Université du Québec en Outaouais

Rendez-vous asynchrone des agents mobiles anonymes dans les réseaux

par

Samuel Guilbault

Thèse présentée au
Département d'informatique et d'ingénierie
Pour l'obtention du grade de

Philosophiae Doctor (Ph.D.)

en Sciences et technologies de l'information

Membres du jury :

Président	Karim El Guemhioui, Ph.D., U.Q.O.
Examinateur interne	Jurek Czyzowicz, Ph.D., U.Q.O.
Examinateurs externes	Michel Barbeau, Ph.D., Carleton University Paola Flocchini, Ph.D., University of Ottawa
Directeur de recherche	Andrzej Pelc, Ph.D., U.Q.O.

Janvier 2014

Résumé

Dans le cadre de cette thèse, nous proposons un sujet concernant le traitement d'information par les agents mobiles. Plus spécialement, nous allons approfondir le problème de rendez-vous des agents dans les réseaux. Nous considérons une collection d'agents qui peuvent être des entités anonymes ou identifiables, avec une visibilité restreinte ou complète, possédant de la mémoire ou étant amnésiques : c'est-à-dire qu'ils n'ont aucun souvenir des événements passés. Dans la littérature, on considère les agents qui évoluent dans des réseaux synchrones ou asynchrones. Un algorithme adapté pour un système asynchrone fonctionnera également dans un système synchrone. Le but du rendez-vous est de regrouper tous les agents en même temps au même endroit.

Nous sommes particulièrement intéressés par des problèmes du rendez-vous des agents anonymes évoluant dans les réseaux asynchrones. Nos résultats concernent la faisabilité du rendez-vous des agents qui ne peuvent pas communiquer entre eux : soit ils n'ont aucune possibilité de connaître la localisation des autres agents, soit ils peuvent détecter leur présence seulement à un rayon restreint dans le réseau. Cette capacité limitée d'observation ajoutée au mode asynchrone de déplacement des agents rend le rendez-vous particulièrement difficile. Par contre, la faiblesse des hypothèses utilisées augmente l'applicabilité de nos modèles.

Abstract

In this thesis, we propose a topic concerning information processing by mobile agents. More specifically, we will study the problem of gathering agents in networks. We consider a collection of agents which can be anonymous or labelled entities, with limited or full vision, equipped with memory or oblivious; i.e., with no memory of past events. In the literature, agents operating in synchronous or asynchronous networks have been considered. An algorithm suitable for an asynchronous system will also work in a synchronous system. The purpose of the gathering is to bring together all the agents at the same time in the same place.

We are particularly interested in problems of gathering anonymous agents operating in asynchronous networks. Our results concern the feasibility of gathering of agents that cannot communicate with each other : either they have no possibility of knowing the location of other agents, or they can detect their presence only at a restricted radius in the network. This limited capacity added to the asynchronous mode of movements of agents makes the gathering particularly difficult. However, the weak assumptions that we use increase the applicability of our models.

Remerciements

Premièrement, je voudrais remercier mon directeur de thèse, le Dr. Andrzej Pelc, qui a dû plusieurs fois s'arracher quelques cheveux à cause de mes nombreux défauts. Il est un modèle pour moi et je n'ai aucune hésitation à lui dire qu'il est un homme d'exception.

Également. je voudrais remercier ma mère qui a corrigé mes travaux et parfois avec des délais plutôt courts. Les mamans sont rarement capables de dire non à leurs enfants mais dans le cas de la mienne, je veux qu'elle sache que j'en ai conscience et que je n'oublierai jamais l'énergie qu'elle a consacrée à la réussite de mes études.

Je voudrais offrir des remerciements spéciaux aux personnes suivantes : le Dr. Karim El Guemhioui, le Dr. Michel Barbeau et le Dr. Paola Flocchini qui ont aimablement accepté de réviser ma thèse et d'être membres de mon jury. J'aimerais offrir des remerciements particuliers au Dr. Jurek Czyzowicz qui a non seulement été membre de mon jury de thèse mais également membre de mon jury lors de ma maîtrise, sans compter les activités supplémentaires auxquelles il a participé, telles que l'examen synthèse de mes études de doctorat.

Sans oublier de remercier certains membres du personnel de l'Hôpital du Sacré-Coeur de Montréal qui ont été présents pour m'aider à vivre avec la maladie dont je suis atteint : le Dr. Daniel Bichet, les infirmières Carole Fortier et Claudia Ménard.

J'aimerais finalement offrir des remerciements à ma famille. À ma conjointe Jacinthe qui a passé du temps à m'aider dans mes travaux mais plus spécialement à mes enfants qui ont dû partager leur père entre ses obligations de carrière, d'études et de santé. J'espère être un aussi bon père et conjoint que vous êtes pour moi de bon enfants et une bonne conjointe.

Dédicace

Je dédie cette thèse aux personnes qui m'ont aidé à passer à travers quelques épreuves durant la période de mon doctorat.

Premièrement à mes enfants. Mon fils Malcolm qui a dû, plus que mes autres enfants, me partager entre mes obligations professionnelles et académiques. Mon fils Noah qui est tout jeune et qui réclame son père lorsque celui-ci travaille sous son nez. À ma petite princesse des fées Mélisande qui a été diagnostiquée d'autisme il y a un peu plus d'un an. Je t'aime très fort et, sans le savoir, tu m'as rappelé à chaque moment difficile que même avec un handicap, la vie peut être un éternel émerveillement.

En second, à mon directeur de recherche, mon mentor, mais surtout mon ami. J'ai progressé beaucoup à son contact et malgré qu'il me reste beaucoup de chemin à faire, je pense être une bien meilleure personne que je ne l'étais, et ce, grâce à lui.

En troisième, à ma conjointe qui a pris une place très importante dans mon coeur...

Finalement, à mon amie A.V. Martin qui est décédée des suites d'une tumeur au cerveau. J'ai tenu la promesse que je t'avais faite. Repose en paix.

Table des matières

Table des figures

Chapitre 1

Introduction

Depuis le début du 20^e siècle, les technologies ont pris de plus en plus part à notre mode de vie. On s'intéresse à toutes sortes de moyens pour réduire la nécessité que l'être humain participe à des tâches qui sont routinières, dangereuses ou tout simplement difficiles à accomplir. C'est pour ces principales raisons que la radio ainsi que la télévision (communication), les automobiles, les avions, les fusées (exploration), les outils de gestion et de rencontres (rendez-vous) ont vu le jour. Toutefois, plus l'être humain accroît sa dépendance face aux machines afin d'alléger son fardeau, plus il a de nouvelles idées pour améliorer davantage sa condition. Ainsi, bien que les appareils électroménagers existent, l'être humain doit les faire fonctionner. Bien qu'il existe des équipement protégeant l'homme d'un feu, il doit quand même le combattre. Bien qu'il existe des réseaux dans la plupart des entreprises, l'homme doit en assurer le maintien en effectuant une kyrielle de tâches diverses. Il est logique de se poser la question suivante : «Serait-il possible de réduire la nécessité de la présence de l'homme pour accomplir des tâches qu'il ne veut pas ou ne peut pas accomplir ?». Parmi les solutions envisageables, l'une d'elles est certainement d'avoir un substitut à l'homme. Toutefois, afin d'être capable de mimer certaines actions de l'homme, une machine, voire un robot (agent mobile) devrait être extrêmement évolué. Certains chercheurs ont réalisé que, bien qu'un robot sophistiqué puisse accomplir des tâches complexes, un groupe de plusieurs robots plus simples peut généralement accomplir les mêmes tâches à plus bas prix. Notons que nous utilisons le terme d'agents mobiles pour parler de robots puisque ce terme englobe aussi bien les robots mobiles que les agents logiciels distribués considérés «intelligents»

qui se déplacent dans les réseaux informatiques alors que le terme «robots» fait plutôt allusion à une classe plus restreinte d'agents mobiles : les entités mécaniques dotées de facultés se rapprochant de celles de l'être humain.

La technologie des agents mobiles est un paradigme qui a connu un essor considérable dans les dernières années et dans lequel chaque agent fonctionne comme une entité autonome. Les agents peuvent agir de manière indépendante d'un moniteur central et en son absence. Dans le cas d'un environnement réseau, ils peuvent suspendre l'exécution d'une tâche sur un noeud du réseau pour la reprendre sur un autre noeud adjacent. Dans un environnement distribué, les agents vont accomplir des tâches en essaim afin d'optimiser l'efficacité avec laquelle la tâche est exécutée, augmenter les chances que la tâche soit accomplie avec succès et accroître la sécurité du système.

Parmi les tâches principales que des agents peuvent accomplir, il y a celle de garantir la sécurité d'un système (ou d'un environnement) ainsi que la tâche d'explorer un environnement. Dans les deux cas, les agents vont éventuellement se disperser. Récupérer ces agents dans un même endroit sera ensuite nécessaire. À titre d'exemple, les agents doivent souvent partager des informations acquises et alors, il est nécessaire qu'ils se regroupent. Il en résulte que le problème de rendez-vous est sous-jacent à celui de l'exploration ainsi que de la maintenance de l'environnement par plusieurs agents mobiles et, par conséquent, il revêt une importance capitale dans la littérature. Nous n'avons qu'à penser au problème de la galerie d'art (problème de géométrie algorithmique). En effet, à l'aide de caméras fixes, nous pouvons garantir la sécurité d'une galerie d'art. Imaginons maintenant que nous voulons économiser sur le nombre de caméras ou encore, nous voulons être capables d'être efficaces pour une galerie dont la disposition nous est inconnue à priori. L'exploration ainsi que la gestion de la sécurité par des robots mobiles est faisable. Bien sûr, plus il y aura de robots plus l'exploration et la sécurité seront efficaces, mais nous sommes alors confrontés au compromis entre l'efficacité de l'accomplissement de la tâche et le coût d'acquisition des nombreux agents.

Nous allons nous pencher sur le problème de rendez-vous des agents mobiles. Nous allons premièrement décrire plus en profondeur ce que sont les agents mobiles ainsi que les intérêts d'approcher un problème avec ce paradigme. Nous allons ensuite détailler la problématique selon l'environnement dans lequel les agents évoluent et finalement, nous allons suggérer un axe

de recherche afin de contribuer à l'avancement de la connaissance dans ce domaine.

1.1 Les agents mobiles

Dans la recherche en robotique, tant du point de vue de l'ingénierie que du point de vue fonctionnel, la tendance semble s'éloigner de plus en plus de la conception et du déploiement d'un faible essaim d'agents de nature complexe et couteuse [12, 13, 33, 47, 50]. L'intérêt semble plutôt s'accroître sur la conception et l'utilisation de grands essaims d'agents génériques simples qui ont des capacités limitées et, par conséquent, sont très peu dispendieux.

Un agent simple est généralement capable de capturer l'état de son environnement (observation), faire des calculs afin de prendre une décision à savoir s'il restera sur place ou se déplacera (calcul), et dans le dernier cas, se déplacera vers le lieu calculé (déplacement). Ces trois phases étant exécutées en un cycle sans fin (ou jusqu'à ce que l'agent décide de l'arrêter). Malgré leurs capacités individuelles limitées, les agents unis en un large essaim devraient être en mesure d'accomplir des tâches complexes. Parmi les tâches que l'essaim peut accomplir, on peut nommer le rendez-vous, l'élection d'un chef, la formation de structures, etc. Une question importante, lors du choix des agents, consiste à déterminer le type d'agents nécessaires ; autrement dit, quelle est leur capacité minimale requise pour accomplir la tâche.

Ces questions ont été étudiées de façon élaborée tant du point de vue expérimental que du point de vue théorique dans le scénario de visibilité complète. Autrement dit, l'hypothèse est que les agents sont capables d'observer l'espace entier dans lequel ils évoluent. En général et de façon plus réaliste, les robots peuvent uniquement voir une partie de leur environnement. Cela peut être dû à des limitations telles que des murs, la mécanique de l'agent et l'environnement possiblement hostile. Nous parlons alors de visibilité restreinte et, de façon générale, les problèmes ainsi que les algorithmes se complexifient.

