapitre 4

Rendez-vous déterministe dans les arbres utilisant les tableaux blancs

4.1 Motivation

L'exploration et le rendez-vous, effectués par des agents mobiles dans des réseaux modélisés par des graphes connexes et non orientés, sont parmi les problèmes les plus largement étudiés en calcul distribué. Dans l'exploration, un seul agent mobile doit visiter tous les nœuds d'un graphe pour, par exemple, collecter les données situées dedans. En revanche, le rendez-vous est une tâche qui se fait par deux agents, placés initialement dans différents nœuds du réseau, dans le but de se rencontrer. Nous nous intéressons aux algorithmes d'exploration et de rendez-vous qui fonctionnent dans le même environnement étudié dans le chapitre précédent, c'est à dire dans des graphes anonymes dont les nœuds possèdent des numéros de port qui peuvent être lus par les agents. Le déplacement des agents se fait d'une manière déterministe et synchrone.

Une remarque intéressante est que l'exploration peut être considérée comme un cas particulier du rendez-vous dans lequel un des agents reste immobile pour toujours et

l'autre fait le parcours du graphe. Puisque l'agent immobile se trouve dans un nœud arbitraire du graphe, le rendez-vous nécessite que l'autre agent visite tous les nœuds en pire cas. Le temps de l'exploration est calculé par le nombre total de rondes nécessaires pour visiter tous les nœuds par un agent tandis que celui du rendez-vous est représenté par le nombre de rondes nécessaires pour faire la rencontre à partir du moment où le dernier agent a été activé. Voici une question très intéressante : Serait-il possible d'avoir un algorithme déterministe qui permet de faire le rendez-vous en un temps proportionnel à celui de l'exploration ? La réponse à cette question est négative, même dans des graphes très simples, si les agents n'ont pas la possibilité de marquer les nœuds du graphe. En effet, les auteurs de [41] ont prouvé que le rendez-vous dans un graphe à deux nœuds nécessite un temps logarithmique en la plus petite étiquette, tandis que l'exploration de ce simple graphe prend évidemment le temps 1. Cette différence dans le temps d'exécution est due à la nécessité de rompre la symétrie entre les agents afin de se rencontrer ; si le marquage des nœuds est impossible alors le rendez-vous doit se faire en un temps proportionnel à la longueur de la représentation binaire de la plus petite des deux étiquettes des agents participants. En revanche, si le marquage est autorisé alors ce problème peut être résolu par le simple algorithme suivant : chaque agent marque son nœud de départ avec sa propre étiquette, performe l'exploration du graphe pour trouver la marque de l'autre et ensuite se rencontrer dans la position initiale de l'agent qui possède la plus petite étiquette. Cet algorithme prend un temps proportionnel à celui de l'exploration la plus efficace mais nécessite en pire cas des marques de la taille $\log L$ où L est la grandeur de l'espace des étiquettes possibles.

Par conséquent, il est naturel de se poser la question sur la plus petite taille de messages pouvant être laissés par des agents dans des nœuds pour réussir à faire le rendez-vous en un temps proportionnel à celui de l'algorithme d'exploration le plus efficace. Afin de considérer cette question, nous formalisons la possibilité de laisser des

messages dans des nœuds par la notion bien connue des *tableaux blancs* qui se trouvent dans chaque nœud. Tous les tableaux blancs ont la même taille W, ce qui signifie que, lors de la visite d'un nœud, chaque agent peut lire le contenu actuel du tableau blanc, peut l'effacer et écrire des messages dont la taille ne dépassera pas W bits qui peuvent alors être lus par l'agent lui-même lors d'une visite ultérieure ou lus et effacés par l'autre agent. Initialement, tous les tableaux blancs sont vides.

4.2 Modèle et résultats

Dans notre étude, nous nous sommes intéressés à la classe des arbres pour définir la plus petite taille du tableau blanc à utiliser. Cet intérêt découle du fait que le temps optimal d'exploration d'un arbre est déjà connu. Deux agents mobiles se trouvent dans deux nœuds différents d'un arbre. Les agents possèdent des étiquettes différentes qui appartiennent à l'ensemble $\{1, \ldots, L\}$. Chacun d'eux connaît sa propre étiquette mais ne connaît pas l'étiquette de l'autre. Les agents n'ont aucune information ni sur la topologie de l'arbre ni sur sa taille mais ils peuvent lire les numéros de ports associés à chaque nœud visité.

Nous allons montrer que pour la classe des arbres de taille n, la plus petite taille d'un tableau blanc permettant aux agents de réaliser le rendez-vous en temps optimal $O(n)$ (ce qui est aussi le temps de l'exploration la plus rapide) est $\Theta((\log L)/n)$.

4.3 Définitions et préliminaires

Avant d'énoncer notre algorithme, nous allons présenter quelques définitions nécessaires à son élaboration. Le *tour de base* est une procédure qui permet à un agent de parcourir un arbre de la manière suivante :

1. Si u est le nœud de départ de l'agent alors il le quitte par le port numéro 0 ;
2. Si l'agent entre dans un nœud v de degré d par le port i alors il doit le quitter par le port dont le numéro est $(i + 1) \bmod d$;
3. L'agent répète les étapes 2 et 3 jusqu'à ce que toutes les arêtes de l'arbre soient traversées pour revenir à la fin à son nœud de départ u.

La deuxième définition concerne le *code de l'arbre enraciné avec m nœuds*. Ce code représente la suite de taille $2(m - 1)$ et dont les éléments sont les numéros de ports consécutifs rencontrés lorsqu'un agent effectue le tour de base débutant dans la racine. Le port 0 à chaque feuille est noté deux fois : à l'entrée et à la sortie. Des codes des arbres enracinés sont identiques si et seulement si ces arbres enracinés sont isomorphes ; ceci produit un ordre linéaire de tous les arbres enracinés non-isomorphes, par l'ordre lexicographique de leurs codes.

Une des particularités des arbres c'est qu'ils possèdent des *centres*. Afin de trouver le centre d'un arbre T, un agent mobile peut exécuter la procédure suivante :

1. Soit T_1 l'arbre obtenu après avoir enlevé toutes les feuilles de l'arbre T ;
2. Répéter l'instruction suivante jusqu'à l'obtention d'un arbre avec au plus deux nœuds : enlever toutes les feuilles de l'arbre T_i et soit T_{i+1} l'arbre qui en résulte ;
3. Si le résultat final est un arbre avec un seul nœud alors ce dernier est appelé le *nœud central* de l'arbre T ;
4. Si le résultat final est un arbre avec deux nœuds alors l'arête qui lie ces derniers est appelée l'*arête centrale* de l'arbre T.

Un arbre T dont les nœuds possèdent des numéros de ports est appelé *symétrique* s'il existe un automorphisme f de l'arbre différent de l'identité (pour un certain nœud u, on a $f(u) \neq u$) et qui préserve la numérotation des ports. Si un arbre avec des numéros

de port possède un nœud central alors il ne peut pas être symétrique. Par contre si son centre est une arête e alors cet arbre est symétrique si et seulement si les numéros de port correspondants aux deux extrémités de e sont les mêmes et les deux sous-arbres enracinées aux extrémités de e sont isomorphes (voir figure 4.1). Ces sous-arbres sont appelés les *moitiés* de l'arbre T. Un code de chaque moitié est obtenu en exécutant le tour de base dans chacune d'elles en commençant à sa racine et en ignorant le port correspondant à e.

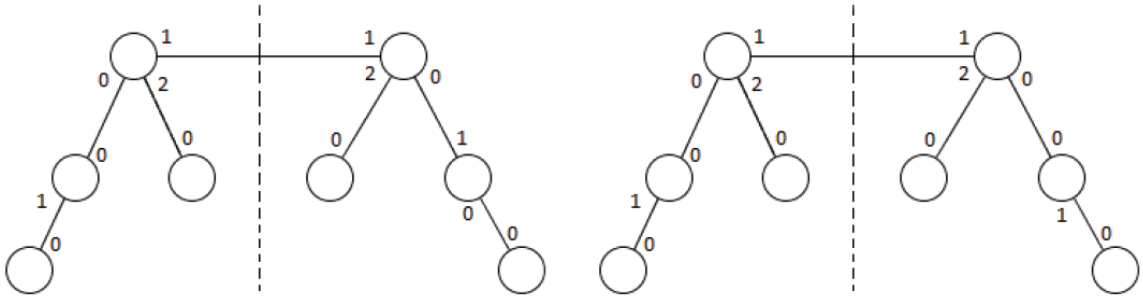

FIGURE 4.1 – Un arbre qui n'est pas symétrique (gauche) versus un arbre qui est symétrique (droite)

Après avoir performé le tour de base dans n'importe quel arbre, l'agent obtient une carte de l'arbre avec tous les numéros de port correctement marqués. Donc la topologie et la taille du graphe sont maintenant connues par l'agent. Il y a quatre cas exclusifs possibles :

1. L'arbre possède un nœud central ;
2. L'arbre possède une arête centrale dont les numéros de ports sont différents ;
3. L'arbre possède une arête centrale dont les numéros de ports sont égaux et l'arbre n'est pas symétrique ;
4. L'arbre est symétrique.

Dans les trois premiers cas, il est possible de définir sans ambiguïté un nœud v unique de l'arbre ; dans le premier cas c'est le nœud central, dans le second cas c'est l'extrémité de l'arête centrale qui possède le numéro de port le plus grand et dans le troisième cas c'est l'extrémité de l'arête centrale qui est la racine de la moitié de l'arbre dont le code est le plus grand lexicographiquement.

4.4 L'algorithme de rendez-vous et son analyse

Nous sommes maintenant prêts à formuler l'algorithme de rendez-vous. Soit λ la représentation binaire de l'étiquette d'un agent.

Algorithme `RV-arbre`

L'agent effectue le tour de base à partir de sa position initiale. Dans les cas 1, 2 et 3, l'agent se rend au nœud v par le chemin le plus court et s'arrête. Par conséquent, dans le reste de la description nous supposons que c'est le cas 4 qui se produit, c'est à dire que l'arbre est symétrique. Le tour de base permet à l'agent d'avoir la carte complète de l'arbre et de connaître en particulier sa taille n qui est un entier pair. Soient $k = \frac{n}{2}$, b la taille de la suite λ et λ^* la suite composée des éléments de λ complétée par une suite de zéros de taille $k - b$ comme préfixe dans le cas où $k > b$. Si $k \leq b$ alors $\lambda = \lambda^*$. Nous appelons λ^* l'*étiquette normalisée* de l'agent. Soient a la taille de la séquence λ^* et $x = \lfloor a/k \rfloor$, donc $x \geq 1$. Nous avons $a = kx + r$ où $0 \leq r < k$. Nous définissions la séquence $(s_1, \ldots, s_k)$ de chaînes binaires de manière à ce que la concaténation de tous les s_i soit égale à λ^*, la taille de s_i soit $x + 1$ pour $i \leq r$ et la taille de s_i soit x pour $i > r$.

L'agent se déplace vers l'extrémité u de l'arête centrale qui est la plus proche de sa position initiale en utilisant le chemin le plus court. Il lit le tableau blanc associé à u. Si ce tableau blanc est vide alors il y écrit sa chaîne s_1 ; dans ce cas, nous mettons $w := u$.

Si ce tableau n'est pas vide, l'agent se dirige vers l'autre extrémité u' de l'arête centrale et écrit sa chaîne s_1 sur le tableau blanc de u' ; dans ce cas, nous mettons $w := u'$. Ensuite, l'agent effectue le tour de base P dans la moitié de l'arbre enraciné en w. Plus précisément, lorsque l'agent visite un nœud par le port i, il le quitte par le port $(i + 1)$ mod d, où d est le degré du nœud. Chaque fois que l'agent visite un nœud, il lit le tableau blanc ; s'il n'est pas vide, l'agent visite le prochain nœud dans P, par contre si le tableau blanc est vide, l'agent écrit la chaîne consécutive s_i de son étiquette normalisée, en commençant par s_2, et va ensuite au nœud suivant dans P.

Après avoir terminé le tour P, l'agent visite l'autre extrémité w de l'arête centrale. Si le tableau blanc associé à w est vide alors l'agent s'immobilise dans ce nœud et attend l'arrivée de l'autre agent. Dans le cas contraire, il effectue le tour de base P' dans la moitié de l'arbre enracinée en w', de la même manière qu'il l'a effectué pour la première moitié. L'agent collecte les chaînes $(s'_1, \ldots, s'_k)$ dans l'ordre des nœuds visités en ignorant les chaînes collectées dans les nœuds déjà visités lors de l'exécution du tour P'. Après avoir complété le tour P' au nœud w', l'agent fait la concaténation μ de toutes les chaînes $(s'_1, \ldots, s'_k)$ qui représente l'étiquette normalisée de l'autre agent. Si μ est lexicographiquement plus petite que l'étiquette normalisée λ^* de l'agent alors l'agent reste immobile dans w' et s'arrête. Dans le cas contraire, il se rend à w et s'arrête.*(fin de l'algorithme)*

Théorème 4.4.1 `RV-arbre` *est un algorithme qui permet à deux agents de faire le rendez-vous dans des arbres arbitraires. Il fonctionne en temps $O(n)$ pour les arbres de taille n et utilise des tableaux blancs de taille $O((\log L)/n)$, où $\{1, \ldots, L\}$ est l'ensemble des étiquettes que peuvent avoir les agents.*

Preuve : Le théorème est évident si l'un des cas 1, 2 ou 3 s'est produit (dans ce cas, les agents n'ont besoin de rien écrire sur les tableaux blancs) et par conséquent, nous assumons le cas 4 dans lequel l'arbre est symétrique.