Considérons un système d'agents mobiles autonomes. Chaque agent a sa propre perception de l'environnement. Cette vue est définie comme une partie

du plan cartésien (dans le plan) ou encore un sous-graphe (dans les réseaux). Souvent, lorsqu'ils évoluent dans le plan, les agents partagent le même système de coordonnées. Par contre, cela ne veut pas dire qu'ils sont d'accord sur l'origine, généralement assumée comme étant la position de l'agent lui-même (ou le noeud qu'il occupe dans un réseau).

Les agents peuvent être sans mémoire ou peuvent posséder différentes capacités qui leur procurent une forme de mémoire (par exemple des jetons, de la mémoire intégrée, etc.). Nous supposons que l'agent est sans mémoire lorsqu'il ne possède aucune capacité lui permettant de se souvenir d'événements passés. Les agents peuvent être également anonymes (identiques) ou étiquettés (différentiables). Les agents peuvent parfois communiquer entre eux mais fréquemment, la seule forme de communication (indirecte) possible est l'observation de leur environnement. Autrement dit, les agents perçoivent le changement de l'environnement dû au déplacement des autres agents. En somme, nous pouvons présumer que la forme la plus simple d'agents est un agent sans mémoire, anonyme, avec une perception restreinte. Chaque agent exécute un algorithme identique qui reçoit en entrée la position des autres agents dans le rayon de perception et retourne une destination vers laquelle l'agent se déplacera. Les solutions à des problèmes de rendez-vous qui utilisent ce type simple d'agents sont très complexes.

L'état initial dans lequel l'agent se trouve est un état d'attente. Dans certains systèmes appelés synchrones, tous les agents bougent en rondes mesurées par une horloge globale. Toutefois, dans les systèmes plus réalistes dits asynchrones, les agents exécutent leur cycle indépendamment et pas nécessairement à la même vitesse ou en même temps. Ce cycle consiste à observer, calculer et se déplacer si nécessaire. Notons qu'un algorithme fonctionnant dans un système asynchrone fonctionnera également dans les systèmes synchrones.

1.2 Définition du problème

Étant donné la croissance permanente des réseaux (tel qu'Internet) dans notre société moderne, il est de plus en plus utile de trouver des approches qui ne nécessitent pas un gestionnaire centralisé pour résoudre les problèmes

concernant ces réseaux. Imaginons la possibilité de pouvoir alléger la tâche de travail d'un utilisateur à l'aide d'agents mobiles, par exemple, pour trouver des noeuds défaillants dans ce réseau ou encore pour construire la carte de ce réseau s'il n'est pas connu à priori. Un défi de base pour beaucoup de problèmes de maintenance et d'exploration des réseaux par plusieurs agents mobiles est la rencontre de ces agents. Nous allons donc nous concentrer sur ce défi particulier : le rendez-vous d'agents mobiles. Le rendez-vous en lui-même est un problème avec beaucoup d'applications. Par exemple, pensons à un groupe de gens qui chercheraient à se rencontrer dans le noir total. De même, un essaim d'agents peut être aveugle à son environnement dans une certaine mesure. Ou encore, considérons deux personnes qui voudraient se rencontrer dans un endroit inconnu, ces personnes n'ayant pas de boussole et bien sûr, n'avançant pas du tout au même rythme. Deux agents pourraient désirer faire de même afin de partager un objet ou encore une connaissance. Les environnements asynchrones sont les plus répandus en pratique et les tâches distribuées dans ces scénarios posent le plus grand défi. Par conséquent, nous allons nous intéresser principalement au problème de rendez-vous dans un environnement asynchrone.

1.3 Modèles

Nous allons tout d'abord considérer le résultat, qui a fait l'objet de la publication [35]. Ce résultat est obtenu dans un modèle qui est une modification du modèle CORDA (modèle bien connu dans la littérature) adopté aux réseaux. Dans ce modèle, on suppose en général que les robots ont une perception complète du réseau. En dehors de cette capacité très puissante, les agents sous le modèle CORDA ont des capacités très limitées. Ils n'ont aucune mémoire et sont tous identiques. De plus, nous considérons des algorithmes fonctionnant dans des systèmes asynchrones. Dans le modèle que nous proposons, en plus d'avoir toutes les limitations du modèle CORDA, nous avons réduit la perception qu'un agent a de son environnement : il ne perçoit que le voisinage immédiat du noeud sur lequel il se trouve. Nous avons suggéré cette modification au modèle CORDA pour réduire le contraste entre la faiblesse des agents quant à leur manque de mémoire et leur puissance au niveau de la perception. Dans notre modèle les agents sont faibles dans tous les aspects de leur comportement.

Nous considérons ensuite le résultat qui a fait l'objet de la publication [37]. Ce résultat est obtenu dans un modèle qui est lui aussi une modification du modèle CORDA. Dans le modèle que nous proposons, en plus d'avoir toutes les limitations du modèle CORDA, nous avons réduit la perception qu'un agent a de son environnement : il ne perçoit que la partie du réseau à distance d du noeud sur lequel il se trouve. Dans [35], nous avons montré qu'il est très difficile de regrouper des agents lorsque ceux-ci ont uniquement une vision de leur voisinage immédiat. Dans [37] nous avons donc suggéré cette généralisation du modèle de [35] pour augmenter un peu la perception des agents tout en conservant une réduction du contraste entre la faiblesse des agents quant à leur manque de mémoire et leur puissance au niveau de la perception. Il s'avère que l'augmentation du rayon de vision des agents de $d = 1$ à des valeurs d plus grandes change beaucoup la possibilité de rendez-vous.

Nous considérons finalement le résultat qui a fait l'objet de la publication [36]. Le résultat concerne le rendez-vous de deux agents avec une visibilité nulle, c'est-à-dire que l'agent ne voit l'autre agent que lorsqu'il le croise auquel cas le rendez-vous est accompli. Toutefois, les deux agents peuvent se rencontrer sur une arête du graphe (en plus de se rencontrer sur un noeud).

Nous utilisons un modèle dans lequel l'agent choisit sa trajectoire, mais sa vitesse peut être arbitraire et variable, ce qui modélise le mode asynchrone du déplacement de l'agent.

Dans plusieurs raisonnements, aussi bien dans la revue de la littérature que dans nos résultats, nous utilisons le concept d'adversaire. Ce concept sert à modéliser le pire cas, pour un algorithme donné, permis par le scénario considéré. Par exemple, l'adversaire peut modéliser la vitesse arbitraire et variable de chaque agent afin de rendre l'exécution d'un algorithme de rendez-vous le plus difficile possible dans un modèle asynchrone.

1.3.1 Présentation des résultats

Dans le chapitre 3 dont les résultats ont été publiés dans [35], nous considérons le problème de rendez-vous des agents mobiles identiques, sans mémoire, qui doivent se rencontrer sur un seul noeud d'un graphe connexe

biparti régulier anonyme non-orienté. Rappelons qu'un graphe biparti régulier est un graphe sans cycles de longueur impair et dont tous les noeuds ont le même degré. Les agents commencent sur des positions différentes dans le graphe. Chacun des agents fonctionne en cycle observation-calcul-déplacement et tous les agents doivent se rencontrer en un seul noeud du graphe. Dans un cycle, un agent observe son environnement immédiat (observation), prend une décision de demeurer où il est ou de se déplacer (calcul), et dans le dernier cas, bouge vers le voisin déterminé de manière instantanée (l'agent n'est pas visible sur une arête lorsqu'il se déplace). Chaque agent fonctionne en cycles asynchrones. Comme mentionné précédemment, la nouveauté du modèle utilisé réside dans la capacité de perception de l'agent qui est restreinte (par rapport au modèle CORDA) au voisinage immédiat de l'agent. Nous disons d'une configuration initiale qu'elle est regroupable si, et seulement si, il existe un algorithme qui permet le rendez-vous de tous les agents de la configuration en un seul noeud du graphe et les immobilise à partir de ce moment-là, peu importe les actions qu'un adversaire asynchrone prendra (l'algorithme peut même être dédié à accomplir le rendez-vous pour cette configuration particulière). Le problème de rendez-vous est de déterminer quelles configurations permettent le rendez-vous et de trouver un algorithme (universel) qui accomplit le rendez-vous pour toutes les configurations regroupables. Nous donnons une solution complète du problème de rendez-vous pour les graphes réguliers bipartis. Notre principale contribution est la preuve que les configurations regroupables sont très rares : elles consistent en des étoiles (un agent A avec tous les autres agents lui étant adjacents) de grandeur au moins 3. D'un point de vue positif, nous donnons un algorithme qui accomplit le rendez-vous lorsque c'est faisable.

Dans le chapitre 4, dont les résultats font partie de la publication [37], nous considérons le problème de regroupement d'agents mobiles identiques, sans mémoire, sur un noeud d'une ligne anonyme infinie. Le modèle diffère de celui du chapitre 3 uniquement sur ce point : les agents ne peuvent voir qu'à une distance bornée d (appelé *rayon de vision*) à partir de leur position actuelle. Le modèle est donc plus général que celui de [35] où on supposait $d = 1$. Un algorithme de regroupement est *universel* s'il regroupe toutes les configurations regroupables. Nous observons d'abord que, si la vision des agents est infinie, alors il existe un algorithme universel de regroupement. Pour le rayon de vision $d = 1$ un algorithme de regroupement universel est présenté dans le chapitre 3 [35]. En revanche, le résultat principal du chapitre

5 [37] est que, pour un rayon de vision $d > 1$ il n'existe pas d'algorithme de regroupement universel. Notre résultat reste valable pour les anneaux de taille d'au moins $7d + 8$.

Dans le chapitre 5 dont les résultats ont été publiés dans [36], nous considérons le rendez-vous de deux agents mobiles dans un graphe connexe arbitraire, possiblement infini. Les agents sont modélisés comme des points, ils commencent dans deux noeuds arbitraires du graphe (choisis par l'adversaire) et la route de chacun de ces agents dépend uniquement de la section du graphe qui a déjà été traversée et, dans le cas du rendez-vous aléatoire, des résultats du tirage au sort à l'aide d'une pièce de monnaie. La marche des agents dépend aussi de l'adversaire asynchrone qui peut varier arbitrairement la vitesse de l'agent, l'immobiliser ou le promener aller-retour, en autant que la marche d'un agent sur un segment de sa route soit continue, ne quitte pas le segment et le couvre en entier. Le rendez-vous veut dire que les deux agents doivent être localisés sur un même noeud ou sur un même point à l'intérieur d'une arête exactement au même moment. Dans le scénario déterministe, nous caractérisons les positions initiales pour lesquelles le rendez-vous est possible et nous donnons un algorithme qui garantit le rendez-vous pour toutes ces positions. Dans le scénario aléatoire, nous montrons un algorithme qui accomplit le rendez-vous avec probabilité 1, pour n'importe quelles positions initiales dans un graphe connexe arbitraire. Dans les deux cas, le graphe peut être fini ou infini (mais dénombrable).

Chapitre 2

Revue de la littérature

Dans cette section, nous passons en revue les résultats concernant principalement les problèmes de rendez-vous déterministe dans les réseaux. Nous allons graduellement restreindre le champ de notre analyse, en partant du problème d'exploration qui est connexe à celui du rendez-vous, pour ensuite explorer plus en profondeur les problèmes de rendez-vous lorsque les algorithmes étudiés sont probabilistes. Nous allons ensuite naviguer vers la classe de problèmes de rendez-vous dans l'environnement géométrique utilisant des algorithmes déterministes (classe qui est étroitement liée à la classe des problèmes dans les réseaux) pour finalement nous concentrer sur le principal objectif de cet état de l'art, le rendez-vous déterministe dans les réseaux. Cette décision a été prise pour deux raisons. La première raison est la volonté de choisir un champ de problèmes dont les solutions sont méthodologiquement homogènes. La deuxième raison et la principale est que l'ensemble de notre recherche s'est portée essentiellement sur cette classe de problèmes et qu'il nous paraît important d'approfondir la littérature la concernant. Il est important de noter que, si le lecteur veut se faire une opinion plus approfondie sur les autres classes de problèmes dont nous allons faire le survol, certaines excellentes enquêtes ont été publiées, en particulier deux enquêtes sur le problème de rendez-vous utilisant des algorithmes aléatoires. Chronologiquement, la première des deux est [2], presque entièrement contenu dans la deuxième partie du livre [3].