Nous commençons par prouver que l'algorithme `RV-arbre` est correct. Considérons qu'un agent A visite l'extrémité u de l'arête centrale la plus proche de sa position initiale, via le chemin le plus court (après avoir terminé au complet son premier tour de base). Si le tableau blanc associé à u n'est pas vide alors l'autre agent B a déjà commencé son tour de base P et par conséquent a déjà commencé à écrire ses chaînes s_i sur les tableaux blancs associés aux nœuds de la moitié de l'arbre enraciné en u. Dans ce cas, l'agent A écrit ses propres chaînes sur les tableaux blancs associés aux nœuds de l'autre moitié de l'arbre. Dans le cas contraire, l'agent A commence à écrire ses chaînes sur les tableaux blancs associés aux nœuds de la moitié de l'arbre enraciné en u. Dans ce cas lorsque l'agent B arrivera plus tard dans u, il verra que le tableau blanc associé à ce nœud n'est pas vide et donc écrira ses propres chaînes sur les tableaux blancs associés aux nœuds de l'autre moitié de l'arbre. Par conséquent, les agents écrivent toujours leurs chaînes dans des moitiés de l'arbre qui sont différentes. Dans le cas particulier où les agents vont simultanément à la même extrémité de l'arête centrale après avoir terminé leur premier tour de base, le rendez-vous est donc réalisé. Puisque le nombre de chaînes que chaque agent possède est exactement le même que le nombre de nœuds de la moitié de l'arbre, il écrira toutes ses chaînes dans des nœuds différents.

Après avoir fini son tour de base P, l'agent A se dirige vers l'autre extrémité w' de l'arête centrale. Si le tableau blanc associé à w' est vide, l'agent A reste immobile dans ce nœud et attend l'arrivée de l'agent B. Cette rencontre est assurée car l'agent B va soit rejoindre le nœud w' après avoir fini son tour de base, soit en voyant que le tableau blanc associé à w n'est pas vide, il se dirigera vers le nœud w' pour débuter son propre tour de base P'. Si le tableau blanc associé à w' nest pas vide, cela signifie que

l'agent B à déjà commencé à écrire ses chaînes dans la moitié de l'arbre enraciné en w'. Puisque l'agent A lira ces chaînes exactement dans l'ordre dans lequel elles sont écrites, il va récupérer l'étiquette normalisée de l'agent B. L'agent B va à son tour récupérer plus tard l'étiquette normalisée de l'agent A. Selon l'algorithme, les deux agents se rencontreront à l'extrémité de l'arête centrale qui correspond à la racine de la moitié de l'arbre qui contient l'étiquette la plus petite lexicographiquement.

Le temps du rendez-vous peut être borné par la somme des temps de l'agent qui a était activé plus tard : le temps du premier tour de base qui est $(2n-2)$, le temps pour rejoindre l'extrémité de l'arête centrale la plus proche de la position initiale qui est $(n/2)$, le temps du tour P qui est $(2k-2=n-2)$, le temps du tour P' qui est $(2k-2=n-2)$ et enfin deux traversées de l'arête centrale qui est un temps constant. Cette somme appartient évidemment à $O(n)$.

La taille des tableaux blancs sufisante pour exécuter l'algorithme `RV-arbre` est au plus proportionnel à $\lceil a/k \rceil$, où a est la taille de l'étiquette normalisée la plus longue. Puisque $k = n/2$, cette taille est de l'ordre de $O((\log L)/n)$, où $\{1, \ldots, L\}$ est l'ensemble des étiquettes que peuvent avoir des agents.

□

Notre prochain résultat montre que la borne établie dans le Théorème 4.4.1 sur la taille des tableaux blancs et permettant un rendez-vous en temps linéaire dans la classe des arbres est optimale. En fait, la borne inférieure suivante est valide même pour les algorithmes de rendez-vous exécutés dans toutes les lignes.

Théorème 4.4.2 *Tout algorithme de rendez-vous qui fonctionne en temps $O(n)$ dans toutes les lignes de taille n requiert des tableaux blancs de taille $\Omega((\log L)/n)$, où $\{1, \ldots, L\}$ est l'ensemble des étiquettes que peuvent avoir les agents.*

Preuve : Soit $n = 2k$ un entier positif pair et considérons une ligne de taille n avec une numérotation de ports dans laquelle les ports aux extrémités de chaque arête sont identiques. Remarquons que cette numérotation est symétrique par rapport à l'axe de symétrie passant à travers l'arête centrale de la ligne. Pour tout nœud v, nous dénotons par $p(v)$ l'image symétrique du nœud v par rapport à son axe de symétrie (voir figure 4.2).

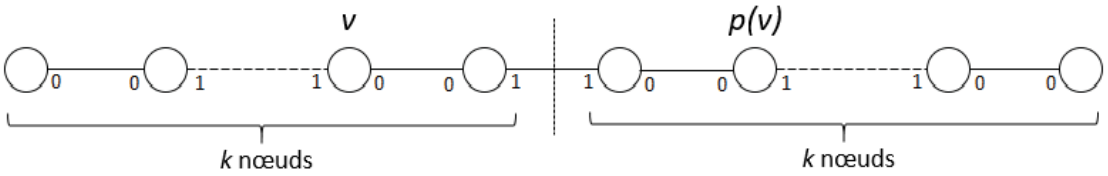

FIGURE 4.2 – Une ligne symétrique dont les extrémités de chaque arête possèdent des numéros de ports identiques

Supposons qu'un certain algorithme A résout le problème du rendez-vous dans cette ligne en temps cn, où c est un entier positif constant. Soient x la taille des tableaux blancs associés aux nœuds de la ligne et S l'ensemble de toutes les séquences binaires de taille au plus x. Considérons maintenant un agent qui exécute l'algorithme A et dont la position initiale est l'une des deux extrémités de la ligne. Le comportement de l'agent durant chaque ronde dépend seulement de son étiquette, du numéro de la ronde et du contenu actuel du tableau blanc associé au nœud visité. Ce comportement consiste en l'écriture ou non d'un message sur le tableau blanc du nœud visité (possiblement après avoir précédemment effacé le message en cours) et en restant immobile dans le nœud actuel ou en se déplaçant vers un nœud voisin.

Nous définissons une *exécution prévisible* de l'algorithme A par un agent, comme une exécution dans laquelle à chaque ronde, le contenu du tableau blanc associé au nœud v actuellement visité est le même que le contenu du tableau blanc associé au nœud de la

paire $\{v, p(v)\}$ la plus récemment visitée par l'agent, immédiatement après sa dernière visite, ou est vide, si l'agent n'a jamais précédemment visité l'un des nœuds v ou $p(v)$.

Remarquons que dans une exécution prévisible, le comportement de l'agent à chaque ronde dépend seulement de son étiquette et du numéro de la ronde, considérant que le contenu actuel du tableau blanc associé au nœud visité peut être calculé par l'agent sur la base de son histoire.

Plus formellement, un *comportement d'une exécution prévisible* peut être représenté par la fonction $f : \{1, \ldots, cn\} \longrightarrow \{-1, 0, 1\} \times S$, que nous définissons comme suit. Si $f(i) = (z, s)$ alors l'agent écrit la séquence s sur le tableau blanc associé au nœud qu'il visite à la ronde i et demeure immobile si $z = -1$, ou il prend le port z dans les autres cas. Le comportement d'une exécution prévisible d'un agent dépend uniquement de son étiquette.

Supposons que le nombre total de comportements d'une exécution prévisible est inférieur à L, donc $(3 \cdot 2^{x+1})^{cn} < L$. Par conséquent, il existe deux étiquettes différentes ℓ_1 et ℓ_2 dans $\{1, \ldots, L\}$ pour lesquelles les agents qui les possèdent ont un comportement d'une exécution prévisible identique. Considérons que ces deux agents exécutent simultanément l'algorithme A à partir des deux extrémités de la ligne.

Par induction sur le nombre de rondes, les propriétés suivantes sont satisfaites :

- L'exécution de l'algorithme A est une exécution prévisible pour chaque agent ;

- À chaque ronde r, les positions actuelles des agents sont symétriques par rapport à l'axe de symétrie qui traverse l'arête centrale de la ligne (figure 4.2), et les contenus des tableaux blancs associés aux nœuds visités par les agents à la ronde r sont identiques.

Par conséquent, le rendez-vous ne peut pas se produire, ce qui est une contradiction. Ceci implique que $(3 \cdot 2^{x+1})^{cn} \geq L$ donc $cn \cdot (x + \log 6) \geq \log L$. D'où $x = \Omega((\log L)/n)$.

□

Les théorèmes 4.4.1 et 4.4.2 impliquent le corollaire suivant :

Corollaire 4.4.1 *L'algorithme* `RV-arbre` *effectue le rendez-vous dans chaque arbre de taille n en temps optimal $O(n)$ et utilise la taille optimale $\Theta((\log L)/n)$ des tableaux blancs pour ce temps optimal, où $\{1, \ldots, L\}$ est l'ensemble des étiquettes que peuvent avoir les agents.*

4.5 Conclusion

Dans ce chapitre, nous avons présenté un algorithme qui permet de faire le rendez-vous déterministe dans n'importe quel arbre de taille n. Cet algorithme fonctionne en temps $O(n)$ qui est aussi l'ordre du temps de l'exploration de ce graphe. Nous avons montré que la taille des tableaux blancs appartient à $O((\log L)/n)$ et nous avons prouvé qu'il n'existe aucun algorithme déterministe qui permet de faire le rendez-vous dans la classe des lignes de longueur n en temps $O(n)$ avec des tableaux blancs de taille moins que $\Omega((\log L)/n)$.

Pour les graphes arbitraires de taille n, les auteurs de [36] ont énoncé et prouvé que l'algorithme qu'ils ont fourni permet de résoudre le problème d'élection entre k agents en temps $O(k \cdot m^2)$ et avec des tableaux blancs de taille $O(\log n)$, où m est le nombre d'arêtes du graphe. Si on utilise le même algorithme de [36] pour résoudre le problème du rendez-vous pour $(k = 2)$ agents, il suffit d'augmenter la taille des tableaux blancs de $O((\log L)/n)$ pour permettre aux agents d'écrire leurs étiquettes et par conséquent ils peuvent se rencontrer en temps $O(n^2)$ tout en utilisant des tableaux blancs de taille $O(\log n + \frac{\log L}{n})$ (Rappelons que le meilleur temps d'exploration d'un graphe est polynomial en la taille du graphe [91]).

Chapitre 5

Rencontre déterministe dans le plan utilisant le reniflement

Dans ce chapitre, nous nous intéressons au problème de la rencontre dans le plan entre deux agents mobiles qui utilisent l'odorat comme moyen de communication. Un agent dégage une odeur qui peut être sentie par l'autre. Nous supposons que les agents ont des capteurs chimiques qui leur permettent d'estimer la distance qui les séparent mais sans connaître la direction où ils se trouvent. L'intensité de l'odeur diminue avec la distance ce qui permet aux agents de faire leurs estimations.

Ce chapitre est structuré de la manière suivante. Nous commençons par la description des deux modèles que nous avons étudiés et par l'énoncé du problème. Nous présentons ensuite notre contribution en décrivant tous les résultats obtenus. Le reste du chapitre est consacré à présenter quelques définitions, les algorithmes et les preuves d'exactitude ainsi que les preuves d'optimalité des résultats.

5.1 Modèles et problèmes

Deux agents mobiles qui se trouvent initialement dans des endroits différents d'un plan doivent se rencontrer. Leur apparition peut être au même moment ou à des instants différents. Les agents sont modélisés par des disques de diamètre 1 et la rencontre se produit lorsque ces disques se touchent ; en d'autres termes, la rencontre est réalisable au moment où les centres des agents se trouvent à la distance 1. Cette manière de formuler le problème de la rencontre dans le plan est équivalente au problème de l'approche [43] où les agents sont modélisés en tant que points qui se déplacent dans le plan et l'approche est définie comme étant une rencontre qui se réalise lorsqu'un point se trouve à une distance au plus 1 par rapport l'autre.

Nous nous intéressons aux algorithmes déterministes qui permettent de réaliser cette tâche. Si les agents sont anonymes (identiques) et s'ils commencent simultanément l'exécution d'un certain algorithme déterministe alors ils vont tracer des trajectoires identiques et par conséquent ne pourront jamais se rencontrer. Pour briser cette symétrie, nous supposons que les agents possèdent des étiquettes différentes sous forme d'entiers positifs. Chaque agent connaît sa propre étiquette qu'il peut utiliser comme un paramètre dans l'exécution de l'algorithme mais il ne connaît pas l'étiquette de l'autre agent.

Les agents sont équipés de boussoles qui montrent les directions cardinales et possèdent des horloges synchronisées. L'adversaire place chaque agent à un certain point du plan, et possiblement à des moments différents, de manière à ce que la distance initiale entre les centres des disques soit supérieure à 1, c'est à dire que les disques sont disjoints ; autrement la rencontre est considérée comme réalisée immédiatement. Une horloge est activée au moment de l'apparition de son propriétaire dans le plan. Chaque agent effectue une série de mouvements ; un mouvement peut être soit une immobilisation pour un certain temps ou pour toujours dans un endroit précis, soit un déplacement selon une

direction et pendant une durée déterminée. Dans notre étude, les agents se déplacent à la même vitesse constante que nous avons normalisée à 1.

Nous supposons que les agents ont des capteurs qui leur permettent d'estimer la distance qui les séparent (définie comme la distance entre les centres des disques) mais ils n'ont aucune idée dans quelle direction l'autre agent se trouve. D'une manière plus précise, si un agent A estime que la distance qui le sépare de l'autre est d alors ce dernier peut se trouver dans un point arbitraire du périmètre du cercle de rayon d et dont le centre est la position actuelle de l'agent A. Nous considérons deux modèles d'estimation. Dans les deux modèles, un agent lit son capteur au moment de son apparition dans le plan et à la fin de chaque mouvement qu'il effectue. Ces lectures (actuelle et précédentes) permettent à l'agent de décider de son prochain déplacement. De plus, la lecture du capteur indique à l'agent si l'autre est déjà présent dans le plan.

Si les agents exécutent un algorithme de rencontre dans le *modèle monotone* alors chacun d'eux peut connaître, pour deux lectures en deux temps différents t_1 et t_2, si la distance qui les séparent à t_1 était plus petite, égale ou plus grande que celle estimée à t_2. En revanche, dans *le modèle binaire* qui est plus faible que le modèle monotone, chaque agent peut savoir à n'importe quelle lecture si la distance qui les séparent est inférieure ou supérieure ou égale à ρ, pour une valeur réelle $\rho > 1$ mais qui est inconnue par les agents. Nous supposons que $\rho > 1$ parce que les agents se trouveront toujours à une distance supérieure à 1 avant de se toucher et par conséquent supposer que $\rho \leq 1$ ne serait d'aucune utilité pour la détection. Un tel mécanisme d'estimation de la distance peut être mis en oeuvre en utilisant par exemple des capteurs chimiques. Chaque agent émet une certaine substance chimique (*odeur*) et le capteur de l'autre la détecte (la *renifle*). Si à un certain moment l'agent se trouve tout seul dans le plan, la lecture de son capteur est 0. Dans le cas contraire, cette lecture est un nombre positif dans le cas du modèle monotone ; si les agents se trouvent dans le modèle binaire alors cette lecture

indique la valeur 1 si la distance qui séparent les agents est inférieure à ρ, ou la valeur 0 si elle est d'au moins ρ. Il faut noter que l'intensité de l'odeur diminue avec la distance.

Dans le modèle monotone, nous supposons que le capteur est idéalement précis : il peut mesurer n'importe quelle variation d'intensité de l'odeur et par conséquent peut comparer les distances à deux lectures différentes (Le nom *monotone* provient du fait que l'intensité de l'odeur précisément captée par l'agent est une fonction strictement décroissante de la distance. Nous ne supposons rien d'autre par rapport à cette fonction : les agents ne peuvent rien apprendre de plus, comme par exemple la valeur exacte de la distance qui les sépare). Par contre dans le modèle binaire, nous supposons seulement que le capteur peut détecter l'odeur en dessous d'une certaine distance sans pouvoir mesurer son intensité et il ne peut rien détecter au delà de cette distance.

Il faut mentionner que notre modèle monotone est similaire au modèle de détection de la distance par des agents mobiles de l'article [35]. Les auteurs de cet article se sont intéressés à résoudre le problème de la rencontre dans le réseau tandis que notre étude touche la rencontre dans le plan pour des agents équipés de boussoles. De plus, la détection de la distance dans [35] se fait après chaque ronde et dans notre cas elle se fait à des moments décidés par les agents. Ces différences sont importantes et mèneront à l'élaboration d'un algorithme plus efficace dans le cas de notre modèle.

5.2 Notre contribution

Nous montrons l'impact des deux manières de capter l'odeur sur le coût de la rencontre ; ce coût représente la distance totale parcourue par les deux agents jusqu'à la réalisation de la rencontre.

Pour le modèle monotone, nous présentons un algorithme qui permet de faire la rencontre au coût $O(D)$, où D est la distance qui sépare les positions initiales des deux agents. Ce coût est optimal.

Pour le modèle binaire nous montrons que, si les agents commencent à une distance inférieure à ρ, c'est à dire lorsqu'ils captent mutuellement leur présence à leur apparition alors la rencontre peut être réalisée au coût $O(\rho \log \lambda)$, où λ est la plus grande étiquette des deux agents. Nous montrons que ce coût ne peut pas être amélioré en général. En effet nous prouvons que, pour une certaine distance initiale inférieure à ρ et pour certaines étiquettes des agents, le temps $\Omega(\rho \log \lambda)$ est nécessaire pour faire la rencontre dans le modèle binaire.

Finalement, nous constatons, dans le modèle binaire, que si les agents démarrent à une distance $\alpha\rho$, où $\alpha > 1$ est une constante, alors le reniflement n'amène aucun ajout, c'est à dire que le coût optimal de la rencontre en pire cas est du même ordre de grandeur que celui de la rencontre sans habileté de reniflement.

Nos résultats montrent une séparation entre l'efficacité de rencontre dans les deux modèles de reniflement. Supposons que les agents commencent à une distance $\rho/3$, alors le coût optimal de la rencontre dans le modèle monotone est $\Theta(\rho)$ tandis que dans le modèle binaire c'est $\Theta(\rho \log \lambda)$.

5.3 Terminologie et préliminaires

Dans cette section, nous présentons des notions et des termes que nous allons utiliser tout au long de ce chapitre.

La direction Nord-Sud est appelée la direction *verticale* et la direction Est-Ouest est appelée la direction *horizontale*. Nous disons que l'agent a se trouve au *Nord* de l'agent b si la ligne horizontale qui contient le centre de l'agent a se trouve au *Nord* de

la ligne horizontale qui contient le centre de l'agent b. De la même manière, on définit les positions *Sud*, *Est* et *Ouest* de l'agent a par rapport à l'agent b.

Nous disons que l'agent a se trouve à une *distance verticale* x de l'agent b si les lignes horizontales qui contiennent les centres des deux agents sont à la distance x. La *distance horizontale* est définie de la même manière.

5.4 Rencontre déterministe dans le modèle monotone

Dans cette section nous présentons un algorithme qui permet de réaliser la rencontre dans le modèle monotone au coût $O(D)$, où D est la distance qui sépare les positions initiales des agents. Ce coût est évidemment optimal puisque D représente une borne inférieure sur le coût de la rencontre.

Dans le modèle monotone, nous présentons deux instructions élémentaires : $lire(C)$ et $avancer(card, x)$. L'instruction $lire(C)$ permet à l'agent de stocker la valeur actuelle de l'intensité de l'odeur détectée par son capteur dans la variable C. Rappelons que la valeur du capteur est 0 si l'agent est seul dans le plan et dans le cas contraire, cette valeur vaut un réel positif. L'instruction $avancer(card, x)$, où $card$ est l'une des directions cardinales (N, E, S, O) et x est un réel positif, permet à un agent de se déplacer dans la direction $card$ pendant le temps x. Considérant que la vitesse de l'agent est 1, ceci signifie qu'il parcourt une distance x dans la direction $card$.

D'une manière générale, voici comment notre algorithme déterministe fonctionne. Au début, l'agent nommé a lit son capteur. Si sa valeur est 0 alors ceci signifie que l'autre agent b n'est pas encore dans le plan. L'agent a va briser la symétrie en restant immobile sur sa position initiale en attendant éventuellement que l'agent b le trouve.

Une fois mis dans le plan, l'agent b doit réaliser que l'autre agent ne bouge pas et il doit le trouver en se plaçant en premier lieu à une distance verticale d'au plus 1 par rapport à l'agent a et ensuite à une distance horizontale d'au plus 1 de a, ce qui va permettre de réaliser la rencontre.

Si les lectures initiales des capteurs des deux agents sont positives, cela signifie que les deux sont placés dans le plan simultanément. Dans ce cas, la seule manière de briser la symétrie est d'utiliser les étiquettes des agents, qui sont différentes par hypothèse. Les agents doivent réaliser qu'ils sont dans cette situation difficile et essayer de s'approcher, premièrement dans la direction verticale et ensuite dans la direction horizontale. Ceci est fait par un déplacement vers le Nord et vers le Sud pour l'approche verticale et vers l'Est et l'Ouest pour l'approche horizontale. Ce déplacement ne dépend que de la valeur des bits de l'étiquette de l'agent. Au premier bit où les représentations binaires de leurs étiquettes diffèrent, ou au $(m+1)$-ème bit de la plus longue représentation binaire dans le cas où la plus courte représentation binaire est un préfixe de taille m de la plus longue, la symétrie entre les agents sera brisée, ils vont le réaliser en lisant leurs capteurs pour ensuite accomplir l'approche. Dans le but de conserver le coût en $O(D)$ (et non en $O(D+\log\lambda)$), les déplacements des agents correspondant aux bits consécutifs se rétrécissent par un facteur de 2 à chaque bit consécutif.

Nous allons maintenant procéder à une description détaillée de notre algorithme. Nous utiliserons les procédures élémentaires suivantes.

La première procédure sert à comparer les lectures précédentes du capteur avec la lecture actuelle et stocke le résultat de la comparaison dans la variable *compare*.

Procédure `Tester`

$P \leftarrow C$

$lire(C)$

Si $P < C$ **alors** $compare \leftarrow grand$

Si $P = C$ **alors** $compare \leftarrow égal$

Si $P > C$ **alors** $compare \leftarrow petit$

La deuxième procédure à pour objectif d'approcher l'autre agent soit dans la direction verticale soit dans la direction horizontale en se déplaçant par des pas de longueur prescrite x. Elle est utilisée lorsque l'agent réalise déjà que l'autre agent est au Nord (respectivement au Sud) ou il est à l'Est (respectivement à l'Ouest). Nous formulons la procédure pour le paramètre *card* qui peut prendre une des valeurs N, E, S ou O.

Procédure `Rapprocher` $(card, x)$

Tant que $compare = petit$ **faire**

    $avancer(card, x)$

    `Tester`

Notre prochaine procédure est appelée dans le cas où les agents apparaissent simultanément dans le plan pour leur permettre de briser la symétrie entre eux. Ceci est réalisé lorsque la valeur de la variable *compare* dans la procédure `Tester` devient différente de *égal*. Si les représentations binaires des étiquettes des agents diffèrent à un certain bit (voir figure 5.1), le bris de la symétrie se réalise lorsque les agents commencent à traiter

ce bit. Si une représentation binaire de taille m_1 est un préfixe de l'autre (voir figure 5.1) alors le bris de la symétrie se produit lorsque l'agent avec l'étiquette la plus longue traite le $(m_1 + 1)$-ème bit de son étiquette et l'agent avec l'étiquette la plus courte demeure immobile dans sa position actuelle.

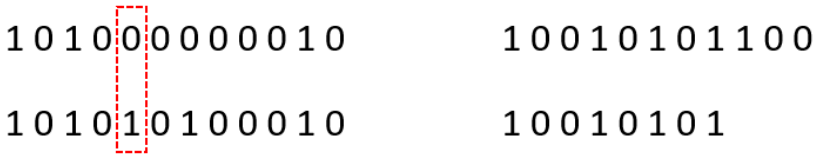

FIGURE 5.1 – À gauche, les étiquettes ont la même longueur mais elles diffèrent pour la première fois au 5-ème bit. À droite, l'étiquette en bas est un préfixe de celle en haut

Notons qu'il y a deux tentatives d'obtenir la valeur de la variable *compare* différente de *égal* lors du traitement de chaque bit de la représentation binaire de l'étiquette. Ceci est nécessaire dans le cas particulier lorsque les représentations binaires des étiquettes des agents diffèrent à un seul bit, par exemple celui avec l'index j, la distance verticale entre les agents est $\frac{1}{2^j}$ et l'agent dont le j-ème bit est 1 se trouve au Sud de l'agent dont le j-ème bit est 0. Dans ce cas particulier, une seule tentative de briser la symétrie passerait inaperçue : les agents vont changer leurs positions dans la direction verticale et la valeur de la variable *compare* va garde la valeur *égal*. Cette procédure est exécutée par un agent qui possède l'étiquette ℓ et dont la représentation binaire est $(c_1 \dots c_m)$.

Procédure `Danser`

$i \leftarrow 1$

Tant que ($compare = égal$ ET $i \leq m$) **faire**

  Si $c_i = 1$ **alors**

    $avancer(N, \frac{1}{2^i})$

    `Tester`

    Si $compare = égal$ **alors**

      $avancer(N, \frac{1}{2^i})$

      `Tester`

  sinon

    $avancer(S, \frac{1}{2^i})$

    `Tester`

    Si $compare = égal$ **alors**

      $avancer(S, \frac{1}{2^i})$

      `Tester`

  $i \leftarrow i + 1$

Après avoir énoncé les procédures élémentaires, nous allons maintenant décrire les deux procédures principales de notre algorithme qui vont être appelées une après l'autre. Le but de la première d'entre elle est d'amener les agents à une distance verticale d'au plus 1 et le but de la deuxième est de les amener à une distance horizontale d'au plus 1.

Dans la procédure `ApprocheVerticale`, le dernier agent qui apparaît, ou les deux agents ensemble dans le cas d'un départ simultané, réalisent en premier lieu lequel de ces deux cas se produit. Pour cela, ils font la chose suivante. L'agent, appelons-le a,

se déplace vers le Nord par 1. Si la distance qui le sépare de l'autre agent diminue, il conclut que ce dernier est immobile et alors il l'approche par des pas de longueur 1/2 jusqu'à aboutir à une distance verticale d'au plus 1. Par contre si la distance qui le sépare de l'autre agent augmente, il conclut que ce dernier est immobile et alors il retourne premièrement à sa position initiale (se déplace vers le Sud par 1) et ensuite il approche l'autre agent par des pas de longueur 1/2 jusqu'à aboutir à une distance verticale d'au plus 1.

Maintenant, si la distance ne change pas, la situation demeure floue : il est possible que le départ des deux agents est simultané et que les deux agents se sont déplacés par 1 vers le Nord, mais une autre situation peut se présenter et peut produire le même résultat ; c'est le cas ou l'autre agent se trouve immobile dans sa position actuelle à une distance 1/2 au Nord de l'agent a avant le déplacement de ce dernier (voir figure 5.2).

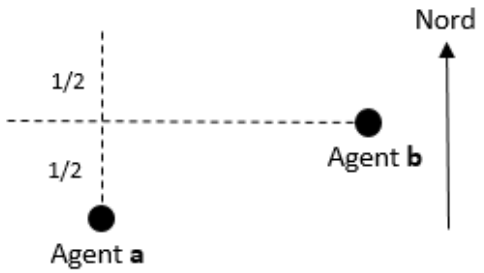

FIGURE 5.2 – La distance ne change pas malgré qu'un des agents est immobile

Pour éliminer cette ambiguïté, l'agent a se déplace au Nord par 1 une deuxième fois. L'agent a ne peut pas diminuer sa distance de b après ce déplacement. Si la distance augmente alors il revient à sa position précédente et approche l'agent b comme auparavant. D'autre part, si la distance ne change pas après ce mouvement alors l'agent a conclut qu'il se trouve dans la situation d'un départ simultané ; il exécute la procédure

`Danser`, à la fin de laquelle la valeur de la variable *compare* est différente de *égal*, sauf dans le cas particulier où la représentation binaire de l'étiquette de l'agent est un préfixe de l'autre agent. Donc, à la fin de l'exécution de la procédure `Danser` par un agent, deux situations sont possibles :

1. l'agent a traité son j-ème bit qui est le premier bit où son étiquette diffère de celle de l'autre agent, ou
2. la représentation binaire de l'étiquette de l'agent est un préfixe de celle de l'autre et par conséquent l'agent demeure immobile après avoir traité le dernier bit de son étiquette.

Ce qui va briser la symétrie.
Supposons premièrement que c'est la situation 1 qui s'est réalisée alors il y aura deux cas.