2.1 Exploration

Une ou plusieurs entités mobiles doivent explorer un réseau, en traversant chacun de ses noeuds et liens (et/ou doivent construire un plan du réseau en trouvant le graphe sous-jacent). Le problème est de trouver le moyen le plus efficace afin de compléter la tâche le plus rapidement possible en évitant les traversées redondantes. Cette tâche a de nombreuses applications dans des domaines allant de l'interaction humaine ou animale au comportement de la programmation des agents mobiles autonomes (robots) et des agents logiciels. Un exemple d'un problème d'exploration qui peut être nécessaire à résoudre par les humains est le problème de trouver des intrus dans un territoire surveillé. Afin d'être efficaces et d'éviter de rendre la tâche plus ardue en laissant aux intrus le temps de se cacher, il est important pour les gardiens de minimiser le temps qu'ils prendront pour explorer l'environnement. Une autre application utilisant des robots peut être nécessaire, par exemple pour éteindre un feu dans un environnement hostile pour l'être humain. Chez les animaux, un problème bien connu est le problème de l'exploration effectuée par les fourmis afin d'approvisionner la colonie en vivres, en trouvant les sources de nourriture les plus près et les chemins les plus faciles pour y accéder.

Les problèmes algorithmiques demandant d'accomplir une tâche nécessitant l'exploration se produisent essentiellement dans deux situations. La première situation concerne des robots mobiles autonomes qui doivent explorer un terrain dans le plan. La seconde situation concerne les agents logiciels, c'est-à-dire des entités logicielles mobiles qui circulent dans un réseau de communication pour explorer ce réseau afin, par exemple, d'y trouver des chemins de moindre résistance pour acheminer des messages d'un point vers un autre. Les algorithmes d'exploration sont influencés par les caractéristiques des agents, telles que les capacités de percevoir leur environnement, de mémoire, des éléments tels qu'une horloge globale (capacité de synchronie), et la nature de l'environnement où ils évoluent.

Par exemple, dans [10], l'auteur montre que deux agents qui coopèrent peuvent découvrir n'importe quel graphe orienté fortement connexe avec n noeuds non différenciables en temps polynomial en fonction de n. L'auteur introduit une nouvelle forme de séquence de ralliement pour deux robots, qui les aident à reconnaître certains noeuds préalablement rencontrés. L'auteur

donne également un algorithme qui apprend cette séquence de ralliement ainsi que le graphe simultanément en explorant. De plus, l'auteur présente un algorithme qui explore le graphe à l'aide de la marche aléatoire. Dans ce dernier cas, l'algorithme fonctionne également en temps polynomial en fonction de n.

Dans [27], les auteurs considèrent un type d'exploration plus particulier : l'environnement réseau contient des trous noirs. Un trou noir est un processus stationnaire localisé sur un noeud qui détruit les agents au moment où ceux-ci entrent dans le noeud, sans laisser aucune trace. Dans ce modèle, les auteurs considèrent l'exploration dans l'anneau de grandeur n. Ils considèrent en premier lieu le cas où les agents partent du même noeud. Dans ce cas, les auteurs prouvent que deux agents sont nécessaires et suffisants pour localiser le trou noir et ils donnent un algorithme fonctionnant en nombre de mouvements $O(n \log n)$ montrant également que cette borne est optimale. Les auteurs considèrent aussi le temps et montrent comment atteindre la borne optimale de $2n-4$ unité de temps en utilisant $n-1$ agents. Ils considèrent également un compromis entre le temps et le nombre d'agents. Ils considèrent ensuite le cas ou les agents partent de noeuds différents et montrent que, si l'anneau est orienté, 2 agents sont tout de même nécessaires et suffisants. Encore une fois, leur algorithme dans ce cas est optimal sur le nombre de mouvements, soit $\Theta(n \log n)$. Ils concluent en montrant que si l'anneau n'est pas orienté, alors 3 agents sont nécessaires et suffisants et ils donnent également un algorithme optimal pour ce cas.

Dans [28], les auteurs considèrent une autre limitation au problème d'exploration dans un graphe $G = (V, E)$. Cette fois, les agents ont comme contrainte une limite sur la distance à laquelle ils peuvent s'éloigner de leur point de départ ou encore la distance qu'un agent peut parcourir avant de devoir revenir à son point de départ pour se recharger. Dans ces deux cas, l'efficacité des algorithmes présentés par les auteurs se mesure en fonction du nombre de traversées d'arêtes. Dans les deux cas mentionnés, les auteurs proposent un algorithme qui permet à l'agent d'explorer le graphe en $\Theta(|E|traverses)$. Les auteurs améliorent ainsi la meilleure borne connue de $O(|E| + |V| \log^2 |V|)$ trouvée par [6]. Puisque leur algorithme est optimal à un facteur constant près, les auteurs répondent ainsi à la question posée dans [5].

Dans [49], les auteurs considèrent l'exploration, par un seul agent, des arbres dans lesquels certaines arêtes sont fautives. L'agent doit explorer les composantes fonctionnelles de l'arbre sans connaître les arêtes fautives à priori. Pour un arbre et une position initiale donnée, la *surcharge* d'un algorithme d'exploration est le ratio en pire cas (en prenant compte de toutes les configurations fautives) de son coût par rapport au coût d'un algorithme optimal qui sait où les fautes sont localisées. Notons que le coût est évalué en fonction du nombre de parcours d'arêtes. Les auteurs donnent un algorithme *parfaitement compétitif* pour les lignes (i.e. un algorithme dont la surcharge est minimale par rapport à tout algorithme d'exploration ne connaissant pas la localisation des fautes) et donnent un algorithme d'exploration pour n'importe quel arbre dont la surcharge est en pire cas $\frac{9}{8}$ plus large qu'un algorithme parfaitement compétitif. Les deux algorithmes présentés par les auteurs sont linéaires en fonction de la grandeur de l'arbre.

Dans [30], les auteurs considèrent le problème d'exploration dans les anneaux anonymes non-orientés par un groupe de k agents identiques, sans mémoire, percevant l'environnement mais n'ayant pas la capacité de communiquer et évoluant dans un environnement asynchrone. Les auteurs démontrent que l'exploration de l'anneau est faisable malgré les importantes restrictions du modèle. Il est prouvé que le nombre minimal $p(n)$ de robots qui peuvent explorer un anneau de grandeur n est $O(\log n)$ et que $p(n) = \Omega(\log n)$ pour n suffisamment grand. Les auteurs donnent un algorithme qui explore l'anneau à partir d'une configuration initiale tant que n et k sont co-premiers et montrent qu'il existe toujours un tel k dans $O(\log n)$. Ils montrent toutefois que $\Omega(\log n)$ agents sont nécessaires pour n suffisamment grand. Les auteurs montrent également que lorsque k divise n, le problème est insoluble.

Dans [31], les auteurs considèrent le problème d'exploration dans les arbres dans un modèle similaire à [30]. Ils prouvent qu'en général, l'exploration ne peut pas être faite efficacement. Ils prouvent qu'il existe des arbres qui requièrent $\Omega(n)$ agents ; résultat qui tient même si le degré maximum est 4. Lorsque le degré est 3, les auteurs montrent qu'il est possible d'utiliser seulement $O(\frac{\log n}{\log \log n})$ agents. Ils démontrent aussi que ce résultat est asymptotiquement optimal. Les auteurs montrent que, pour les arbres qui n'ont pas d'automorphisme non trivial, 4 agents sont toujours suffisants et parfois nécessaires.

Finalement, dans [23], les auteurs considèrent l'exploration déterministe d'un anneau non-orienté anonyme contenant des robots asynchrones, amnésiques et myopes. Par myopes, les auteurs veulent dire que les robots n'ont qu'une visibilité limitée. Les auteurs étudient les contraintes de calcul imposées par ces robots et montrent que, sous certaines conditions, le problème de l'exploration peut encore être résolu. Ils étudient les cas où la visibilité des robots est limitée à 1, 2 et 3 noeuds voisins, respectivement.

2.2 Rendez-vous

Deux ou plusieurs entités mobiles, qui débutent sur des positions distinctes, doivent se rencontrer. Cette tâche, appelée rendez-vous, a de nombreuses applications dans des domaines allant de l'interaction humaine au comportement de la programmation des agents mobiles autonomes (robots) et des agents logiciels. Un exemple d'un problème de rendez-vous qui peut être nécessaire à résoudre par les humains est le problème de l'astronaute [2], dans lequel deux astronautes atterrissent à deux endroits éloignés sur un corps sphérique et doivent minimiser le temps qu'ils prendront pour atteindre un lieu où ils seront capables de se voir. Une application similaire plus connue est celle d'un groupe de soldats dispersés qui doivent se regrouper en un point de rendez-vous non déterminé à l'avance, ou encore des sauveteurs qui veulent trouver une personne perdue dans une mine. Schelling [52] a probablement été le premier à étudier les questions qui peuvent être considérées comme un prototype du problème de rendez-vous : deux joueurs doivent se rencontrer dans une ville inconnue en une seule tentative. Il est naturel de se poser la question suivante : est-ce que deux ou plusieurs personnes peuvent supposer qu'un endroit est plus propice qu'un autre, sans qu'ils se soient mis d'accord, afin de se rencontrer (par exemple, la mairie d'une ville ou encore au poste de police). Dans la réalité, il y a des situations où c'est faisable mais pas toujours. Quant aux problèmes algorithmiques concernant le rendez-vous, ces lieux sont généralement inexistants lorsque les agents évoluent dans un plan ou dans un réseau ne possédant pas de caractéristiques qui permettent d'identifier un lieu plutôt qu'un autre (par exemple un graphe régulier).

Les problèmes algorithmiques demandant d'accomplir une tâche nécessitant le rendez-vous se produisent essentiellement dans deux situations. La

première situation concerne des robots mobiles autonomes qui commencent dans des lieux différents d'un terrain (plan) et qui doivent se rencontrer, par exemple, afin d'échanger des informations obtenues, tout en explorant le terrain et de coordonner leurs actions futures. La seconde situation concerne les agents logiciels, c'est-à-dire, des entités logiciels mobiles qui circulent dans un réseau de communication afin d'effectuer la maintenance de ses composantes, le diagnostic du système ou pour recueillir des données distribuées dans les noeuds du réseau. Ces agents logiciels ont également besoin de se rencontrer périodiquement, afin d'échanger des données collectées et de planifier leurs actions futures. Les algorithmes de rendez-vous sont influencés par les caractéristiques des agents, telles que la capacité de percevoir leur environnement, la mémoire, les éléments tels qu'une horloge globale (capacité de synchronie), et la nature de l'environnement où ils évoluent. Lorsque les algorithmes de rendez-vous s'appliquent à plus de deux agents, nous parlons souvent d'algorithmes de regroupement. Toutefois, par souci d'uniformité, nous allons utiliser le terme rendez-vous pour l'un ou l'autre de ces cas.