Si, à la fin de l'exécution de la procédure `Danser`, la valeur de *compare* est *petit* alors l'agent dont le j-ème bit est 1 était au Sud de l'autre agent *avant* le dernier déplacement. Les agents reviennent en arrière d'une distance de $\frac{1}{2^j}$ et ensuite se rapprochent l'un de l'autre : l'agent dont le j-ème bit est 1 se déplace vers le Nord et l'agent dont le j-ème bit est 0 se déplace vers le Sud. Si, à la fin de l'exécution de la procédure `Danser`, la valeur de *compare* est *grand* alors l'agent dont le j-ème bit est 1 se trouve au Nord de l'autre agent *après* le dernier déplacement. Dans ce cas, les agents n'ont pas besoin de revenir en arrière mais ont besoin simplement de se rapprocher l'un de l'autre : l'agent dont le j-ème bit est 1 se déplace vers le Sud et l'agent dont le j-ème bit est 0 se déplace vers le Nord. Dans les deux cas, les pas par lesquels les agents se rapprochent (grâce à la procédure `Rapprocher`) sont maintenant de taille 1/4 au lieu de 1/2 parce que les agents performent l'approche simultanément.

Dans la situation 2, l'agent reste immobile pour toujours et l'autre agent essaye de l'approcher de la même manière que dans la situation 1.

Procédure `ApprocheVerticale`

$sim \leftarrow faux$

$inerte \leftarrow faux$

$avancer(N, 1)$

`Tester`

Si $compare = petit$ **alors**

    `Rapprocher` $(N, \frac{1}{2})$

sinon

    Si $compare = grand$ **alors**

        $avancer(S, 1)$

        `Rapprocher` $(S, \frac{1}{2})$

    sinon

        $avancer(N, 1)$

        `Tester`

        Si $compare = grand$ **alors**

            $avancer(S, 1)$

            `Rapprocher` $(S, \frac{1}{2})$

        sinon $(*compare = égal*)$

Procédure `ApprocheVerticale` *(suite)*

$sim \leftarrow vrai$

`Dance`

Si *compare* =*égal* **alors**

Rester immobile pour toujours

$inerte \leftarrow vrai$

sinon

$j \leftarrow i - 1$

Si $compare = petit$ **alors**

Si $c_j = 1$ **alors**

$avancer(S, \frac{1}{2^j})$

`Rapprocher` $(N, \frac{1}{4})$

sinon

$avancer(N, \frac{1}{2^j})$

`Rapprocher` $(S, \frac{1}{4})$

sinon (* $compare = grand$ *)

Si $c_j = 1$ **alors**

`Rapprocher` $(S, \frac{1}{4})$

sinon

`Rapprocher` $(N, \frac{1}{4})$

La procédure `ApprocheHorizontale` est appelée après la procédure `ApprocheVerticale` et elle utilise les variables globales *inerte*, *sim* et j dont les valeurs leur sont assignées dans la procédure `ApprocheVerticale`. La procédure `ApprocheHorizontale` (exécutée

uniquement lorsque la variable *inerte* a la valeur $faux$) est plus simple à exécuter parce que, lorsqu'elle est appelée, l'agent possède deux informations importantes. Premièrement, il sait déjà que sa distance verticale de l'autre agent est d'au plus 1 et par conséquent les agents ne peuvent pas se croiser horizontalement sans se toucher (la rencontre). Deuxièmement, l'agent sait si le départ était simultané, ou si l'autre agent a apparu avant lui et par conséquent il va rester immobile. Cette information est stockée dans la variable booléenne *sim* qui a la valeur *vrai* dans la procédure `ApprocheVerticale` si et seulement si le départ des agents est simultané. De plus, si le départ était simultané, l'agent connaît déjà le premier bit de son étiquette qui diffère de celui de l'étiquette de l'autre : l'index de ce bit est j.

Par conséquent, si $sim = faux$ alors l'agent se rapproche de l'autre agent (immobile) de la même manière qu'auparavant ; par contre, si $sim = vrai$ alors les agents se rapprochent l'un de l'autre horizontalement en se dirigeant soit vers l'Est soit vers l'Ouest, dépendemment de la valeur du j-ème bit de leur étiquette, et ceci jusqu'à la réalisation de la rencontre.

Procédure `ApprocheHorizontale`

Si $(\neg inerte)$ **alors**

Si $sim = faux$ **alors**

$avancer(E, 1)$

`Tester`

Si $compare = petit$ **alors** `Rapprocher` $(E, 1)$

sinon

$avancer(W, 1)$

`Rapprocher` $(W, 1)$

sinon

Si $c_j = 1$ **alors**

$avancer(E, 1)$

`Tester`

Si $compare = petit$ **alors** `Rapprocher` $(E, 1)$

sinon

$avancer(W, 1)$

`Rapprocher` $(W, 1)$

sinon

$avancer(W, 1)$

`Tester`

Si $compare = petit$ **alors** `Rapprocher` $(W, 1)$

sinon

$avancer(E, 1)$

`Rapprocher` $(E, 1)$

Nous sommes maintenant prêts à présenter notre algorithme principal qui permet de résoudre le problème de la rencontre dans le plan dans le modèle monotone . Il est exécuté par un agent qui possède une étiquette ℓ (qui intervient dans l'exécution de la procédure `Danser`) ; l'algorithme est interrompu lorsque les agents se touchent (lorsque la distance qui sépare leur centre est égale à 1) puisque la rencontre est alors accomplie. Nous prouvons que la rencontre se produit toujours à la fin de l'exécution de cet algorithme.

Algorithme `RencontreCapteurPrécis`

$(c_1 \dots c_m) \leftarrow$ représentation binaire de ℓ

$lire(C)$

Si $C = 0$ **alors**

    rester immobile pour toujours

sinon

    `ApprocheVerticale`

    `ApprocheHorizontale`

Théorème 5.4.1 *La rencontre entre deux agents, qui sont arbitrairement placés dans le plan et qui exécutent l'algorithme* `RencontreCapteurPrécis` *dans le modèle monotone, se réalise à la fin de l'exécution de cet algorithme. Si les agents se trouvent à une distance initiale D alors la rencontre se fait en temps $O(D)$ après l'apparition du dernier agent.*

Preuve : Soient α_1 et α_2 les représentations binaires des étiquettes des agents. Considérons deux cas.

Cas 1. Ni α_1 est un préfixe de α_2 ni α_2 est un préfixe de α_1.

Sous-cas 1.1. Les agents ne commencent pas simultanément.

Dans ce cas, le premier agent, appelons-le b, obtient la valeur initiale 0 de son capteur et par conséquent reste immobile pour toujours. L'autre agent, appelons-le a, se déplace vers le Nord par 1. Si sa distance de b diminue, ce dernier se trouve au Nord de l'agent a avant ce déplacement et il est soit encore au Nord de a après le déplacement soit il est au Sud de a, à une distance verticale inférieure à 1/2. Dans les deux cas, la procédure `Rapprocher` $(N, \frac{1}{2})$ met l'agent a à une distance verticale au plus 1 de l'agent b. Si la distance qui sépare a de b augmente après le déplacement alors soit a était au Sud de b à une distance verticale inférieure à 1/2 avant le déplacement et il se trouve au Nord de b après le déplacement, soit l'agent a était déjà au Nord de b avant le déplacement. Dans le premier cas, l'agent a retourne à sa position précédente (déplacement au Sud par 1) et se trouve à nouveau au Sud de b à une distance verticale inférieure à 1/2. La procédure `Rapprocher` $(S, \frac{1}{2})$ permet de faire un déplacement vers le Sud de longueur 1/2 et par conséquent l'agent a terminera la procédure `ApprocheVerticale` à une distance verticale inférieure à 1 de l'agent b. Dans le deuxième cas, après son retour à sa position précédente, l'agent a sera encore au Nord de b et par conséquent la procédure `Rapprocher` $(S, \frac{1}{2})$ se terminera en positionnant les agents à une distance verticale inférieure à 1.

Si, après le premier déplacement, la distance entre les agents n'a pas changé alors l'agent b était à une distance verticale de 1/2 au Nord de l'agent a avant le déplacement de ce dernier, et après le déplacement il est au Sud de l'agent a à une distance verticale de 1/2. Un autre déplacement au Nord de 1 va augmenter la distance entre les agents. Ensuite l'agent a retourne à sa position précédente et la procédure `Rapprocher` $(S, \frac{1}{2})$ se terminera après deux déplacements et les agents se trouvent alors à une distance verticale exactement de 1/2.

Donc, l'agent a termine la procédure `ApprocheVerticale` à une distance verticale d'au plus 1 de l'agent b. D'une manière similaire, la procédure `ApprocheHorizontale` (effec-

tuée avec $sim = faux$) mettra l'agent a à une distance horizontale d'au plus 1 de l'agent b au même moment de la réalisation de la rencontre. Soient x la distance initiale entre les agents dans la direction verticale et y celle dans la direction horizontale. L'agent b parcourt la distance 0 et l'agent a parcourt une distance verticale d'au plus $x + 4$ et une distance horizontale d'au plus $y + 4$, d'où la distance totale parcourue est au plus $x + y + 8 \in O(x + y) = O(D)$, où D est la distance initiale entre les agents.

Sous-cas 1.2. Les agents commencent simultanément.

La première lecture du capteur est positive pour chaque agent. Puisque les agents commencent simultanément, chacun d'eux effectue un déplacement au Nord par 1, la variable *compare* est assignée à *égal* et ils effectuent un deuxième déplacement vers le Nord. Après ces deux déplacements, les deux agents réalisent que leurs départs étaient simultanés. Donc, ils affectent *vrai* à la variable *sim* et exécutent la procédure `Danser`. Soit j l'index du premier bit où les étiquettes des agents diffèrent. Cette procédure se termine pour les deux agents après le traitement du j-ème bit avec la valeur de la variable *compare* différente de *égal*. Si $compare = petit$ alors l'agent dont le j-ème bit est égal à 1 était au Sud de l'autre agent *avant* le dernier déplacement. Les agents font un retour en arrière avec une distance de $\frac{1}{2^j}$ dans le but de restaurer la situation qui était avant le dernier déplacement, puis se rapprochent l'un de l'autre de la manière suivante : l'agent dont le j-ème bit est égal à 1 se dirige vers le Nord et l'agent dont le j-ème bit est égal à 0 se dirige vers le Sud. Si $compare = grand$ alors l'agent dont le j-ème bit est égal à 1 se trouve au Nord de l'autre agent après le dernier déplacement. Maintenant, les agents n'ont pas besoin de faire un retour en arrière et ils vont essayer de se rapprocher l'un de l'autre comme suit : L'agent dont le j-ème bit est égal à 1 se dirige vers le Sud et l'agent dont le j-ème bit est égal à 0 se dirige vers le Nord. De la même manière que dans le cas précédent, les deux agents terminent l'exécution de la procédure `ApprocheVerticale` à une distance verticale d'au plus 1. La procédure `ApprocheHorizontale` (effectuée avec

$sim = vrai$) mettra les deux agents à une distance horizontale d'au plus 1 au même moment de la réalisation de la rencontre.

Il reste à éstimer le coût de la rencontre. Nous allons estimer la distance parcourue par chaque agent. Chaque agent parcourt une distance égale à 2 avant l'appel de la procédure `Danser`. Pendant l'exécution de cette procédure, chaque agent parcourt une distance d'au plus 1, dû au fait que la distance parcourue lors du traitement d'un bit est deux fois plus petite que celle parcourue lors du traitement du bit précédent. Soient x la distance initiale entre les agents dans la direction verticale et y celle dans la direction horizontale. Pendant l'approche verticale qui se fait après l'exécution de la procédure `Danser`, chaque agent parcourt une distance plus petite que $x + 1$ et une distance plus petite que $y + 1$ pendant l'approche horizontale. D'où la distance totale parcourue par un agent dans le cas d'un départ simultané est inférieure à $x + y + 5$. Ainsi, le coût total est inférieur à $2(x + y + 5) \in O(x + y) = O(D)$, où D est la distance initiale entre les agents.

Cas 2. α_1 est un préfixe de α_2 (le cas symétrique est traité de la même manière en interchangeant α_1 par α_2).

Sous-cas 2.1. Les agents ne commencent pas simultanément.

Les mêmes arguments présentés dans le sous-cas 1.1 montrent qu'à la fin de l'exécution de la procédure `ApprocheHorizontale`, la rencontre est réalisée. L'analyse du coût est similaire.

Sous-cas 2.2. Les agents commencent simultanément.

Comme dans le sous-cas 1.2, les deux agents réalisent après deux déplacements que leur départs étaient simultanés. Ils mettent la variable *sim* à *vrai* et exécutent la procédure `Danser`. L'agent avec l'étiquette α_1 dont la représentation binaire est de longueur m_1 complète cette procédure avec *comparer* $=$*égal*. Il met ensuite la variable *inerte* à *vrai* et reste immobile pour toujours. L'autre agent termine l'exécution de la procédure

`Danser` lorsqu'il traite le bit avec l'index $j = m_1 + 1$ de la représentation binaire de son étiquette. Le reste de l'argument est similaire à celui présenté dans le sous-cas 1.2 mais il se réfère uniquement à l'agent avec l'étiquette α_2. Les actions de cet agent dans la procédure `ApprocheVerticale` dépendent de la valeur de son j-ème bit (1 ou 0) comme auparavant. Finalement, l'agent exécute la procédure `ApprocheHorizontale` qui va le mettre à une distance horizontale d'au plus 1 par rapport à l'autre agent au même moment de la réalisation de la rencontre. L'analyse du coût est similaire au sous-cas 1.2.

□

5.5 Rencontre déterministe dans le modèle binaire

Dans cette section, nous présentons un algorithme qui permet à deux agents de se rencontrer au coût $O(\rho \log \lambda)$ dans le modèle binaire, où λ est l'étiquette la plus grande et ρ est la distance maximale sur laquelle les agents peuvent se détecter. Nous supposons que les agents se trouvent initialement à une distance plus petite que ρ, c'est à dire qu'à son état initial, chaque agent peut détecter l'autre lorsque les deux se trouvent ensemble dans le plan. Nous montrons aussi une borne inférieure $\Omega(\rho \log \lambda)$ sur le coût de la rencontre, pour certaines positions initiales qui se trouvent à une distance inférieure à ρ et pour certaines étiquettes des agents.

Dans le modèle binaire, nous utilisons deux instructions élémentaires : *vérifier*(C) et *avancer*$(card, x)$. L'instruction *vérifier*(C) permet à l'agent de stocker la valeur actuelle de l'intensité de l'odeur détectée par son capteur dans la variable C. Rappelons que la valeur du capteur est 0 si l'agent est seul dans le plan ou la distance qui le sépare de l'autre agent est au moins ρ et dans le cas contraire, cette valeur vaut 1. L'instruction *avancer*$(card, x)$, où *card* est l'une des directions cardinales (N, E, S, O) et x est un réel positif, permet à un agent de se déplacer dans la direction *card* pendant le temps x.

Considérant que la vitesse de l'agent est 1, ceci signifie qu'il parcourt une distance x dans la direction *card*.

À haut niveau, l'idée de notre algorithme est la suivante. Au début, l'agent lit son capteur. Si sa valeur est 0, ceci signifie que l'autre agent n'est pas encore dans le plan, ce qui permet à l'agent de briser la symétrie comme dans le modèle monotone. L'agent demeure immobile pour toujours et sera éventuellement trouvé par l'autre agent. Comme nous allons l'expliquer ci-dessous, la rencontre sera faite d'une manière très différente de celle vue dans le modèle monotone, à cause que la détection de l'autre agent dans le modèle binaire est moins précise.

Si les lectures initiales des capteurs des deux agents ont la valeur 1, cela signifie que les deux agents sont placés simultanément dans le plan. Dans ce cas, ils brisent la symétrie en utilisant leurs étiquettes de la manière suivante. Chaque agent effectue en premier lieu la transformation suivante de son étiquette ℓ : il remplace chaque bit 1 de la représentation binaire de son étiquette par 101010 et chaque bit 0 de cette représentation par 010101. Nous notons $T(\ell)$ la séquence de bits résultante par cette transformation. Il faut remarquer qu'avec une telle transformation appliquée à n'importe quelle séquence de bits, le résultat obtenu ne peut pas contenir une séquence de plus de deux 0 ou deux 1 consécutifs (voir figure 5.3)

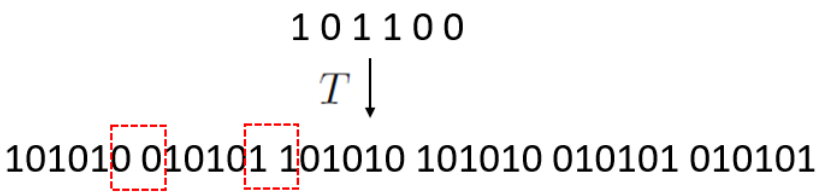

FIGURE 5.3 – Après la transformation, le nombre maximal de bits consécutifs égaux est 2 (deux 0 ou deux 1)

Initialement, un agent n'a aucun moyen de décider s'il a été placé dans le plan plus tard ou s'il a été placé avec l'autre agent dans le plan simultanément. Ainsi, lorsque la valeur lue initialement dans *vérifier*(C) est 1, l'agent effectue des actions qui lui permettent de perdre le contact avec l'autre agent, c'est à dire obtenir la lecture 0 dans *vérifier*(C). Perdre le contact arrivera toujours lorsqu'un agent se déplace et l'autre reste immobile. Ceci permettra aux agents de briser la symétrie : l'agent en mouvement qui a perdu le contact essayera de trouver l'autre agent qui est devenu immobile (dans le cas d'un départ arbitraire, l'agent qui effectuera la recherche est le dernier agent activé puisque le premier agent activé est resté immobile depuis son activation). Une fois la symétrie est brisée, l'agent qui se déplace réalise la rencontre en utilisant les lectures binaires de son capteur et les propriétés géométriques du plan.

Nous allons maintenant présenter une description détaillée de notre algorithme. Commençons par la première procédure dont le but est de permettre aux agents de perdre le contact entre eux, c'est à dire avoir la valeur de *vérifier*(C) à 0 pour les agents dans le cas d'un départ simultané ou pour le dernier agent activé dans le cas d'un départ arbitraire. Ceci est réalisable lorsque le déplacement se fait vers le Nord ou en demeurant immobile pendant des périodes de temps qui sont doublées après chaque déplacement selon les valeurs des bits des étiquettes transformées : lorsque le bit est 1, l'agent se déplace et lorsque c'est 0, il demeure immobile dans sa position actuelle. Pour des raisons techniques, les longues périodes d'immobilité des agents sont divisées en segments de longueur 1/2 après lesquelles l'agent en attente lit son capteur. Il sera prouvé que les agents s'éloignent éventuellement à une distance d'au moins ρ, ce qu'ils réaliseront en lisant leur capteur. La perte du contact se produit lorsqu'un agent se déplace vers le Nord et l'autre agent reste immobile.

Lorsque l'agent en mouvement perd le contact, il retourne vers le Sud par des petits pas jusqu'à ce que le contact soit rétabli de nouveau (*vérifier*(C) à la valeur 1).

Cette stratégie va permettre à cet agent de déterminer le point du plan à partir duquel le contact a été perdu avec une précision suffisante. La procédure `PerdreContact` est exécutée par un agent qui possède une étiquette ℓ et est appelée lorsque la valeur de C est 1.

Procédure `PerdreContact` (C)

$(c_1 \dots c_m) \leftarrow T(\ell)$

$d \leftarrow 1$

$principal \leftarrow faux$

Tant que $C = 1$ **faire**

    $i \leftarrow 1$

    Tant que $(C = 1$ **ET** $i \leq m)$ **faire**

        Si $c_i = 1$ **alors** $avancer(N, d)$

        sinon

            $k \leftarrow 1$

            Tant que $(C = 1$ **et** $k \leq 2d)$ **faire**

                Rester immobile pendant $1/2$

                $vérifier(C)$

                $k \leftarrow k + 1$

        $i \leftarrow i + 1$

        $vérifier(C)$

    $d \leftarrow 4d$

$j \leftarrow i - 1$

Si $c_j = 1$ **alors**

    $principal \leftarrow vrai$

    Rester immobile pendant 1

    Tant que $C = 0$ **faire**

        $avancer(S, \frac{1}{2})$

        $vérifier(C)$

La deuxième procédure est basée sur l'observation suivante. Considérons deux points X et A dans le plan tel que A se trouve au Nord de X. Supposons que la distance entre X et A est ρ. Soient p la ligne verticale qui passe par le point A et l'autre point B qui se trouve à une distance ρ de X (Cela inclut aussi le cas où A et B coïncident). Soit le point Z le milieu du segment AB. Donc Z se trouve sur la même ligne horizontale qui traverse X parce que le triangle XAB est isocèle en X (Voir figure 5.4).

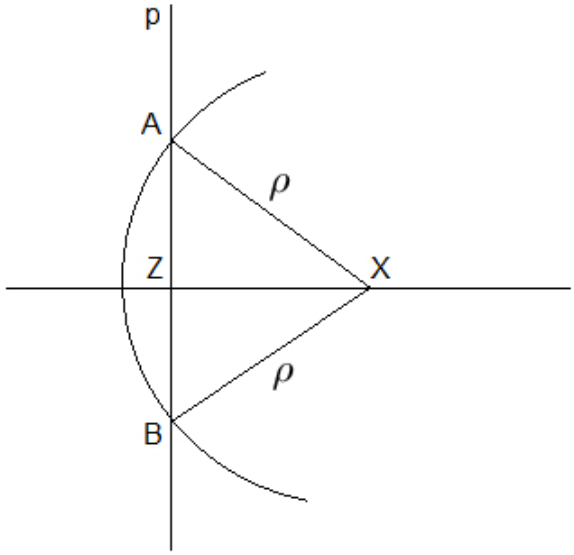

FIGURE 5.4 – Ingrédient géométrique pour la procédure `TriangleRecherche`

L'observation ci-dessus peut être utilisée pour construire la prochaine procédure, appelée après l'exécution de la procédure `PerdreContact`, lorsque (le centre d') un des agents, appelons-le a, coïncide avec le point A qui se trouve au Nord de l'agent immobile dont le centre coïncide avec le point X et de manière à ce que la distance entre X et A soit ρ. L'agent a commence la procédure avec la lecture 1 de son capteur. Il se déplace vers le Sud avec des pas de longueur $1/2$ (le nombre de pas est stocké dans un compteur t) jusqu'à ce que la lecture de son capteur devienne 0. À ce moment, le centre de a coïncide avec l'autre point B qui se trouve à la distance ρ du point X et qui est situé sur la ligne

verticale qui englobe le trajet de a. Ensuite, ce dernier revient en arrière en se dirigeant vers le Nord sur cette ligne verticale en parcourant la distance $\lceil t/2 \rceil \cdot \frac{1}{2}$. À la fin de ce déplacement, l'agent a va se trouver sur le point Z qui représente le milieu du segment AB (Voir figure 5.4). À ce moment, il va se déplacer horizontalement en zigzag dans les directions Est et Ouest par des sauts croissants dont les longueurs doublent à chaque fois, jusqu'à la rencontre de l'agent immobile. Considérant que les pas utilisés par l'agent lorsqu'il se déplaçait verticalement étaient suffisamment petits, les approximations sont suffisamment bonnes pour que la rencontre se produise éventuellement.

Procédure `TriangleRecherche` (C)

Si $principal = vrai$ **alors**

$t \leftarrow 0$

Tant que $C = 1$ **faire**

$avancer(S, \frac{1}{2})$

$vérifier(C)$

$t \leftarrow t + 1$

$avancer(N, \lceil t/2 \rceil \cdot \frac{1}{2})$

$d \leftarrow 1$

Répéter jusqu'à la rencontre

$avancer(E, d)$

$avancer(O, 2d)$

$avancer(E, d)$

$d \leftarrow 2d$

Notre algorithme principal pour le modèle binaire peut être maintenant formulé comme suit. Il est exécuté par un agent qui possède une étiquette ℓ qui va être utilisée

dans la procédure `PerdreContact`. L'exécution de cet algorithme est interrompu lorsque les agents se touchent, c'est à dire que la distance qui sépare leurs centres est égale à 1 et donc la rencontre est réalisée. Nous prouverons que ceci se passera toujours à la fin de l'exécution de l'algorithme.

Algorithme `RencontreAvecCapteurBinaire`

vérifier(C)

Si $C = 0$ **alors**

    Rester immobile pour toujours

sinon

    `PerdreContact` (C)

    `TriangleRecherche` (C)

Théorème 5.5.1 *La rencontre de deux agents, qui sont placés dans le plan à une distance initiale inférieure à ρ et qui exécutent l'algorithme* `RencontreAvecCapteurBinaire` *dans le modèle binaire, se réalise à la fin de l'exécution de cet algorithme. Cette rencontre se produit au coût $O(\rho \log \lambda)$, où λ est l'étiquette la plus grande.*

Preuve : Nous allons démontrer en premier lieu l'affirmation suivante.

Affirmation. Les agents se trouvent à une distance d'au moins ρ durant l'exécution de la procédure `PerdreContact`.

Dans le cas d'un départ non-simultané, la preuve de l'affirmation est simple : lorsque d dépasse ρ, l'agent actif qui se déplace toujours vers le Nord se trouve à une distance plus grande que ρ dans la direction verticale, donc la distance entre les agents dépasse ρ. Par conséquent, nous pouvons supposer que les agents commencent simultanément. Considérons deux cas.

Cas 1. Les étiquettes des agents ont la même taille.

Dans ce cas, les étiquettes transformées ont également la même taille et donc les phases de la procédure `PerdreContact` correspondantes à une valeur donnée d débutent au même moment pour les deux agents. Puisque les étiquettes sont différentes, il existe au moins une position dans les étiquettes transformées dans laquelle les bits sont différents. Nous montrons que les agents doivent se trouver à une distance d'au moins ρ au plus tard à la fin de l'exécution de la boucle pour le plus petit d qui dépassent 2ρ.
En effet, si cette situation ne s'est pas produite auparavant, les agents se trouvent donc à une distance plus petite que ρ dans la direction verticale, au début de leur exécution de la boucle. Soit i le plus petit index des étiquettes transformées à partir duquel ces dernières diffèrent. Avant de traiter ce bit dans cette exécution de la boucle, les agents sont toujours à une distance plus petite que ρ dans la direction verticale. Lors du traitement de ce bit, un des agents demeure immobile et l'autre va se déplacer vers le Nord à une distance plus grande que 2ρ, et par conséquent la distance qui sépare les agents est supérieure à ρ.

Cas 2. Les étiquettes des agents sont de tailles différentes.

Soient x la longueur de l'étiquette transformée la plus petite et $x+a$ celle de l'étiquette transformée la plus grande. Appelons l'agent avec l'étiquette de longueur x l'agent le plus petit et celui avec l'étiquette de longueur $x+a$ l'agent le plus grand. Par définition de la transformation, nous avons $a \geq 6$. Soit i le plus petit entier tel que $4^i > 2\rho$. Pour tout entier positif j, soit S_j (respectivement T_j) le moment où la phase de l'agent le plus petit (respectivement le plus grand), correspondante à la distance $d = 4^j$, se termine. Nous avons donc $S_j = \frac{x}{3}(4^{j+1} - 1)$ et $T_j = \frac{x+a}{3}(4^{j+1} - 1)$. Puisque $a \geq 6$ donc $T_i \geq S_i + 6 \cdot 4^i$. Considérons maintenant le segment du temps final de longueur $6 \cdot 4^i$ de la phase de l'agent le plus grand qui correspond à la distance $d = 4^i$. Durant ce segment de temps, l'agent le plus grand traite les 6 derniers bits de son étiquette transformée et

l'agent le plus court est déjà dans une phase qui correspond à la distance $d = 4^j$, pour un certain $j > i$. Ceci implique que le temps de traitement de chaque bit de l'agent le plus petit est d'au moins 4^{i+1}. Au moins 3 des 6 segments de temps consécutifs de longueur 4^i consacrés par l'agent le plus grand pour le traitement des 6 derniers bits de sa phase qui correspond à la distance $d = 4^i$ doivent être inclus dans un segment de temps I de longueur $d = 4^j$, pour un certain $j > i$, et durant lequel l'agent le plus petit traite un seul bit. Puisqu'il n'y a pas trois 0 consécutifs ou trois 1 consécutifs dans l'étiquette transformée d'un agent, il s'en suit qu'il existe un segment de temps de longueur 4^i, inclus dans le segment de temps I, pendant lequel l'agent le plus grand traite un bit différent de celui traité par l'agent le plus petit durant le segment I. D'où la conclusion qu'il existe un segment de temps de longueur $4^i > 2\rho$ durant lequel un agent est immobile et l'autre de déplace vers le Nord. Durant ce segment de temps, les agents se trouvent à une distance d'au moins ρ ce qui conclut la preuve de l'affirmation. Par ailleurs, grâce à la lecture de *vérifier*(C), chaque agent réalise qu'il s'est rendu à une distance d'au moins ρ de l'autre, ce qui provoque la fin de l'exécution de la procédure `PerdreContact` par l'agent ; pour garantir ceci, cette procédure oblige l'agent à faire deux choses : rester immobile durant un segment de temps de taille 1/2 après chaque lecture de son capteur lorsqu'il traite le bit 0 de son étiquette transformée et rester immobile durant un segment de temps de taille 1 si c'est lui l'agent qui va se déplacer vers le Sud une fois le contact est perdu. Une situation délicate à prévoir est celle dans laquelle l'agent en attente reste immobile pour une longue période sans lire son capteur et durant laquelle l'autre agent en mouvement perd le contact, réalise qu'il l'a perdu et commence immédiatement à se déplacer vers le Sud pour regagner le contact avant que l'agent en attente se rende compte que le contact était déjà perdu.