Puisque les agents qui doivent se rencontrer ne sont pas contrôlés par un moniteur central et doivent prendre les décisions sur leur déplacement de façon autonome, les problèmes de rendez-vous appartiennent naturellement au calcul réparti. Cependant, diverses hypothèses sur les scénarios de rendez-vous donnent lieu à un grand nombre de différents modèles. Parmi les diverses hypothèses qui ont été utilisées pour étudier le problème de rendez-vous, il y en a deux qui ont influencé plus significativement les méthodologies pour résoudre ces problèmes. La première hypothèse concerne la manière dont les agents se déplacent : soit de façon déterministe soit aléatoire. Plus précisément, dans tout scénario déterministe, les positions initiales des agents sont choisies par un adversaire qui modélise la situation dans le pire des cas, et chaque mouvement de l'agent est déterminé uniquement par son histoire qui peut inclure l'identité de l'agent (le cas échéant), et la partie de l'environnement que l'agent a vue à ce jour. En revanche, dans un scénario aléatoire, les positions initiales des agents sont choisie au hasard et leurs mouvements peuvent également impliquer des lancer de monnaie. Du point de vue méthodologique, les problèmes de rendez-vous déterministe nécessitent généralement des méthodes combinatoires, tandis que les problèmes de rendez-vous aléatoire requièrent plutôt des outils analytiques. La deuxième des hypothèses concerne l'environnement dans lequel les agents doivent évoluer : ils peuvent évoluer ou bien sur un terrain (un plan), ou bien dans un réseau mo-

délisé comme un graphe non orienté. Alors que le rendez-vous dans le plan mène principalement à des considérations géométriques, comme [50, 51, 53], le rendez-vous dans un réseau implique essentiellement des méthodes provenant de la théorie des graphes, comme, par exemple [21, 39, 40].

2.2.1 Rendez-vous probabiliste

Lorsque nous considérons les algorithmes qui sont développés pour permettre le rendez-vous des agents, il y a deux types d'algorithmes qu'il faut considérer : les algorithmes de nature déterministe ou probabiliste. Avant de considérer les algorithmes de nature déterministe, étant le principal sujet de cette thèse, nous allons aborder le sujet des algorithmes aléatoires. La publication [2] presque inclusivement comprise dans le livre [3] couvre très bien le problème de rendez-vous aléatoire. [46] a également fait un bon survol du problème de rendez-vous en général et plus spécifiquement le problème de rendez-vous aléatoire.

Dans [16], les auteurs considèrent deux jetons situés sur deux noeuds arbitraires d'un graphe connexe non orienté de grandeur n. Étant donné la nature des jetons qui peuvent se déplacer, nous pouvons considérer que ces jetons sont en fait des agents. Nous considérons qu'un agent exécute une marche aléatoire si l'agent se déplace avec probabilité égale à chacun des noeuds adjacents à sa position actuelle. Les auteurs ont amélioré les bornes préalablement connues qui étaient exponentielles et ont donné une nouvelle borne qui est polynomiale en fonction de n.

Dans [11], les auteurs considèrent un problème semblable au rendez-vous. Dans ce cas-ci, l'agent doit se rendre sur un lieu précis dans le plan en évitant des obstacles rectangulaires sur son parcours. Dans ce modèle, l'agent n'a pas de connaissance a priori de la position ou de la dimension des obstacles et il acquiert cette connaissance lorsqu'il rencontre un obstacle spécifique. L'objectif est de réduire la distance parcourue dans le plan afin que l'agent atteigne son objectif. Les auteurs donnent un algorithme probabiliste possédant un ratio compétitif de $O(n^{\frac{4}{9}} \log n)$, où n est la distance Euclidienne entre l'agent et son objectif, battant ainsi la borne inférieure compétitive de $\Omega(\sqrt{n})$ existante pour les algorithmes déterministes.

Dans [1], les auteurs considèrent k agents dans un graphe de n noeuds exécutant une marche aléatoire. Lorsque deux agents se rencontrent sur un noeud, ils s'unissent pour former un simple agent. Les auteurs présentent une chaîne de Markov modélisant le comportement des agents et montrent comment cela peut être utilisé pour capturer la borne supérieure sur le temps afin que tous les agents s'unissent en un seul agent. Les auteurs donnent quelques topologies de graphes sur lesquels ils ont calculé la borne supérieure. Ils commencent avec la topologie en étoile qui prend un temps constant $O(1)$ pour que deux agents se rencontrent, étendant cela pour k agents donnant le résultat $O(\log k)$. Dans le cas d'un graphe complet, le temps requis pour deux agents est linéaire en n. De même, pour k agents, le calcul donne $O(n \log k)$. Ensuite, les auteurs étudient le cas de l'hypercube, avec deux agents, pour lequel le temps est de $O(n \log \log n)$ tandis que pour k agents, ils obtiennent le temps $O(kn \log \log n)$.

Finalement, dans [45], les auteurs considèrent le compromis entre le temps que prennent deux agents, qui évoluent dans un environnement synchrone, pour se rencontrer dans un anneau orienté anonyme et la mémoire requise pour se rencontrer. Ils montrent qu'il existe des agents avec $2t$ états qui peuvent accomplir le rendez-vous sur un anneau de grandeur n en temps espéré $O(\frac{n^2}{2^t+2^t})$ et que n'importe quels agents avec $\frac{t}{2}$ états requièrent un temps espéré de $\Omega(\frac{n^2}{2^t})$. Les auteurs observent également que $\Theta(\log \log n)$ bits de mémoire sont nécessaires et suffisants pour effectuer le rendez-vous en temps espéré linéaire.

2.2.2 Rendez-vous déterministe dans un environnement géométrique

Un agent est considéré comme une unité mobile (souvent mécanique dans le plan) capable de percevoir la position des autres agents et son environnement, de faire des calculs sur les données recensées, et de bouger vers la destination calculée. Les calculs sont effectués par des algorithmes déterministes qui utilisent la position des autres agents comme principale information afin de déterminer une position vers où se déplacer. Tous les agents exécutent le même algorithme et la vue personnelle de chaque agent comprend une mesure de grandeur, un point d'origine (généralement la position de l'agent lui-même), et un système de coordonnées cartésiennes. En général, le seul

moyen de communication entre les agents est une communication indirecte, c'est-à-dire qu'ils tirent leur conclusion en dessrvant les déplacements des autres agents.

Une notion importante qui doit être connue a priori est la capacité des agents. Il y a trois capacités qui ont une influence considérable sur la nature des algorithmes : la perception des agents, l'accord sur le système de coordonnées et la mémoire des agents.

La perception peut être globale ou limitée. Le modèle CORDA d'origine utilise une capacité de perception globale. La perception inclut également la capacité de détecter s'il y a plus d'un seul robot sur un même point. Nous parlons alors de capacité de détection de multiplicité. Un agent peut être capable de compter le nombre d'agents sur un point, il peut être seulement capable de percevoir qu'il y a un robot ou plus d'un robot, ou encore, il ne peut pas voir qu'il y a plus d'un robot jusqu'à ce qu'il atteigne lui-même ce point.

Les agents ne partagent pas nécessairement le même système de coordonnées, ou encore la même échelle de distance de même qu'ils ne partagent pas nécessairement la même perception du point d'origine. En général, la capacité d'orientation des agents n'est pas axée dans la même direction, c'est-à-dire que le nord peut varier d'un agent à un autre. Nous disons alors que les robots n'ont aucun accord. Nous parlerons d'accord total lorsque les agents s'accordent sur la direction et l'orientation des deux axes et nous parlerons plutôt d'accord partiel lorsqu'ils s'accordent seulement sur un seul axe.

La mémoire est une autre capacité importante. Nous disons qu'un agent n'a pas de mémoire lorsqu'il ne garde aucun historique des observations et/ou calculs qu'il a faits dans la passé. Résoudre un problème de rendez-vous est généralement plus complexe lorsque les robots n'ont pas de mémoire. Certains articles se concentrent à déterminer quelle est la capacité minimale requise afin d'accomplir le rendez-vous dans un scénario donné.

Une autre notion importante qui a une influence majeure sur la conception d'algorithmes ainsi que sur la capacité d'établir la preuve de la faisabilité du rendez-vous est la synchronie. De façon générale, nous parlons d'environnement synchrone lorsque les agents peuvent se référer à une horloge globale

et respecter un horaire. Nous parlons plutôt d'environnement asynchrone lorsque c'est un adversaire asynchrone qui détermine à quel moment un agent accomplit ses actions. Enfin, plus particulièrement dans les algorithmes de rendez-vous dans l'environnement géométrique, un compromis est fait sur la synchronie. Nous parlons alors de modèle semi-synchrone dans lequel les agents fonctionnent de la même manière que dans l'environnement complètement synchrone mais un adversaire peut décider de ne pas «réveiller» un ou plusieurs agents dans un cycle donné, en autant qu'au moins un agent soit fonctionnel, durant chaque cycle.

Nous présentons tout d'abord l'article [12] dans lequel l'auteur donne un algorithme pour regrouper un ensemble de n agents sur un point dans le plan. Dans le modèle utilisé, les agents n'ont pas de système de coordonnées communes, aucun identificateur, et ils ne peuvent pas communiquer. Dans cet article, l'auteur montre que le problème de rendez-vous peut être résolu en ajoutant uniquement la capacité de mémoire illimitée. L'auteur commence par présenter un algorithme pour exactement deux robots. Dans ce cas, l'idée est que les deux agents bougent sur une ligne l qui connecte leur position initiale. Aussitôt que les deux robots ont observé la configuration au moins une fois (ils connaissent l), ils commencent à bouger chacun sur une ligne différente perpendiculaire à l jusqu'à ce qu'ils aient vue la ligne perpendiculaire de l'autre robot. Finalement, ils se rencontrent dans le centre de l déterminé par les deux lignes perpendiculaires. L'auteur donne ensuite un algorithme qui accomplit le rendez-vous lorsqu'il y a plus de deux robots.

Dans [13], les agents sont similaires à ceux dans [12] sauf qu'il n'ont pas de mémoire. Les auteurs donnent un algorithme de rendez-vous pour $n > 4$ agents. En fait, il a été démontré que grâce au point de Weber (ou Fermat ou Torricelli) défini comme le point de minimisation de la somme des distances des points d'échantillonnage [48, 55] qui peut être calculé pour quatre agents et moins [8], le rendez-vous est faisable. Le point de Weber des positions intiales des agents a la propriété qu'il ne change pas lorsque les agents se déplacent dans sa direction. Si le point de Weber peut être calculé, le rendez-vous est accompli en déplaçant tous les agents sur ce point.

Il suffit donc de traiter le cas $n > 4$. Si la configuration initiale est biangulaire (i.e. s'il existe un point c, une séquence de robots, et deux angles α et β, de telle sorte que les angles entre les robots adjacents par rapport à c

sont soit α, soit β, et que les angles alternes), les auteurs s'assurent que tous les agents bougent vers le centre de biangularité [4]. La configuration future demeure biangulaire jusqu'à ce que deux agents ou plus atteignent le centre. Après cet événement, une unique multiplicité est créée et les autres agents s'y dirigent. Pour toutes les autres configurations, les auteurs sélectionnent un sous-ensemble strict d'agents. Cette sélection est faite en utilisant une chaîne d'angles d'agents par rapport au centre du plus petit cercle les englobant. Les auteurs montrent que si un unique agent peut être élu, il va aller sur un autre agent créant une unique multiplicité. Dans le cas contraire, les agents sélectionnés bougent vers le centre du plus petit cercle les englobant, s'assurant que le centre ne change pas en fonction de leur mouvement. Si une configuration biangulaire est évitée durant le mouvement, deux agents ou plus atteignent le centre du cercle et, à nouveau, il y a une unique multiplicité.

Dans [33], les agents sont très similaires à ceux dans [13] à l'exception qu'ils possèdent un compas et qu'ils ont une visibilité restreinte sur leur environnement. Les auteurs présentent un algorithme de rendez-vous dont l'idée est la suivante. Afin de comprendre l'algorithme, définissons l'univers : le plus petit rectangle rectiligne (i.e. rectangle dont les côtés sont parallèles à un des axes des coordonnées) contenant la configuration initiale d'agents. Les agents doivent bouger vers le bas ou la droite de l'univers (un agent ne bougera jamais vers la gauche ou vers le haut) d'une telle façon qu'après un certain nombre d'étapes, ils vont se regrouper au coin en bas et à droite de l'univers.

Finalement, dans [14], les agents sont similaires à ceux dans [12]. Toutefois, [14] s'intéresse au problème du rendez-vous aussi bien dans le modèle asynchrone que dans le modèle semi-synchrone. Dans ce dernier modèle, les auteurs présentent un algorithme où les robots convergent vers le centre de gravité de leur positions initiales (sans jamais l'atteindre), l'algorithme fonctionne également pour deux robots dans le modèle asynchrone et cela, même pour un espace à d dimensions.

2.2.3 Rendez-vous déterministe dans un environnement réseaux

Dans la section 2.2.3.1, nous présentons une taxonomie détaillée relative au problème de rendez-vous déterministe dans les réseaux. Il faut savoir, avant de lire cette section, qu'il y a quelques hypothèses communes en ce qui a trait à la majorité des articles que nous allons présenter. Une hypothèse, fort probablement la plus importante, est celle qui considère que les réseaux sont modélisés comme des graphes simples, non-orientés, connexes dont les noeuds représentent les processeurs et les arêtes représentent les canaux de communication bidirectionnels entre ces processeurs. Notons que certaines situations requièrent que les canaux de communications soient orientés, auquel cas le graphe passe d'un graphe non-orienté à un graphe orienté. L'hypothèse que, dans la plupart des cas, le graphe ne soit pas orienté est justifié par le fait que dans les situations réelles, il est rare qu'un agent puisse aller d'un point vers un autre en passant par un canal de communication mais qu'il ne puisse pas revenir par ce même canal. La justification de l'autre hypothèse, à l'effet que le graphe est connexe, est aisée à comprendre : deux agents localisés dans deux composantes connexes d'un graphe non-connexe ne pourront jamais se rencontrer.

Une autre hypothèse commune dans la plupart des articles de la littérature est que le réseau est anonyme, c'est-à-dire que les noeuds ne sont pas différenciables par les agents à l'aide d'un identificateur et que les agents ne peuvent utiliser que des caractéristiques topologiques pour s'aider, tel que le degré d'un noeud. Cette hypothèse est justifiée par deux éléments. Premièrement, le souci de confidentialité d'un noeud. Il n'est pas toujours prudent de révéler son identificateur à un agent, particulièrement dans les conditions actuelles des réseaux où plusieurs entités étrangères peuvent malicieusement envahir le réseau. L'autre élément qui justifie une telle hypothèse est de nature théorique : en effet, si les agents peuvent distinguer les noeuds, ils peuvent déterminer un noeud où ils se regrouperont, par exemple le noeud avec le plus petit identificateur et alors, le problème de rendez-vous se réduit à un problème d'exploration.

Finalement, dans certains modèles, les ports d'un noeud (les sorties des arêtes) peuvent être identifiés par un numéro de ports. Toutefois, il n'y a pas de relation supposée entre l'étiquetage d'un noeud à l'autre. Dans certains

algorithmes dont les ports ne sont pas identifiés, il arrive qu'un agent ne va pas être en mesure de visiter tous les voisins de ce noeud puisqu'il ne pourra pas identifier quel port il a exploré jusqu'à maintenant. Une technique qui permet de ne pas utiliser de ports sur les noeuds consiste à donner la capacité de perception globale aux agents. C'est-à-dire qu'ils sont capables de percevoir le réseau en entier.

Taxonomie du rendez-vous

Lorsque les algorithmes déterministes de rendez-vous dans les environnements réseaux sont construit, il y a principalement un problème qui rend la tâche difficile afin de trouver un algorithme qui fonctionne pour chaque configuration permettant le rendez-vous ou encore afin de montrer que c'est impossible : c'est le problème de symétrie dans le graphe. Nous entendons par symétrie un automorphisme du graphe qui porte les noeuds vides sur les noeuds vides et les noeuds occupés par des agents sur les noeuds occupés par des agents. Pour faire l'observation que la symétrie peut rendre le rendez-vous impossible, considérons le scénario suivant : prenons un anneau et considérons 2 agents étant symétriquement opposés. L'adversaire peut toujours faire en sorte de bouger ces deux agents simultanément et de conserver ainsi les positions symétriques des agents.

Bien sûr, les chercheurs ont trouvé, au fil des années, quelques manières pour contourner le problème de symétrie en intégrant des capacités dans les modèles utilisés tels que des identificateurs distincts des agents, la possibilité que les agents laissent des jetons sur un noeud (afin de marquer leur passage) lors de leur déplacement, ou encore qu'il existe un tableau sur chaque noeud utilisé par les agents afin de laisser des messages. Dans le premier cas, il est généralement supposé que l'agent connaît son identificateur mais ne connaît pas celui des autres agents. Si tous les identificateurs sont connu par tous les agents, l'algorithme le plus simple est un algorithme du type : «attends maman». Un agent, par exemple celui avec le plus petit identificateur, attend que les autres agents se rendent jusqu'à lui et alors, le problème se réduit à un problème d'exploration. L'autre technique qui concerne les jetons approche le problème de manière différente. Les agents possèdent chacun un (ou plusieurs) jetons identiques qui peuvent être abandonnés sur un noeud du graphe. Les jetons peuvent être de nature fixe ou déplaçable. C'est-à-dire

que, si les jetons sont fixes, une fois qu'un agent les a déposés sur un noeud, aucun agent (incluant le propriétaire) ne peut le récupérer. Par contre, si les jetons sont de nature déplaçable, alors un agent qui croise un jeton sur un noeud peut non seulement le voir, mais également décider de le récupérer afin de le réutiliser ultérieurement. Finalement, l'utilisation d'un tableau sur chaque noeud est une autre méthode très puissante de briser la symétrie. Chaque noeud possède un tableau où rien n'est écrit à priori et sur lequel les agents peuvent laisser des notes. Cette méthode, bien que très puissante, a un niveau élevé de réalisme dans certains modèles où des processeurs interagissent avec leur environnement.

Il arrive toutefois qu'à l'occasion, le modèle utilisé n'offre aucun des avantages mentionnés plus haut mais il demeure que la symétrie cause un problème. Dans ce cas, une supposition sur les positions initiales peut être faite ou encore le graphe sélectionné peut présenter de faibles caractéristiques de symétrie (un graphe très irrégulier par exemple). La supposition sur les positions initiales est telle que l'algorithme commence dans une configuration qui n'est pas symétrique et alors, le défi n'est plus de briser la symétrie mais plutôt de préserver l'absence de symétrie dans la configuration, pendant le fonctionnement de l'algorithme.

Lorsqu'un problème de rendez-vous est étudié, il est important pour le chercheur de déterminer de quelle manière la symétrie sera brisée (ou évitée). Ce choix aura un impact majeur sur la sélection des capacités des agents nécessaires pour le rendez-vous. Par exemple, un agent peut généralement être capable de distinguer le degré d'un noeud sur lequel il est localisé ou de percevoir des numéros de ports sur chacune des arêtes sortantes. L'agent peut être capable de percevoir une partie de son environnement ou encore la totalité de son environnement. Il peut laisser des messages très détaillés sur un tableau ou au contraire, avoir une restriction importante sur la taille des messages. Ceci concerne les capacités sensorielles des agents telles que la perception et la communication mais qu'en est-il des autres capacités ? Il y a une capacité qu'un agent peut avoir et qui est très importante : la mémoire. Un agent peut avoir des capacités mémorielles nulles, restreintes ou illimitées. Résoudre le rendez-vous avec des agents qui ont des capacités très fortes rend la tâche plus simple ; par contre, dans la réalité, le coût en est nettement plus dispendieux. C'est pour ces raisons qu'il est commun de voir différentes combinaisons entre ces capacités dans les différents modèles explorés. En

général, pour un type de problème donné, le chercheur essaiera d'utiliser le minimum de capacité requise afin de résoudre le problème. Il est facile de comprendre pourquoi : équiper les agents des capacités supplémentaires comporte un coût. Qui voudrait payer des coûts exorbitants pour accomplir une tâche qui peut être faite à un moindre coût ?

Une autre hypothèse qui doit être envisagée est la synchronie de l'environnement dans lequel les agents évoluent. Les déplacements des êtres vivants sont asynchrones, il n'y a personne qui bouge constamment à la même vitesse. Les algorithmes qui peuvent fonctionner pour des réseaux asynchrones sont donc plus applicables.

Quelles sont les questions d'intérêt dans le champs de rendez-vous dans les réseaux ? Il y a premièrement le problème de faisabilité. Dans une classe de graphes, est-ce que toutes les situations où le rendez-vous peut être accompli sont couvertes par l'algorithme ? Est-ce qu'il est démontré pour quelles configurations le rendez-vous est faisable et pour lesquelles ça ne l'est pas ? Est-ce qu'il existe une borne inférieure sur la mémoire ou encore sur le nombre de déplacements pour accomplir la tâche et, si oui, est-ce que l'algorithme est optimal ? Ce sont de nombreuses questions qui peuvent dans certains cas être répondues aisément mais qui, d'en d'autre cas, se révèlent être un défi de taille.

Le reste du chapitre se profilera comme suit. Nous allons d'abord présenter les algorithmes qui fonctionnent dans un environnement synchrone. Les algorithmes seront divisés en fonction des capacités que les agents ont. Ensuite, nous allons nous concentrer sur les algorithmes qui fonctionnent dans l'environnement bien plus complexe : l'environnement asynchrone. Encore une fois, nous diviserons les résultats selon les caractéristiques dont les agents font preuve.

Modèle synchrone

Nous commençons l'étude du rendez-vous synchrone en utilisant des résultats qui concernent une topologie de graphes fort intéressante dans la littérature due à ses particularités et à sa facilité d'implantation : les anneaux.

Dans [17], les auteurs considèrent deux variantes au problème de rendez-vous : la variante sans détection où les agents doivent se rencontrer lorsqu'ils débutent dans des positions de l'anneau non symétriques et la variante avec détection lorsque les agents doivent en plus détecter lorsqu'ils ne peuvent accomplir le rendez-vous parce qu'ils ont débuté dans des positions symétriques et alors s'immobiliser. Dans cet article, les auteurs ne se concentrent que sur les jetons amovibles et ils recherchent essentiellement les compromis entre le nombre de jetons disponibles aux agents et le temps requis afin d'accomplir le rendez-vous dans l'anneau de grandeur n. Notons que l'anneau peut être unidirectionnel ou bidirectionnel dans le scénario qu'ils utilisent. Les auteurs considèrent les agents avec une mémoire constante. Ils montrent que si ces agents utilisent un seul jeton, le rendez-vous avec la détection de symétrie n'est pas faisable et le rendez-vous sans détection n'est faisable que si les agents évoluent dans un anneau bidirectionnel, auquel cas le temps optimal de rendez-vous est $\Theta(n^2)$. En augmentant le nombre de jetons de un par agent (les agents ont maintenant deux jetons), ils rendent le rendez-vous avec la détection possible, peu importe si l'anneau est bidirectionnel ou non et alors, le temps optimal de rendez-vous est toujours $\Theta(n^2)$. En augmentant le nombre de jetons d'avantage (trois jetons ou plus) ainsi qu'en ajoutant une capacité de mémoire logarithmique, ils montrent que le rendez-vous avec la détection est faisable dans l'anneau bidirectionnel en temps $O(n^{\frac{(t-1)}{(t-2)}}t)$ où t est le nombre de jetons.