Dans le but de montrer ceci, soit I le segment de temps dans lequel un des agents traite son bit b' et soit $J \subseteq I$ le segment de temps dans lequel l'autre agent traite son bit b'' tel

que $b' \neq b''$ et le contact est perdu durant le segment de temps J. Si $b' = 1$, l'agent en attente qui traite son bit 0 durant le segment J va réaliser la perte du contact au plus tard à la fin de J et demeurera immobile pour toujours puisque sa variable *principal* aura toujours la valeur $faux$. De son côté, l'agent en mouvement réalisera la perte du contact à la fin du segment I et continuera l'exécution du reste des instructions de l'algorithme. Par contre si $b' = 0$ alors la situation est plus subtile. Encore une fois on suppose que la perte du contact se produit durant le segment de temps J. L'agent en attente renifle après chaque période de temps de longueur 1/2. L'agent en mouvement réalise la perte du contact à la fin du segment de temps J et reste immobile sans rien faire pour une durée de temps de longueur 1. Ce temps d'attente va permettre à l'agent qui traite le bit 0 de réaliser que le contact a été perdu et par conséquent de terminer l'exécution de la procédure `PerdreContact` avant que l'autre agent commence son déplacement vers le Sud et avant qu'il regagne éventuellement le contact.

À la fin de l'exécution de la procédure `PerdreContact`, la symétrie entre les agents sera effectivement brisée. Appelons maintenant l'agent qui va se déplacer vers le Sud après avoir détecter la perte du contact l'agent *principal*. C'est l'agent dont la variable *principal* aura la valeur *vrai* en sortant de la boucle « **Tant que** $C = 1$ »(Notons que dans le cas d'un départ arbitraire, l'agent *principal* est bien sûr le dernier agent). Cet agent sait qu'il est devenu *principal* et que le reste de l'algorithme va être seulement exécuté par lui.

Soit p la ligne verticale suivant laquelle l'agent *principal* se déplace durant l'exécution de la procédure `PerdreContact` et soit X le centre de l'autre agent. Notons par A le point dans p qui se trouve au Nord à la distance ρ de X et par B le point dans p qui se trouve au Sud à la distance ρ de X. À la fin de son exécution de la procédure `PerdreContact`, l'agent *principal* se trouve sur la ligne p à une distance d'au plus 1/2 du point A et à la fin de l'exécution de la boucle **Tant que** de la procédure `TriangleRecherche`, il se

trouve sur la ligne p à une distance d'au plus $1/2$ du point B. Soit C le point que l'agent atteint après avoir parcouru la distance $\lceil t/2 \rceil \cdot \frac{1}{2}$ en direction du Nord sur la ligne p. Le point C va se trouver donc à une distance d'au plus 1 du centre Z du segment AB. Durant l'exécution de la boucle **Répéter** de la procédure `TriangleRecherche`, l'agent se déplace sur la ligne horizontale qui passe par le point C en faisant des bonds croissants à l'Est et à l'Ouest. Par conséquent, il doit éventuellement toucher l'autre agent qui se trouve sur le point X.

Il reste maintenant à estimer le coût de l'algorithme `RencontreAvecCapteurBinaire` qui est représenté par la distance parcourue par les deux agents. Considérons un agent dont l'étiquette transformée a une longueur m. Durant le tour de la boucle « **Tant que** $C = 1$ » de la procédure `PerdreContact`, l'agent parcours la distance $O(dm)$ pour chaque valeur du paramètre d. Puisque le paramètre est multiplié par 4 à chaque tour et il appartient à $\Theta(\rho)$ au dernier tour, il s'en suit que la distance parcourue après une exécution complète de cette boucle est $O(\rho m)$. Le reste des instructions de la procédure `PerdreContact` sont les déplacements vers le sud et ceci coûte $O(\rho)$. Les déplacements sur la ligne verticale p dans la procédure `TriangleRecherche` coûtent aussi $O(\rho)$. D'autre part, les déplacements horizontaux, qui se doublent à chaque instant, effectués à la fin de cette procédure doivent couvrir une distance d'au plus ρ, donc ces mouvements coûtent aussi $O(\rho)$. Par conséquent, la totalité de la distance parcourue par un agent coûte $O(\rho m)$. Puisque que m appartient à $O(\log \lambda)$, le coût de l'algorithme est dans $O(\rho \log \lambda)$.

□

Nous allons maintenant présenter notre dernier résultat qui montre que la complexité de l'algorithme `RencontreAvecCapteurBinaire` ne peut pas être améliorée en général dans le modèle binaire.

Théorème 5.5.2 *Il n'y a pas d'algorithme déterministe pour le modèle binaire qui garantit la rencontre de n'importe quels deux agents qui se trouvent à une distance initiale inférieure à ρ dans le plan et qui fonctionnent au coût $o(\rho \log \lambda)$, où λ est la plus grande étiquette.*

Preuve : Considérons un algorithme déterministe $\mathcal{A}$ qui permet de réaliser la rencontre entre deux agents dans le modèle binaire. Supposons que $\rho > 30$. Nous définissons le carrelage du plan en carreaux disjoints de taille $\rho/5$. Chaque carreau comprend ses arêtes Nord et Est. Pour un carreau fixe 0, nous énumérons ses 8 carreaux voisins par les nombres $1, \ldots, 8$ en commençant par le carreau qui se trouve au Nord de 0 et en suivant le sens des aiguilles de la montre (voir figure 5.5).

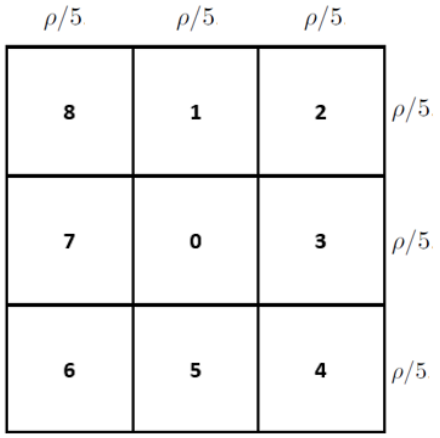

FIGURE 5.5 – Modèle de carrelage du plan

Soient A et B deux carreaux qui se trouvent sur la même ligne horizontale séparés par un carreau. Lorsque A et B se trouvent dans cette configuration on dit qu'ils sont *conjugués*. Plaçons un agent à n'importe quel point de A, l'autre agent à n'importe quel point de B et supposons qu'ils commencent l'exécution de l'algorithme $\mathcal{A}$ simultanément à l'instant 0.

Propriété. Si deux carreaux a et b qui portent le numéro 0 sont conjugués et si les carreaux a' et b' ont le même nombre $i \in \{0, 1, \ldots, 8\}$ par rapport à a et b respectivement alors les carreaux a' et b' sont aussi conjugués (voir figure 5.6).

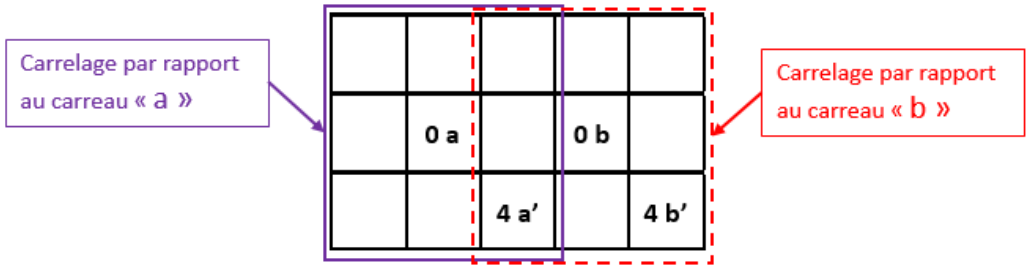

FIGURE 5.6 – a et b sont conjugués et a' associé à a et b' associé à b sont aussi conjugués parce qu'ils se trouvent dans des carreaux qui portent le même numéro 4.

Considérons l'entier positif x et l'ensemble S des entiers positifs dont les représentations binaires ont une longueur de taille x. La taille de l'ensemble S est 2^{x-1} (le premier chiffre d'une représentation binaire doit être 1).

Supposons que l'algorithme $\mathcal{A}$ fonctionne en temps d'au plus $c\rho x$, où $c < \frac{1}{24 \log 9}$ est une constante. Divisons le temps en segments consécutifs de longueur $\rho/6$. Si l'agent se trouve dans un carreau au début d'un segment de temps alors à la fin de ce segment il va se trouver soit dans ce même carreau ou dans un des carreaux voisins. Le *modèle de comportement* d'un agent est une séquence $(a_1, \ldots, a_k)$ de termes $0, 1, \ldots, 8$ et qui est défini comme suit. Si l'agent se trouve dans un carreau 0 au début du i-ème segment alors $a_i = j$ si et seulement si l'agent se trouve dans le carreau j à la fin de ce segment. Puisque l'algorithme fonctionne en temps d'au plus $c\rho x$ alors la longueur de son *modèle de comportement* est d'au plus $k = \lceil 6cx \rceil$. D'autre part puisque $c < \frac{1}{24 \log 9}$ alors nous avons $k \leq \lceil \frac{x}{4 \log 9} \rceil < \frac{x-1}{\log 9}$, pour un x suffisamment grand et par conséquent $9^k < 2^{x-1} = |S|$.

Tant que les agents se trouverons toujours à une distance inférieure à ρ, ce qui implique

que la lecture de leur capteur sera toujours égale à 1, les actions de chaque agent dépendent seulement de son étiquette. Ceci nous pousse à énoncer l'affirmation suivante.

Affirmation. Pour tout $i \leq k$, il existe au moins $\frac{|S|}{9^i}$ entiers dans l'ensemble S tels que si deux de ces entiers sont assignés en tant qu'étiquettes à des agents alors ces derniers se trouvent dans des carreaux conjugués à la fin du i-ème segment de temps et ils sont à une distance d'au moins ρ durant ce i-ème segment.

La preuve de cette affirmation se fait par induction sur i. Pour le cas de base, considérons que $i = 1$. Les agents commencent l'exécution de $\mathcal{A}$ dans des carreaux conjugués. Par conséquent ils commencent à une distance d'au plus $\frac{\sqrt{10}\rho}{5}$. Puisque les segments de temps sont de longueur $\rho/6$ alors la distance entre les agents au cours du premier segment de temps est toujours d'au plus $\frac{\sqrt{10}\rho}{5} + \frac{2\rho}{6} < \rho$. Donc les actions de chaque agent durant le premier segment de temps dépendent uniquement de son étiquette. Soit S_j, pour $j \in \{0, 1, \ldots, 8\}$, l'ensemble des étiquettes de S tel que si un agent reçoit une étiquette de cet ensemble et commence dans un carreau qui a le numéro 0 alors il terminera le premier segment de temps dans le carreau numéro j.Il existe un nombre $j \in \{0, 1, \ldots, 8\}$ pour lequel l'ensemble S_j a une taille d'au moins $\frac{|S|}{9}$. Si les agents reçoivent des étiquettes de l'ensemble S_j alors ils terminent le premier segment dans des carreaux conjugués et ceci conclut la preuve du cas de base.

Supposons maintenant que l'affirmation est vraie pour tout $i' \leq i$, pour un entier $i < k$, et prouvons qu'elle est vraie pour $i + 1$.

Soit T un sous-ensemble de S de taille $\frac{|S|}{9^i}$ tel que si deux entiers de T sont assignés comme étiquettes aux agents alors ces derniers se trouvent dans des carreaux conjugués à la fin de chaque segment de temps $i' \leq i$ et ils sont à une distance inférieure à ρ durant tous ces segments de temps. L'existence de l'ensemble T découle de l'hypothèse d'induction.

Supposons que les agents reçoivent deux entiers de l'ensemble T en tant qu'étiquettes.

Donc ils commencent le $(i+1)$-ième segment de temps dans des carreaux conjugués et par conséquent ils commencent à une distance d'au plus $\frac{\sqrt{10}\rho}{5}$. Puisque les segments de temps sont de longueur $\rho/6$ alors la distance entre les agents duant le $(i+1)$-ième segment de temps est toujours d'au plus $\frac{\sqrt{10}\rho}{5} + \frac{2\rho}{6} < \rho$. Donc les actions de chaque agent durant ce segment de temps dépendent uniquement de son étiquette. Soit T_j, pour $j \in \{0, 1, \ldots, 8\}$, l'ensemble des étiquettes de T tel que si un agent reçoit une étiquette de cet ensemble et commence le $(i+1)$-ième segment de temps dans un carreau qui a le numéro 0 alors il terminera le $(i+1)$-ième segment de temps dans le carreau numéro j. Il existe un nombre $j \in \{0, 1, \ldots, 8\}$ pour lequel l'ensemble T_j a une taille d'au moins $\frac{|T|}{9} = \frac{|S|}{9^{i+1}}$. Si les agents reçoivent leurs étiquettes de l'ensemble T_j alors ils terminent le $(i+1)$-ième segment de temps dans des carreaux conjugués et ceci conclut la preuve par induction de l'affirmation.