Dans [47], les auteurs s'intéressent toujours à l'anneau dans un scénario similaire au précédent. Cette fois, l'anneau peut être orienté ou non. L'anneau est considéré comme étant orienté lorsque les deux agents s'entendent sur un sens de la direction. Dans ce scénario, les agents débutent leur exécution simultanément à deux endroits à distance d l'un de l'autre. Les agents peuvent se rencontrer sur un noeud ou sur une arête du graphe. Nous considérons que les agents se rencontrent sur un noeud lorsqu'ils occupent ce noeud au même moment alors qu'il se rencontreront sur une arête s'ils la traversent dans des directions opposées (et se croisent). Les auteurs considèrent le temps optimal de rendez-vous en considérant le nombre d'étapes avant le rendez-vous selon les paramètres d, n ainsi que s'ils se sont entendus sur un sens de la direction. Les auteurs ont montré que si $d = \frac{n}{2}$, alors le rendez-vous est impossible. Les auteurs supposent donc que $d < \frac{n}{2}$ et que les agents savent que cette inéquation est respectée. Les auteurs montrent des bornes

inférieures de même que supérieures selon les cas. Si l'anneau est orienté et que la valeur de d est connue des agents et peu importe la connaissance sur n, le temps optimal de rendez-vous est de $\frac{3d}{2}$. Toutefois, si n n'est pas connu et que la direction de l'anneau n'est pas considérée, le temps optimal de rendez-vous passe à $(n+d)/2$. Dans le cas ou ni n ni d ne soient connus et peu importe l'orientation de l'anneau, ce résultat passe à $n+\frac{d}{2}$. Toute ces bornes sont exactes. Toutefois, si l'anneau est non orienté et d est connu, la borne inférieure est de $\frac{3d}{2}$ et la borne supérieure est de $\frac{5d}{2}$. Finalement, dans le scénario complexe tel que ni n, ni d ne sont connus et que l'anneau n'est pas orienté, les auteurs offrent un compromis entre la mémoire et le temps. Ils présentent un algorithme qui fonctionne en temps $n+\frac{d}{2}$ et qui utilise $\Theta(\log n)$ bits de mémoire ou offre le compromis suivant : on peut économiser de la mémoire en passant à $\Theta(\log\log n)$ en payant le prix par un temps de $O(\frac{n\log n}{\log\log n})$. Il est enfin prouvé que n'importe quel algorithme qui accomplit le rendez-vous lorsque $d<\frac{n}{2}$ et s'arrête lorsque $d=\frac{n}{2}$ requiert une mémoire de $\Omega(\log\log n)$ bits.

Dans [43], la propriété des jetons est élargie afin de comparer les scénarios avec des jetons fixes ou amovibles. Dans ce scénario, un jeton fixe ne peut être qu'observé alors qu'un jeton amovible peut être pris par un autre agent et déplacé. Les auteurs considèrent le nombre de bits de mémoire pour deux agents identiques qui sont nécessaires pour résoudre le problème de rendez-vous dans le tore orienté. Comme dans [47], le rendez-vous peut se produire sur un noeud ou sur une arête. Les auteurs ont tout d'abord démontré que le rendez-vous est impossible peu importe le nombre de jetons dans un tore $n\times m$ si le vecteur entre leurs positions initiales est soit $(\frac{n}{2},0)$, ou $(0,\frac{m}{2})$, ou $(\frac{n}{2},\frac{m}{2})$. Les agents résolvent le problème de rendez-vous s'ils se rencontrent en tout temps sauf lorsque la condition précédente est satisfaite. Les agents résolvent également le problème de rendez-vous avec la détection dans les mêmes conditions. Dans ce dernier cas, si la condition n'est pas satisfaite et que les agents ne peuvent accomplir le rendez-vous, ils s'immobilisent et déclarent que le rendez-vous est impossible. Dans cet article, il y a 5 résultats principaux. Premièrement, deux agents avec $o(\log n)$ bits de mémoire et avec un nombre constant de jetons fixes chacun ne peuvent pas accomplir le rendez-vous dans un tore $n\times n$. Deuxièmement, deux agents avec $o(\log n)$ bits de mémoire et un seul jeton amovible chacun ne peut pas accomplir le rendez-vous dans un tore $n\times n$. Troisièmement, il existe deux agents avec $O(\log n)$ bits de mémoire et un jeton fixe chacun qui peuvent résoudre le

problème de rendez-vous avec détection dans un tore $n \times n$. Quatrièmement, il existe deux agents avec un nombre constant de bits de mémoire et deux jetons amovibles chacun qui peuvent résoudre le problème de rendez-vous avec (ou sans) détection dans un tore $n \times n$ (respectivement $n \times m$). Finalement, il existe deux agents avec un nombre constant de bits de mémoire et trois jetons amovibles chacun qui peuvent résoudre le problème de rendez-vous pour n'importe quel tore $n \times m$.

Les résultats précédents considéraient le rendez-vous pour 2 agents. Ci-dessous, nous allons présenter des résultats concernant le cas plus général où un plus grand nombre d'agents doivent de rencontrer (bien que deux agents soient également considérés). Dans [32], un modèle très semblable au modèle étudié dans [43] est présenté, sauf que le rendez-vous dans ce modèle ne peut se produire que sur les noeuds du réseau. Les auteurs considèrent le problème de rendez-vous pour $k \geq 2$ agents. Ils étudient la faisabilité du rendez-vous et donnent des algorithmes pour le faire lorsque c'est possible. Si n et k sont inconnus des agents, alors le rendez-vous est impossible. Dans le reste de l'article on suppose que k est connu. Dans ce cas, les auteurs ont prouvé que le rendez-vous est possible, si la suite des distances entre les jetons est apériodique, c'est-à-dire qu'il n'existe aucune rotation de cette configuration d'agents qui la porte sur elle même. Il y a trois algorithmes principaux présentés pour les configurations apériodiques. Ces algorithmes établissent un compromis entre le temps de fonctionnement et la mémoire des agents. Le premier algorithme fonctionne en temps $O(n)$ et utilise $O(k \log n)$ bits de mémoire, le second fonctionne en temps $O(kn)$ et utilise $O(\log n)$ bits de mémoire, le troisième fonctionne en temps $O(\frac{n \log n}{\log \log n})$ et utilise $O(k \log \log n)$ bits de mémoire. En ce qui concerne l'anneau, le livre [44] donne davantage d'informations sur le problème de rendez-vous dans l'anneau.

Une autre topologie de réseaux qui est intéressante à considérer est l'arbre. Dans [34], les auteurs montrent que si le délai entre le temps de départ des agents est arbitraire, alors la borne inférieure sur la mémoire est de $\Omega(\log n)$ bits, même pour la ligne de grandeur n. Ce résultat en relation avec la borne supérieure identique dans [34] montre que la mémoire requise est donc de $\Theta(\log n)$. Toutefois, lorsque les agents débutent en même temps, la quantité de mémoire requise dépend de deux paramètres dans l'arbre : le nombre n de noeuds ainsi que le nombre l de feuilles. Ils montrent que deux agents identiques avec $O(\log l + \log \log n)$ bits de mémoires résolvent le problème de

rendez-vous pour n'importe quel arbre qui contient n noeuds et l feuilles. Pour la classe des arbres contenant $O(\log n)$ feuilles, cela montre une différence exponentielle entre la quantité de mémoire minimale pour le scénario avec les délais de départ différents et quand il n'y a pas de délai. De plus, les auteurs montrent que la borne atteinte lorsque le nombre de noeuds est n et le nombre de feuilles est l est optimal, même dans la classe des arbres dont le degré est borné par 3. En fait, pour un nombre infini de nombres entiers l, ils montrent une classe d'arbres arbitrairement grands avec un degré maximum de 3 et avec l feuilles tel que le rendez-vous demande $\Omega(\log l)$ bits de mémoire. Cela implique qu'en combinant ce résultat et celui de [34], la borne supérieure $O(\log l + \log\log n)$ ne peut pas être améliorée et cela même pour la classe des arbres avec un degré maximum de 3.

Alors que dans [34], le problème de la quantité minimale de mémoire nécessaire pour effectuer le rendez-vous dans les arbres est étudié, le même problème est considéré dans [18] mais pour la classe des graphes connexes arbitraires. Les auteurs démontrent la quantité de mémoire minimale nécessaire pour garantir le rendez-vous lorsque c'est faisable. Ils montrent que cette quantité minimale est de l'ordre de $\Theta(\log n)$, où n est le nombre de noeuds du graphe, peu importe le décalage entre les départs des agents. En fait, les auteurs utilisent des agents identiques munis de $\Theta(\log n)$ bits de mémoires afin de résoudre le problème de rendez-vous dans n'importe quel graphe contenant n noeuds lorsque les agents débutent avec n'importe quel décalage et ils montrent une borne inférieure $\Omega(\log n)$, égale à la performance de leur algorithme.

Dans [25], les auteurs analysent le coût du rendez-vous de deux agents dans un graphe connexe arbitraire inconnu. Les résultats démontrés peuvent également être appliqués à un nombre arbitraire d'agents. Dans ce modèle, les agents ont des nombres entiers positifs comme identificateurs distincts et chaque agent ne connait à priori que son identificateur. Les agents ne connaissent pas la topologie du graphe dans lequel le rendez-vous doit être fait. L'agent ne voit pas l'extrémité d'une arête qui n'a pas été explorée. Si un agent décide de parcourir une arête inconnue (en empruntant un numéro de port qu'il n'a pas visité jusqu'à maintenant), le choix de l'arête sélectionné est fait par l'adversaire afin d'envisager le pire scénario. Les agents ont des capacités de mémoire illimitées et l'objectif des auteurs est d'optimiser le coût du rendez-vous. Ce coût est mesuré en fonction du nombre de rondes en pire

cas après le départ du second agent jusqu'à ce que les agents se rencontrent. Afin de comprendre plus en détail leurs résultats, nous utiliserons les notations suivantes. Les identificateurs des agents sont L_1 et L_2, l'identificateur le plus petit entre les deux est dénoté par l, le décalage de départ entre les deux agents est dénoté par τ, le nombre de noeuds est dénoté par n, et la distance entre les positions initiales des agents par D. Les auteurs montrent que le rendez-vous peut être garanti au coût de $O(n + \log l)$ pour n'importe quel arbre de grandeur n même avec un décalage de départ arbitraire. Les auteurs montrent également que pour certains arbres, cette borne ne peut pas être améliorée même s'il n'y a pas de décalage de départ. Si le graphe où les agents évoluent contient des cycles, la technique utilisée est plus complexe. Les auteurs démontrent que dans l'anneau, lorsque les agents n'ont pas de décalage sur leur départ, le coût optimal de rendez-vous est de $\Theta(D \log l)$ et ils donnent un algorithme rencontrant cette borne. Lorsqu'il y a un décalage, les auteurs présentent deux algorithmes : si n est connu ($O(n \log l)$) et si n est inconnu ($O(l\tau + ln^2)$). Les auteurs donnent également une borne inférieure : $\Omega(n + D \log l)$. En ce qui a trait aux graphes connexes arbitraires, les auteurs donnent un algorithme avec coût de $O(n^5 \sqrt{\tau \log l} \log n + n^{10} \log^2 n \log l)$ qui résout le problème de rendez-vous pour n'importe quel graphe G de grandeur n pour tous les identificateurs $L_1 > L_2 = l$ et pour les décalages τ. Finalement, les auteurs montrent une borne inférieure $\Omega(n^2)$ sur le coût du rendez-vous dans certaines familles de graphes. Si le démarrage simultané est supposé, ils construisent un algorithme de rendez-vous générique, fonctionnant pour tous les graphes connexes, et qui est optimal pour la classe de graphes de degré borné, si la distance entre les agents est également bornée.

Dans [25], les auteurs ont posé un problème ouvert concernant la dépendance du coût du rendez-vous en fonction du paramètre τ. Existe-t-il un algorithme déterministe de rendez-vous pour les graphes connexes arbitraires avec un coût polynomial en fonction de n et de l mais qui ne dépend pas de τ ? Dans [42], une réponse positive à cette question a été présentée. Encore une fois, les auteurs ont présenté leur résultat pour deux agents bien qu'ils aient expliqué comment leurs résultats s'appliquent à un plus grand nombre d'agents en autant que les agents qui se rencontrent sur un noeud puissent partager des informations. Les auteurs présentent un algorithme fonctionnant en temps $O(\log^3 l + n^{15} \log^{12} n)$. Également, les auteurs ont amélioré un résultat pour le rendez-vous dans un anneau, préalablement présenté dans [25]. Ils ont donné un algorithme fonctionnant en temps $O(n \log l)$ même lorsque

n est inconnu.

Nous concluons cette section en présentant un résultat qui répond à une question posée dans [25]. Bien que les algorithmes dans [25] et [42] ont un coût polynomial, ils prennent un temps considérable de calcul local dû à l'utilisation d'un objet combinatoire dont l'existence est prouvé par la méthode probabiliste. Comme les auteurs le mentionnent, les agents peuvent trouver cet objet par une recherche exhaustive mais pour ce faire, ils augmentent le temps des calculs locaux de manière exponentielle (bien que cela n'influence pas directement le coût de l'algorithme qui est définie comme le nombre de rondes). Dans [54] les auteurs ont résolu ce problème en utilisant la notion de séquence d'exploration universelle (UXS) [41]. Soit $(a_1, a_2, \ldots, a_k)$ une séquence d'entiers. Une application de cette séquence sur un graphe G au noeud u est la séquence de noeuds $(u_0, \ldots, u_{k+1})$ obtenu comme suit : $u_0 = u$, u_1 est le noeud joint à u par l'arête correspondant au port 0 localisé sur le noeud u ; pour tous $1 \leq i \leq k$, u_{i+1} est le noeud joint à u_i par l'arête correspondante (localisé sur le noeud u_i) au port $(p + a_i)$ mod $d(u_i)$, où p est le numéro de ports localisé à u_i correspondant à l'arête $\{u_i, u_{i-1}\}$ et $d(u_i)$ dénote le degré du noeud u_i. En utilisant cette notion, les auteurs de [54] proposent un algorithme de rendez-vous dont non seulement le coût, mais aussi le temps de calculs locaux est polynomial.

Modèle asynchrone

Dans le modèle asynchrone, les agents n'utilisent plus l'horloge afin de synchroniser leurs mouvements. Ils ont toujours le pouvoir de déterminer à quel endroit ils veulent se déplacer mais cette fois, le déplacement en lui-même est contrôlé par l'adversaire, ce qui complique considérablement le rendez-vous. Dans la littérature, il y a quelques modèles importants, dont un en particulier que nous aimerions présenter dans cet état de l'art. Le modèle CORDA utilisé dans l'environnement géométrique [50, 51, 53] a été adapté à l'environnement réseau. Ce modèle suppose que les agents sont vraiment faibles en terme de mémoire puisqu'ils ne conservent aucun historique de leurs déplacements ou de leurs perceptions, mais toutefois, leurs capacités de perception sont puissantes ; ils peuvent observer le réseau en entier incluant la position des agents à l'intérieur de ce réseau. Ce modèle adapté de CORDA a été notamment utilisé dans [40], [39], et [21] qui étudient le rendez-

vous de plusieurs agents dans l'anneau qui ne contient pas de numéros de ports et où les noeuds (et les agents) ne possèdent pas d'identificateurs. Les auteurs se sont concentrés sur le problème de la faisabilité du rendez-vous. Autrement dit, ils se sont intéressés non seulement aux algorithmes pour accomplir le rendez-vous mais également à caractériser les configurations pour lesquelles le rendez-vous était faisable. Chaque agent fonctionne de manière asynchrone en cycles d'observation, de calcul et de déplacement lorsque les calculs indiquent qu'ils doit se déplacer. La seule contrainte est que chaque mouvement se produit de manière instantanée (donc, les agents sont toujours perçus sur les noeuds) et par conséquent, le rendez-vous ne peut se produire que sur un noeud. C'est d'ailleurs en cela que le modèle CORDA originel diffère de celui-ci. Dans le modèle originel, les agents étaient perceptibles lorsqu'ils étaient en déplacement. Il est important de souligner qu'un agent peut capturer une vue de la configuration, faire des calculs et qu'au moment de se déplacer, la configuration n'est possiblement plus dans le même état qu'au moment de l'observation. Il en résulte que les agents peuvent bouger sur la base d'informations qui sont désuètes. Une précision importante doit-être faite pour parler de ce résultat : la capacité qu'ont les agents de percevoir les multiplicités [13, 51] lorsqu'ils observent la configuration. Nous disons que, s'il y a plus d'un agent sur un même noeud, alors ces agents forment une multiplicité. Notons que la capacité de détection de multiplicité dans [40], [39], et [21] permet uniquement de savoir s'il n'y a aucun agent, un agent ou plus d'un agent mais à aucun moment il n'est possible de faire la différence entre un noeud occupé et un autre, si les deux contiennent des nombres différents d'agents supérieurs à un. Il a été prouvé dans [40] que sans la capacité de détection, le rendez-vous est impossible.

La contribution principale de [40] est la solution du problème de rendez-vous pour toutes les configurations initiales d'un nombre impair d'agents. Les auteurs prouvent que pour un nombre impair d'agents, le rendez-vous est possible dans l'anneau lorsque la configuration initiale n'est pas périodique et ils donnent un algorithme qui accomplit le rendez-vous. (Une configuration est périodique si elle est fixée par une rotation non triviale de l'anneau dans le sens où une telle rotation bouge les noeuds occupés sur les noeuds occupés et les noeuds vides sur les noeuds vides). Les auteurs ont également prouvé que si le nombre d'agents est pair, le rendez-vous est impossible s'il y a exactement deux agents, ou si la configuration initiale est périodique, ou s'il y a un axe de symétrie sur lequel il n'y pas de noeud. Les auteurs donnent également

un algorithme pour toutes les configurations initiales rigides (c'est-à-dire les configurations qui ne sont pas préservées par aucun automorphisme non-trivial de l'anneau). Ils ont laissé comme problème ouvert les configurations symétriques, apériodiques contenant un nombre pair d'agents et ayant un axe de symétrie passant par au moins un noeud.

Des auteurs de [39] ont donné une solution complète positive du problème ouvert de [40] lorsqu'il y a plus de 18 agents. Cela laisse ouvert les configurations particulières mentionnées ci-haut lorsqu'il y a exactement 6, 8, 10, 12, 14, 16, ou 18 agents. Les auteurs de [21] ont repris ce problème et ont résolu le cas pour exactement 6 agents. Finalement, dans [22], ces mêmes auteurs ont résolus le problème complètement.

Dans [38], un scénario différent a été envisagé. Cette fois, les agents ont une capacité de détection de multiplicité locale. C'est-à-dire qu'ils ne peuvent percevoir une multiplicité que s'ils en font partie. Cela implique qu'un agent peut percevoir qu'un autre noeud est occupé mais il lui est impossible de dire par combien d'agents. En utilisant cette hypothèse plus faible, les auteurs considèrent le rendez-vous pour un nombre arbitraire d'agents qui satisfait $2 < k \leq \lfloor \frac{n}{2} \rfloor - 1$ dans un anneau de grandeur n et où la configuration initiale est rigide. Ils ont donné un algorithme qui accomplit le rendez-vous en nombre de rondes asynchrones $O(n)$. Notons qu'une ronde asynchrone est définie comme étant le plus court fragment d'une exécution, dans lequel chaque agent est activé, et se déplace vers le noeud voisin ou reste sur le noeud courant au moins une fois.

Dans [7], le compromis entre la quantité de mémoire et le temps pour effectuer le rendez-vous est analysé. Les auteurs considèrent le rendez-vous dans les arbres avec un nombre arbitraire d'agents anonymes qui ne peuvent pas percevoir les autres noeuds, mais qui perçoivent les ports sur le noeud où ils se trouvent. Dans ce scénario spécifiquement, les auteurs affaiblissent les exigences pour accomplir le rendez-vous. C'est-à-dire que le rendez-vous se fait sur un noeud dans le cas d'un arbre asymétrique et sur deux noeuds adjacents dans la cas inverse. Ils montrent que la borne inférieure sur le temps de rendez-vous pour les arbres avec n noeuds est de $\Omega(n)$ et que, afin d'obtenir cette borne, $\Omega(n)$ bits de mémoire sont requis pour chaque agent. Ils donnent un algorithme qui accomplit le rendez-vous en temps $O(n)$ en utilisant $O(n)$ bits de mémoire par agent. Et puis les auteurs montrent un autre algorithme

qui fonctionne en temps polynomial mais qui utilise seulement $O(\log n)$ bits de mémoire.

Dans [24], les auteurs utilisent un autre modèle d'asynchronie pour le rendez-vous de deux agents. Les agents n'ont aucune vision du réseau en dehors du noeud visité actuellement. Un adversaire asynchrone peut arbitrairement varier la vitesse de l'agent, l'arrêter, voir le bouger d'avant en arrière, aussi longtemps que la marche de l'agent dans chaque segment de son parcours est continue, ne le quitte pas et le couvre en entier. Dans ce modèle, les agents ont des identificateurs, ce qui permet de distinguer un agent d'un autre. Le rendez-vous doit-être garanti pour n'importe quel comportement de l'adversaire asynchrone. La mesure de performance est le coût du rendez-vous, défini comme le nombre total des traversées d'arêtes par les agents. Premièrement, les auteurs considèrent le rendez-vous dans la ligne infinie. Pour des agents localisés à distance D dans la ligne, ils montrent un algorithme de rendez-vous au coût de $O(D|L_{min}|^2)$ si D est connu et $O((D|L_{max}|)^3)$ lorsque ce n'est pas le cas, ou $|L_{min}|$ et $|L_{max}|$ correspondent à la longueur de l'identificateur le plus court et le plus long respectivement. Ce résultat tient toujours pour l'anneau de dimension connu ou non mais les auteurs donnent deux autres algorithmes pour les anneaux, qui ne dépendent pas du paramètre D, et dont le premier est au coût optimal de $O(n|L_{min}|)$ si la grandeur de l'anneau est connue et l'autre lorsque la grandeur n'est pas connue au coût de $O(n|L_{max}|)$. Pour les graphes arbitraires, les auteurs prouvent que le rendez-vous est faisable si une borne supérieure sur la grandeur du graphe est connue. Ils donnent également un algorithme optimal fonctionnant au coût de $O(D|L_{min}|)$ si la topologie du graphe ainsi que les positions des agents sont connues par les agents.

Dans [20], le problème de la faisabilité du rendez-vous pour les graphes connexes arbitraires posé précédemment dans [24] est résolu. Les auteurs proposent un algorithme qui garantit le rendez-vous asynchrone pour n'importe quel graphe dénombrable (fini ou infini) pour des positions de départ arbitraires. L'algorithme de rendez-vous est basé sur la notion d'un tunnel. Considérons un graphe G et deux routes R_1 et R_2 partant des noeuds v et w respectivement. Une route est une séquence d'arêtes, telle que les arêtes consécutives dans la séquence sont incidentes dans G. Les routes R_1 et R_2 forment un tunnel s'il existe un préfixe $[e_1, e_2, \ldots, e_n]$ de la route R_1 et un préfixe $[e_n, e_{n-1}, \ldots, e_1]$ de la route R_2, pour certaines arêtes e_i dans le graphe,

tel que $e_i = \{v_i, v_{i+1}\}$, où $v_1 = v$ et $v_{n+1} = w$. De manière plus intuitive, la route R_1 a un préfixe P se terminant sur w et la route R_2 a un préfixe se terminant sur v et qui est l'inverse de P. Les auteurs ont montré que si la route R_1 et la route R_2 forment un tunnel, alors le rendez-vous est garanti, peu importe les actions de l'adversaire asynchrone.

Pour cet algorithme, nous allons donner une idée de sa conception puisque ce résultat est en relation directe avec le résultat que nous présentons au chapitre 4. L'idée ici est de forcer la route de deux agents possédant différents identificateurs à former un tunnel, pour n'importe quelle combinaison des noeuds de départ ainsi que des identificateurs des agents. Les auteurs le font en énumérant tous les quadruples (i, j, s', s''), tel que i et j sont des entiers positifs différents et s', s'' sont des séquences finies d'entiers positifs, et les arrangent en une séquence dénombrable infinie. Cette énumération est la même pour chaque agent qui l'exécute dans l'ordre. N'importe quelle configuration de départ avec les agents ayant les identificateurs i situés sur le noeud v et j situés sur le noeud w, sélectionnée par l'adversaire, correspond à un quadruple (i, j, s', s''), tel que s' est une séquence de ports qui induit un chemin de v vers w et s'' est une séquence de ports qui induit un chemin inverse de w vers v. Durant l'exécution d'un quadruple, un suffixe est ajouté à la partie déjà construite du segment initial de la route des agents qui ont les identificateurs i et j d'une telle manière que si les identificateurs et les positions initiales des agents correspondent au quadruple exécuté, les routes des agents forment un tunnel. Puisque pour un quadruple quelconque cette condition sera toujours vraie, les agents placés sur des positions arbitraires dans le graphe vont se rencontrer.