Puisque $9^k < |S|$, l'affirmation implique qu'il y a au moins deux étiquettes dans l'ensemble S telles que si deux agents possèdent ces deux étiquettes et commencent dans des carreaux A et B alors ils auront le même modèle de comportement et leurs actions dépendent uniquement de leurs étiquettes. Assignons ces étiquettes aux agents qui commencent dans les carreaux A et B. Il s'ensuit que ces agents se trouvent dans des carreaux conjugués à la fin de chaque segment de temps. Puisque les segments de temps sont de longueur $\rho/6$, nous montrons que les agents sont toujours à une distance d'au moins $\frac{\rho}{5} - \frac{\rho}{6}$. En effet, supposons que durant un segment de temps les agents se trouvent une distance plus petite que $\frac{\rho}{5} - \frac{\rho}{6}$. Ceci peut se produire uniquement lorsque les deux agents se trouvent dans le carreau qui les séparent au début du segment. Par contre, durant ce segment de temps, chaque agent peut entrer dans ce carreau qui les séparent seulement à une distance d'au plus $\rho/12$ parce qu'à la fin de leurs segments de temps, les agents doivent se trouver de nouveau séparés par ce carreau. Puisque les agents étaient à une distance d'au moins $\rho/5$ au début du segment de temps alors cette distance ne

pouvait pas diminuer de plus que $\rho/6$ durant ce segment de temps, ce qui nous donne une distance d'au moins $\frac{\rho}{5} - \frac{\rho}{6} = \frac{\rho}{30} > 1$ à tous les moments. Par conséquent les agents ne peuvent pas se rencontrer.

Cette contradiction implique que le temps de l'algorithme doit être plus grand que $c\rho x$, pour ρ et x suffisamment grands. Puisque x apparient à $\Theta(\log \lambda)$, ceci conclut la preuve.

□

Nous concluons cette section avec l'analyse de la situation dans laquelle les agents se trouvent à une distance initiale de taille $\alpha\rho$ pour une constante $\alpha > 1$ dans le modèle binaire. Nous allons montrer que dans ce cas, le reniflement n'aide pas, c'est à dire que l'ordre du coût optimal de la rencontre en pire cas est le même que celui pour réaliser une rencontre sans que les agents possèdent la possibilité du reniflement. En effet, considérons deux agents qui se trouvent à une distance initiale $D = \alpha\rho$, pour une constante $\alpha > 1$ et soit $\epsilon = \frac{\alpha-1}{2}$. Dans le but d'accomplir la rencontre, les agents doivent en premier lieu se tenir à une distance $\rho(1+\epsilon)$(Il est suffisant de considérer ρ assez grand pour que $\rho(1+\epsilon) > 1$). Cette tâche partielle doit être accomplie dans le modèle sans aucune information supplémentaire puisque la lecture des capteurs des deux agents sera toujours 0 dans ce cas. La tâche partielle est alors équivalente à la tâche de l'approche définie par Dieudonné et Pelc dans leur article [43] quand les agents commencent à une distance initiale $D' = 1 + \epsilon\rho$. Par conséquent, la tâche originale ne peut pas être complétée à un coût inférieur à $Opt(D')$ qui représente le coût optimal de l'approche sans capteurs, lorsque les agents commencent à une distance initiale D'. L'ordre exact de ce coût optimal de l'approche est inconnu mais il est polynomial en D' et en $\log \lambda$, où λ est l'étiquette la plus grande (Les auteurs de [43] ont montré que ce résultat est valable même si les agents sont autorisés à se déplacer d'une manière asynchrone avec quelques restrictions). Puisque $D \in \Theta(D')$ alors $Opt(D) \in \Theta(Opt(D'))$, et par conséquence le coût optimal de la rencontre dans le pire cas dans le modèle binaire avec reniflement,

lorsque les agents se trouve à la distance initiale $D = \alpha\rho$ telle que $\alpha > 1$, est du même ordre de grandeur que s'ils commençaient à la même distance initiale et n'avaient pas de capteurs.

5.6 Conclusion

Nous avons fourni des algorithmes optimaux qui permettent, à deux agents équipés de capteurs de reniflement, de réaliser leur rencontre dans deux modèles qui dépendent de la précision fournie par ces capteurs. Dans le modèle monotone, on suppose que les capteurs sont parfaitement précis, c'est à dire qu'ils peuvent détecter tout changement de distance entre les agents sans pouvoir ni la mesurer, ni connaître la direction dans laquelle se trouve l'autre agent. Dans le cas du modèle binaire qui est plus faible par rapport au monotone, le capteur peut seulement indiquer si l'autre agent est proche ou loin, pour un certain seuil de proximité. Nous avons montré une séparation entre les deux modèles : tandis que dans le modèle monotone la rencontre est garantie à un coût proportionnel à la distance initiale entre les agents, nous avons montré que le coût optimal dans le modèle binaire est $\Theta(\rho \log \lambda)$, où ρ est le seuil sur la distance de détection et λ est l'étiquette la plus grande. Dans les deux modèles nous supposons que les deux agents se déplacent à la même vitesse 1 et cette hypothèse est fortement utilisée dans nos algorithmes et dans leur analyse.

Chapitre 6

Conclusion

Dans cette thèse, nous nous sommes intéressés à étudier l'impact de la communication sur la réalisation du rendez-vous et sur l'amélioration du temps de son accomplissement. La motivation qui nous a poussé à étudier ce problème est que dans le rendez-vous classique, les agents n'ont pas la capacité de communiquer. Nous avons présenté des algorithmes du rendez-vous dans des modèles qui permettent aux agents différents moyens de communication.

Dans le chapitre 3, nous avons fourni des solutions qui permettent de résoudre le problème du rendez-vous *avec détection* en utilisant des *bips*. Malgré la grande faiblesse de ce modèle de communication, nous avons réussi à développer un algorithme de rendez-vous déterministe avec détection qui fonctionne en un temps du même ordre que celui du rendez-vous classique. Pour répondre aux exigences des agents qui possèdent une énergie limitée, nous avons présenté un autre algorithme qui permet de résoudre le même problème en temps polynomial en la taille de l'étiquette la plus grande des deux agents dans n'importe quel graphe de taille bornée. Nous avons ensuite comparé nos deux algorithmes et nous avons remarqué que la différence entre leurs temps d'exécution dans les graphes de taille au plus n est exponentielle. Cette observation nous a aidé à prouver

une borne inférieure exponentielle sur le temps du rendez-vous dans cette situation. Pour contourner cette différence exponentielle, nous avons fait une petite modification dans la manière que les agents utilisent pour écouter les bips. Plus précisément, pour nos deux premiers algorithmes, les agents n'entendent un bip dans un certain nœud u que s'ils s'y trouvent les deux ensemble à la même ronde. Pour notre troisième algorithme, nous avons ajouté à l'agent la capacité d'entendre aussi les bips qui sont émis en dehors du nœud sur lequel il est positionné. Nous avons prouvé que cet algorithme est optimal et qu'il fonctionne en temps logarithmique en la taille de la plus petite étiquette des deux agents. Cette étude s'avère la première de son genre pour résoudre le problème du rendez-vous en utilisant les bips. Un autre problème qui serait intéressant à étudier est celui de comment les bips peuvent être utilisés dans le problème du rendez-vous asynchrone.

Dans le chapitre 4, nous avons fourni un algorithme qui permet de faire le rendez-vous déterministe synchrone dans un arbre entre deux agents, en même ordre du temps que celui de l'exploration. Notre objectif était de trouver la taille minimale des tableaux blancs pour réaliser ce compromis. Nous avons prouvé que notre algorithme fonctionne en temps linéaire en n avec des tableaux blancs de taille $\Theta((\log L)/n)$, où n est la taille de l'arbre et L est une borne supérieure sur l'ensemble des étiquettes. Nous avons montré que cette grandeur de tableaux est optimale pour le rendez-vous en temps $O(n)$ dans les arbres. Pour les graphes arbitraires de taille m, il s'ensuit de [36] que le temps du rendez-vous peut être $O(n^2)$ si la taille des tableaux blancs appartient à $O(\log n + (\log L)/n)$, mais l'optimalité de cette taille reste toujours un problème ouvert.

Finalement dans le chapitre 5, nous avons traité le problème du rendez-vous dans le plan en utilisant des agents qui ont la capacité de renifler. Les agents sont modélisés par des disques de diamètre 1 et la rencontre est réalisée lorsque les disques se touchent, c'est à dire lorsque la distance qui sépare leurs centres est égale à 1. Nous avons montré

comment le reniflement peut aider les agents à se rapprocher de plus en plus en se basant sur l'intensité de l'odeur. Nous avons fourni deux algorithmes optimaux qui permettent à ce genre d'agents de réaliser la rencontre déterministe dans le plan : le premier algorithme fonctionne en temps linéaire en la distance qui sépare les positions initiales des deux agents ; cet algorithme a été réalisé dans le modèle monotone qui permet aux agents, grâce à l'odorat, d'estimer la croissance ou la décroissance de la distance entre eux. Le deuxième algorithme, quant à lui, a été conçu dans un modèle très faible par rapport au modèle monotone, appelé le modèle binaire et qui ne permet aux agents de se sentir que lorsqu'ils se trouvent dans un rayon d'au plus un nombre réel positif ρ, inconnu par les agents ; cet algorithme fonctionne en temps $\Theta(\rho \log \lambda)$, où λ est la plus grande étiquette des deux agents. Nous avons aussi montré que le reniflement ne serait d'aucune utilité si les positions initiales des agents se trouvent à une distance $\alpha\rho$, pour une certaine constante $\alpha > 1$. Un problème qui serait très intéressant à étudier est l'impact du reniflement dans le modèle de navigation asynchrone des agents dans le plan.

Bibliographie

[1] Afek, Y., Alon, N., Bar-Joseph, Z., Cornejo, A., Haeupler, B., and Kuhn, F. Beeping a maximal independent set. *Distributed Computing 26*, 4 (2013), 195–208.

[2] Agathangelou, C., Georgiou, C., and Mavronicolas, M. A distributed algorithm for gathering many fat mobile robots in the plane. In *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013* (2013), pp. 250–259.

[3] Alexanderson, G., Pólya, G., and Boas, R. *The Random Walks of George Polya*. MAA spectrum. Mathematical Association of America, 2000.

[4] Alpern, S. The rendezvous search problem. *SIAM Journal on Control and Optimization 33*, 3 (1995), 673–683.

[5] Alpern, S., and Gal, S. *The Theory of Search Games and Rendezvous*. International Series in Operations Research & Management Science. Springer US, 2006.

[6] Balamohan, B., Flocchini, P., Miri, A., and Santoro, N. Improving the optimal bounds for black hole search in rings. In *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO 2011, Gdansk, Poland, June 26-29, 2011. Proceedings* (2011), pp. 198–209.

[7] Bar-Eli, E., Berman, P., Fiat, A., and Yan, P. Online navigation in a room. *J. Algorithms 17*, 3 (Nov. 1994), 319–341.

[8] Barrière, L., Flocchini, P., Fraigniaud, P., and Santoro, N. Election and rendezvous in fully anonymous systems with sense of direction. In *SIROCCO 10 : Proceedings of the 10th Internaltional Colloquium on Structural Information Complexity, June 18-20, 2003, Umeå Sweden* (2003), pp. 17–32.

[9] Barrière, L., Flocchini, P., Fraigniaud, P., and Santoro, N. Rendezvous and election of mobile agents : Impact of sense of direction. *Theory Comput. Syst. 40*, 2 (2007), 143–162.

[10] Baston, V., and Gal, S. Rendezvous search when marks are left at the starting points. *Naval Research Logistics (NRL) 48*, 8 (2001), 722–731.

[11] Bouchard, S., Bournat, M., Dieudonné, Y., Dubois, S., and Petit, F. Asynchronous approach in the plane : A deterministic polynomial algorithm. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria* (2017), pp. 8 :1–8 :16.

[12] Bouchard, S., Dieudonné, Y., and Ducourthial, B. Byzantine gathering in networks. *Distributed Computing 29*, 6 (2016), 435–457.

[13] Bouchard, S., Dieudonné, Y., and Lamani, A. Byzantine gathering in polynomial time. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic* (2018), pp. 147 :1–147 :15.

[14] Bshouty, N. H., Higham, L., and Warpechowska-Gruca, J. Meeting times of random walks on graphs. *Inf. Process. Lett. 69*, 5 (1999), 259–265.

[15] Caissy, D., and Pelc, A. Exploration of faulty hamiltonian graphs. *Int. J. Found. Comput. Sci. 27*, 7 (2016), 809–828.

[16] Chalopin, J., Das, S., and Kosowski, A. Constructing a map of an anonymous graph : Applications of universal sequences. In *Principles of Distributed Systems - 14th International Conference, OPODIS 2010, Tozeur, Tunisia, December 14-17, 2010. Proceedings* (2010), pp. 119–134.

[17] Chalopin, J., Das, S., and Santoro, N. Rendezvous of mobile agents in unknown graphs with faulty links. In *Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, Proceedings* (2007), pp. 108–122.

[18] Chalopin, J., Dieudonné, Y., Labourel, A., and Pelc, A. Fault-tolerant rendezvous in networks. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II* (2014), pp. 411–422.

[19] Chalopin, J., Dieudonné, Y., Labourel, A., and Pelc, A. Rendezvous in networks in spite of delay faults. *Distributed Computing 29*, 3 (2016), 187–205.

[20] Chatterjee, A., Chaudhuri, S. G., and Mukhopadhyaya, K. Gathering asynchronous swarm robots under nonuniform limited visibility. In *Distributed Computing and Internet Technology - 11th International Conference, ICDCIT 2015, Bhubaneswar, India, February 5-8, 2015. Proceedings* (2015), pp. 174–180.

[21] Chevaleyre, Y. Theoretical analysis of the multi-agent patrolling problem. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 20-24 September 2004, Beijing, China* (2004), pp. 302–308.

[22] Cieliebak, M., Flocchini, P., Prencipe, G., and Santoro, N. Distributed computing by mobile robots : Gathering. *SIAM Journal on Computing 41*, 4 (2012), 829–879.

[23] Collins, A., Czyzowicz, J., Gasieniec, L., Kosowski, A., Kranakis, E., Krizanc, D., Martin, R., and Ponce, O. M. Optimal patrolling of fragmented boundaries. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013* (2013), pp. 241–250.

[24] Collins, A., Czyzowicz, J., Gasieniec, L., Kosowski, A., and Martin, R. A. Synchronous rendezvous for location-aware agents. In *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings* (2011), pp. 447–459.

[25] Cornejo, A., and Kuhn, F. Deploying wireless networks with beeps. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings* (2010), pp. 148–162.

[26] Czyzowicz, J., Dobrev, S., Kranakis, E., and Krizanc, D. The power of tokens : Rendezvous and symmetry detection for two mobile agents in a ring. In *SOFSEM 2008 : Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings* (2008), pp. 234–246.

[27] Czyzowicz, J., Gasieniec, L., Kosowski, A., and Kranakis, E. Boundary patrolling by mobile agents with distinct maximal speeds. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings* (2011), pp. 701–712.

[28] Czyzowicz, J., Gasieniec, L., and Pelc, A. Gathering few fat mobile robots in the plane. *Theor. Comput. Sci. 410*, 6-7 (2009), 481–499.

[29] Czyzowicz, J., Ilcinkas, D., Labourel, A., and Pelc, A. Worst-case optimal exploration of terrains with obstacles. *Information and Computation 225*, Supplement C (2013), 16 – 28.

[30] Czyzowicz, J., Kosowski, A., Kranakis, E., and Taleb, N. Patrolling trees with mobile robots. In *Foundations and Practice of Security - 9th International Symposium, FPS 2016, Québec City, QC, Canada, October 24-25, 2016, Revised Selected Papers* (2016), pp. 331–344.

[31] Czyzowicz, J., Kosowski, A., and Pelc, A. How to meet when you forget : log-space rendezvous in arbitrary graphs. *Distributed Computing 25*, 2 (2012), 165–178.

[32] Czyzowicz, J., Kosowski, A., and Pelc, A. Time versus space trade-offs for rendezvous in trees. *Distributed Computing 27*, 2 (2014), 95–109.

[33] Czyzowicz, J., Kranakis, E., Pajak, D., and Taleb, N. Patrolling by robots equipped with visibility. In *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings* (2014), pp. 224–234.

[34] Czyzowicz, J., Pelc, A., and Labourel, A. How to meet asynchronously (almost) everywhere. *ACM Trans. Algorithms 8*, 4 (2012), 37 :1–37 :14.

[35] Das, S., Dereniowski, D., Kosowski, A., and Uznanski, P. Rendezvous of distance-aware mobile agents in unknown graphs. In *Structural Information and Communication Complexity - 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings* (2014), pp. 295–310.

[36] Das, S., Flocchini, P., Nayak, A., and Santoro, N. Effective elections for anonymous mobile agents. In *Algorithms and Computation, 17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006, Proceedings* (2006), pp. 732–743.

[37] Das, S., Flocchini, P., Prencipe, G., Santoro, N., and Yamashita, M. Autonomous mobile robots with lights. *Theor. Comput. Sci. 609* (2016), 171–184.

[38] Das, S., Focardi, R., Luccio, F. L., Markou, E., Moro, D., and Squarcina, M. Gathering of robots in a ring with mobile faults. In *Proceedings of the 17th Italian Conference on Theoretical Computer Science, Lecce, Italy, September 7-9, 2016.* (2016), pp. 122–135.

[39] Das, S., Luccio, F. L., and Markou, E. Mobile agents rendezvous in spite of a malicious agent. In *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers* (2015), pp. 211–224.

[40] Deng, X., Kameda, T., and Papadimitriou, C. How to learn an unknown environment. i : The rectilinear case. *J. ACM 45*, 2 (Mar. 1998), 215–245.

[41] Dessmark, A., Fraigniaud, P., Kowalski, D. R., and Pelc, A. Deterministic rendezvous in graphs. *Algorithmica 46*, 1 (2006), 69–96.

[42] Dessmark, A., and Pelc, A. Optimal graph exploration without good maps. *Theor. Comput. Sci. 326*, 1-3 (Oct. 2004), 343–362.

[43] Dieudonné, Y., and Pelc, A. Deterministic polynomial approach in the plane. *Distributed Computing 28*, 2 (2015), 111–129.

[44] Dieudonné, Y., and Pelc, A. Anonymous meeting in networks. *Algorithmica 74*, 2 (2016), 908–946.

[45] Dieudonné, Y., Pelc, A., and Peleg, D. Gathering despite mischief. *ACM Trans. Algorithms 11*, 1 (2014), 1 :1–1 :28.

[46] Dieudonné, Y., Pelc, A., and Villain, V. How to meet asynchronously at polynomial cost. *SIAM J. Comput. 44*, 3 (2015), 844–867.

[47] Dobrev, S., Flocchini, P., Kralovic, R., Prencipe, G., Ruzicka, P., and Santoro, N. Black hole search by mobile agents in hypercubes and related

networks. In *Procedings of the 6th International Conference on Principles of Distributed Systems. OPODIS 2002, Reims, France, December 11-13, 2002* (2002), pp. 169–180.

[48] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. Multiple agents rendezvous in a ring in spite of a black hole. In *Principles of Distributed Systems, 7th International Conference, OPODIS 2003 La Martinique, French West Indies, December 10-13, 2003 Revised Selected Papers* (2003), pp. 34–46.

[49] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. Searching for a black hole in arbitrary networks : optimal mobile agents protocols. *Distributed Computing 19*, 1 (2006), 1–99999.

[50] Dobrev, S., Flocchini, P., Prencipe, G., and Santoro, N. Mobile search for a black hole in an anonymous ring. *Algorithmica 48*, 1 (2007), 67–90.

[51] Doyle, P. G., and Snell, J. L. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.

[52] Dumitrescu, A., Ghosh, A., and Tóth, C. D. On fence patrolling by mobile agents. *Electr. J. Comb. 21*, 3 (2014), P3.4.

[53] Elouasbi, S., and Pelc, A. Time of anonymous rendezvous in trees : Determinism vs. randomization. In *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland, June 30-July 2, 2012, Revised Selected Papers* (2012), pp. 291–302.

[54] Elouasbi, S., and Pelc, A. Deterministic rendezvous with detection using beeps. In *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers* (2015), pp. 85–97.

[55] Elouasbi, S., and Pelc, A. Deterministic meeting of sniffing agents in the plane. In *Structural Information and Communication Complexity - 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers* (2016), pp. 212–227.

[56] Elouasbi, S., and Pelc, A. Deterministic rendezvous with detection using beeps. *Int. J. Found. Comput. Sci. 28*, 1 (2017), 77.

[57] Elouasbi, S., and Pelc, A. Deterministic meeting of sniffing agents in the plane. *Fundam. Inform. 160*, 3 (2018), 281–301.

[58] Flocchini, P., Kranakis, E., Krizanc, D., Luccio, F. L., Santoro, N., and Sawchuk, C. Mobile agents rendezvous when tokens fail. In *Structural Information and Communication Complexity, 11th International Colloquium , SIROCCO 2004, Smolenice Castle, Slowakia, June 21-23, 2004, Proceedings* (2004), pp. 161–172.

[59] Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., and Sawchuk, C. Multiple mobile agent rendezvous in a ring. In *LATIN 2004 : Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings* (2004), pp. 599–608.

[60] Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci. 337*, 1-3 (2005), 147–168.

[61] Fraigniaud, P., and Pelc, A. Delays induce an exponential memory gap for rendezvous in trees. *ACM Trans. Algorithms 9*, 2 (2013), 17 :1–17 :24.

[62] Gal, S. *Search Games*. Mathematics in science and engineering : a series of monographs and textbooks. Academic Press, 1980.

[63] Gasieniec, L., Kranakis, E., Krizanc, D., and Zhang, X. Optimal memory rendezvous of anonymous mobile agents in a unidirectional ring. In *SOFSEM 2006 : Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science, Merín, Czech Republic, January 21-27, 2006, Proceedings* (2006), pp. 282–292.

[64] Gilbert, S., and Newport, C. C. The computational power of beeps. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings* (2015), pp. 31–46.

[65] Gorain, B., and Pelc, A. Deterministic graph exploration with advice. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland* (2017), pp. 132 :1–132 :14.

[66] Guilbault, S., and Pelc, A. Asynchronous rendezvous of anonymous agents in arbitrary graphs. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings* (2011), pp. 421–434.

[67] Holzer, S., and Lynch, N. A. Beeping a maximal independent set fast. *CoRR abs/1704.07133* (2017).

[68] Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., and Yamashita, M. The gathering problem for two oblivious robots with unreliable compasses. *SIAM J. Comput. 41*, 1 (2012), 26–46.

[69] Kawamura, A., and Kobayashi, Y. Fence patrolling by mobile agents with distinct speeds. *Distributed Computing 28*, 2 (2015), 147–154.

[70] Koucký, M. Universal traversal sequences with backtracking. *J. Comput. Syst. Sci. 65*, 4 (Dec. 2002), 717–726.

[71] Kowalski, D. R., and Malinowski, A. How to meet in anonymous network. *Theor. Comput. Sci. 399*, 1-2 (2008), 141–156.

[72] Kowalski, D. R., and Pelc, A. Polynomial deterministic rendezvous in arbitrary graphs. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings* (2004), pp. 644–656.

[73] Kranakis, E., and Krizanc, D. An algorithmic theory of mobile agents. In *Trustworthy Global Computing, Second Symposium, TGC 2006, Lucca, Italy, November 7-9, 2006, Revised Selected Papers* (2006), pp. 86–97.

[74] Kranakis, E., Krizanc, D., and Markou, E. *The Mobile Agent Rendezvous Problem in the Ring*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2010.

[75] Kranakis, E., Krizanc, D., and Markou, E. Deterministic symmetric rendezvous with tokens in a synchronous torus. *Discrete Applied Mathematics 159*, 9 (2011), 896–923.

[76] Kranakis, E., Krizanc, D., and Morin, P. Randomized rendezvous with limited memory. *ACM Trans. Algorithms 7*, 3 (2011), 34 :1–34 :12.

[77] Kranakis, E., Krizanc, D., and Rajsbaum, S. Mobile agent rendezvous : A survey. In *Structural Information and Communication Complexity, 13th International Colloquium, SIROCCO 2006, Chester, UK, July 2-5, 2006, Proceedings* (2006), pp. 1–9.

[78] Kranakis, E., Santoro, N., Sawchuk, C., and Krizanc, D. Mobile agent rendezvous in a ring. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA* (2003), pp. 592–599.

[79] Langetepe, E. On the optimality of spiral search. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010* (2010), pp. 1–12.

[80] Langetepe, E. Searching for an axis-parallel shoreline. *Theor. Comput. Sci. 447* (2012), 85–99.

[81] Machado, A., Ramalho, G., Zucker, J., and Drogoul, A. Multi-agent patrolling : An empirical analysis of alternative architectures. In *Multi-Agent-Based Simulation, Third International Workshop, MABS 2002, Bologna, Italy, July 15-16, 2002, Revised Papers* (2002), pp. 155–170.

[82] Marco, G. D., Gargano, L., Kranakis, E., Krizanc, D., Pelc, A., and Vaccaro, U. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci. 355*, 3 (2006), 315–326.

[83] Miller, A., and Pelc, A. Fast rendezvous with advice. *Theor. Comput. Sci. 608* (2015), 190–198.

[84] Miller, A., and Pelc, A. Tradeoffs between cost and information for rendezvous and treasure hunt. *J. Parallel Distrib. Comput. 83* (2015), 159–167.

[85] Miller, A., and Pelc, A. Time versus cost tradeoffs for deterministic rendezvous in networks. *Distributed Computing 29*, 1 (2016), 51–64.

[86] Ooshita, F., Kawai, S., Kakugawa, H., and Masuzawa, T. Randomized gathering of mobile agents in anonymous unidirectional ring networks. *IEEE Trans. Parallel Distrib. Syst. 25*, 5 (2014), 1289–1296.

[87] Pearson, K. The problem of the random walk. *Nature 72*, 294 (1905).

[88] Pelc, A. Deterministic rendezvous in networks : A comprehensive survey. *Networks 59*, 3 (2012), 331–347.

[89] Peleg, D. Distributed coordination algorithms for mobile robot swarms : New directions and challenges. In *Distributed Computing – IWDC 2005* (Berlin, Heidelberg, 2005), A. Pal, A. D. Kshemkalyani, R. Kumar, and A. Gupta, Eds., Springer Berlin Heidelberg, pp. 1–12.

[90] Rao, N. S. V., Kareti, S., Shi, W., and Iyengar, S. S. Robot navigation in unknown terrains : Introductory survey of non-heuristic algorithms, 1993.

[91] Reingold, O. Undirected connectivity in log-space. *J. ACM 55*, 4 (Sept. 2008), 17 :1–17 :24.

[92] Schelling, T. *The strategy of conflict*. Harvard University Press, 1981.

[93] Stachowiak, G. Asynchronous deterministic rendezvous on the line. In *SOFSEM 2009 : Theory and Practice of Computer Science, 35th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 24-30, 2009. Proceedings* (2009), pp. 497–508.

[94] Ta-Shma, A., and Zwick, U. Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences. *ACM Trans. Algorithms 10*, 3 (2014), 12 :1–12 :15.

[95] Tsuchida, M., Ooshita, F., and Inoue, M. Byzantine gathering in networks with authenticated whiteboards. In *WALCOM : Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29-31, 2017, Proceedings.* (2017), pp. 106–118.

[96] Yamashita, M., and Kameda, T. Computing on anonymous networks : Part i-characterizing the solvable cases. *IEEE Trans. Parallel Distrib. Syst. 7*, 1 (1996), 69–89.

[97] Yu, J., Jia, L., Yu, D., Li, G., and Cheng, X. Minimum connected dominating set construction in wireless networks under the beeping model. In *2015*

IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015 (2015), pp. 972–980.

[98] Yu, X., and Yung, M. Agent rendezvous : A dynamic symmetry-breaking problem. In *Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings* (1996), pp. 610–621.