Le coût de l'algorithme dépend de l'énumération des quadruples et plus précisément de la position du quadruple qui correspond à la position initiale des agents. Il en résulte que le coût en pire cas de cet algorithme est au moins exponentiel en grandeur du graphe (si celui-ci est fini). La question suivante en découle. Existe-t-il un algorithme déterministe asynchrone, qui fonctionne pour les graphes finis arbitraires, qui a un coût polynomial en fonction des identificateurs des agents ainsi qu'en fonction de la grandeur du graphe ?

Une solution partielle à ce problème a été donnée dans [9, 15] pour certains graphes spécifiques et avec des conditions additionnelles. Dans [15], les auteurs considèrent le rendez-vous dans une grille à deux dimensions,

avec les ports qui sont identifiés de manière régulière N, E, S, W et les agents ont une boussole et connaissent leurs coordonnées initiales dans la grille. Les auteurs ont prouvé que sous ces suppositions, le rendez-vous peut être accompli au coût de $O(d^{2+\epsilon})$ où d est la distance initiale entre les agents et ϵ est un nombre réel positif arbitraire. Puis, ce résultat a été amélioré dans [9] sous les mêmes suppositions. Les auteurs donnent un algorithme fonctionnant dans la grille infinie à δ dimensions au coût de $O(d^{\delta} polylog(d))$. Cette complexité se rapproche de la borne inférieure de $\Omega(d^{\delta})$.

Finalement, une solution complète du problème de [20] a été donnée par Dieudonné et al. [26] à l'aide d'un algorithme asynchrone pour le rendez-vous avec coût polynomial en fonction de la grandeur du graphe ainsi que la longueur de la plus petite étiquette des agents. À l'aide de leur algorithme, ils ont également résolu plusieurs problèmes fondamentaux impliquant des équipes de grandeur inconnue supérieure à 1 d'agents étiquetés se déplaçant de manière asynchrone dans des graphes inconnus. Parmis ces problèmes, il y a notamment celui du calcul de la grandeur d'une équipe, dans lequel chaque agent doit trouver le nombre total d'agents, l'élection du chef, dans lequel les agents doivent trouver l'étiquette d'un unique agent qui sera le chef, le changement de noms parfait, dans lequel chaque agent doit adopter une nouvelle étiquette de l'ensemble $\{1, \ldots, k\}$, où k est le nombre d'agents et finalement le commérage, dans lequel chaque agent possède initialement une partie de l'information et chaque agent doit connaître la totalité de l'information.

Dans [26] un algorithme polynominal pour tous ces problèmes a été proposé dans le scénario des agents qui se déplacent de façon asynchrone.

Chapitre 3

Rendez-vous asynchrone des agents mobiles anonymes avec une vision locale dans les graphes bipartis réguliers

Nous considérons le problème de rendez-vous pour des agents identiques, sans mémoire, qui doivent se rencontrer dans un noeud du graphe anonyme. Ils opèrent en cycles d'observation, de calcul et de déplacement, et doivent terminer sur le même noeud. Lors d'un cycle, un agent observe son voisinage immédiat, c'est-à-dire le noeud où il se trouve et les voisins de ce noeud (observation), prend une décision de demeurer où il est ou de bouger sur l'un de ses voisins (calcul) et, dans le dernier cas, il se déplace sur ce voisin (déplacement). Les cycles sont accomplis de façon asynchrone pour chaque agent. La nouveauté de ce modèle, par rapport à la littérature existante sur les problèmes de rendez-vous des agents mobiles anonymes, est que les agents ont une perception vraiment restreinte : ils ne peuvent voir que leur voisinage immédiat.

Une configuration initiale d'agents est dite «regroupable» s'il existe un algorithme qui réunit tous les agents d'une configuration sur un seul noeud et laisse les agents en mode «attente» à partir de ce moment, peu importe les actions d'un adversaire asynchrone. L'algorithme peut même être dédié à cette configuration particulière. Le problème de rendez-vous est de déterminer

quelles configurations sont regroupables et trouver un algorithme universel qui peut accomplir le rendez-vous pour chaque configuration regroupable. Nous donnons la solution complète du problème de rendez-vous pour les graphes bipartis réguliers. Notre principale contribution est la preuve que la classe des configurations initiales regroupables est très petite : elle consiste en des étoiles (un agent A avec les agents qui lui sont adjacents) de grandeur au moins trois. Du côté positif, nous donnons un algorithme qui accomplit le rendez-vous lorsque c'est faisable.

3.1 Problème et modèle

Des efforts considérables ont été déployés pour étudier le rendez-vous dans des scénarios très faibles, où les agents représentent des appareils simples qui pourraient être massivement produits. Un de ces scénarios est le modèle CORDA, originalement formulé pour les agents opérant dans le plan [12, 13, 14, 33, 50, 51, 53] et puis adapté à l'environnement réseau [38, 39, 40]. Dans ce résultat, nous étudions le problème de rendez-vous dans les réseaux, à l'aide d'un scénario encore plus faible que celui précédemment mentionné. Nous allons décrire notre modèle et souligner les différences par rapport à celui de [38, 39, 40].

Considérons un graphe simple non-orienté. Les noeuds et arêtes du graphe sont anonymes. Initialement, certains noeuds du graphe sont occupés par des agents anonymes (c.-à-d. identiques) et il y a, au plus, un agent par noeud. L'objectif est de regrouper les agents dans un noeud du graphe et de les immobiliser. Les agents opèrent en cycles Observation-Calcul-Déplacement. Lors d'un cycle, un agent capture son voisinage immédiat (Observation) puis, sur la base de ses observations, prend une décision de demeurer au même endroit ou de se déplacer (Calcul) et, dans le dernier cas, fait un mouvement instantané sur le voisin déterminé (Déplacement). Les cycles sont accomplis de manière asynchrone pour chaque agent. Cela veut dire que le temps entre les opérations d'observation, de calcul ainsi que de déplacement est fini mais non-borné, et est décidé par l'adversaire pour chaque agent. La seule contrainte est que les agents bougent instantanément et, par conséquent, chaque agent observant son environnement verra les autres agents sur le même noeud ou sur un noeud adjacent mais jamais sur une arête lorsqu'un autre agent bouge.

Toutefois, un agent A peut faire une opération d'observation à un temps t, percevoir des agents sur des noeuds adjacents, faire le calcul d'un noeud où se déplacer lors du temps $t' > t$, et puis bouger sur ce noeud voisin durant un temps futur $t'' > t'$ pour lequel les agents ne sont plus localisés aux mêmes endroits que lorsqu'ils étaient observés par A en temps t, parce que durant ce laps de temps, d'autres agents ont exécuté des déplacements. Il en résulte que les agents peuvent bouger sur la base d'observations qui ne sont plus d'actualité, ajoutant une difficulté supplémentaire au problème de rendez-vous. Il est important de souligner que les agents ne possèdent aucune forme de mémoire des événements passés. Cela implique que le noeud ciblé par un agent pour un déplacement (qui est la position actuelle ou un noeud adjacent à sa position) est décidé par l'agent durant l'opération de calcul uniquement basé sur les observations qu'il a faites de la position des autres agents durant le même cycle. Les agents sont anonymes et exécutent la même version de l'algorithme déterministe. Ils ne peuvent pas laisser d'indices sur les noeuds visités ou envoyer des messages aux autres agents.

La seule différence entre notre scénario et celui de [38, 39, 40] est à propos de ce que l'agent perçoit durant l'opération d'observation. Bien que dans les résultats mentionnés ci-haut, les agents soient capables de percevoir la configuration entière de tous les agents dans le graphe, dans ce chapitre, nous assumons que les agents voient uniquement leur voisinage immédiat ; autrement dit, ils perçoivent uniquement les agents localisés au même endroit qu'eux ou sur un noeud adjacent. La raison de ce changement de modèle est l'applicabilité. Il est en effet difficile d'imaginer qu'un agent avec des capacités si faibles qu'il ne peut pas avoir de mémoire et autres capacités, soit capable de percevoir un système réseau en entier, alors que de percevoir son voisinage immédiat est facilement réalisable en échangeant des signaux entre les noeuds voisins.

Une capacité importante, étudiée en profondeur dans la littérature sur les agents, est la capacité de détection de multiplicité [33, 38, 39, 40, 51]. C'est la capacité qu'un agent a de percevoir, durant l'opération d'observation, s'il y a un seul ou plusieurs agents sur une position précise. Il a été prouvé dans [40] que sans cette capacité, le rendez-vous dans les réseaux est généralement impossible (même pour l'anneau et même si les agents peuvent faire une observation du réseau en entier). Toutefois, il a été démontré dans [38] que la capacité de détection de multiplicité peut être affaiblie et s'appliquer unique-

ment sur le noeud sur lequel l'agent est présentement localisé. Dans notre résultat, nous considérons aussi bien la version faible que celle plus forte de la détection de multiplicité. Notre résultat négatif reste vrai même avec la détection plus forte de multiplicité (qui veut dire dans notre cas que l'agent perçoit la multiplicité où il est situé présentement ou sur un noeud voisin) et notre résultat positif reste vrai pour la détection de multiplicité plus faible (lorsqu'un agent perçoit qu'il y a plus d'un agent sur un noeud uniquement lorsqu'il est lui-même sur ce noeud). Il est important de souligner que lorsqu'un agent observe un noeud, il peut seulement dire si celui-ci est vide (il n'y a pas d'agent), s'il y a un agent, ou s'il y a plus d'un agent. L'agent ne fait pas la différence entre un noeud occupé par a agents ou par b agents, pour les valeurs $a, b > 1$ distinctes.

Dans ce chapitre, nous étudions le problème de rendez-vous dans les graphes réguliers bipartis. Une configuration d'agents est dite *regroupable* s'il existe un algorithme de rendez-vous qui regroupe tous les agents de la configuration sur un seul noeud et les garde inactifs par la suite, peu importe les actions de l'adversaire asynchrone (l'algorithme peut même être dédié à une configuration particulière). Le problème de rendez-vous est de déterminer quelles configurations initiales sont regroupables et trouver un algorithme (universel) qui accomplit le rendez-vous pour toutes les configurations regroupables.

3.2 Discussion des hypothèses

Nous aimerions argumenter que notre modèle est fort possiblement le modèle le plus faible permettant aux problèmes de rendez-vous d'être étudiés significativement dans les réseaux. Le modèle CORDA adapté aux réseaux dans [38, 39, 40] est déjà extrêmement faible, puisque les agents n'ont pas de mémoire des événements passés et que l'adversaire asynchrone a un pouvoir absolu d'ordonnancer arbitrairement les cycles d'observation, calcul et déplacement de chaque agent. De plus, les agents sont identiques (anonymes). Il en résulte que les protocoles de rendez-vous ne peuvent pas utiliser d'identificateurs pour briser la symétrie. Toutefois, la capacité de perception des agents était préalablement assumée comme étant très puissante, résultant en un contraste avec les autres capacités telles que la mémoire qui est absente.

Sous notre modèle, la perception est également réduite à la perception du voisinage immédiat. Nous pouvons constater qu'une restriction plus poussée de la perception rendait le rendez-vous impossible puisque les agents ne voyant que leur propre noeud ne peuvent pas se rencontrer. En ce qui a trait à la détection de multiplicité, nous avons résolu le problème de rendez-vous tant pour la détection faible de multiplicité que pour la détection forte de multiplicité.

Il reste à discuter les hypothèses concernant la classe de réseaux dans lesquelles les agents opèrent. Considérons les graphes bipartis réguliers. C'est une large classe de réseaux incluant, entre autre, les cycles pairs, les tores multidimensionnels avec au moins un côté pair et les hypercubes. Notons que notre résultat demeure valide (tel quel) dans les cycles arbitraires de dimension différente de trois et dans les tores multidimensionnels arbitraires avec chaque dimension de grandeur différente de trois. Pour les tores multidimensionnels avec au moins une dimension de grandeur trois, le résultat changera de la façon suivante. Pour la détection forte de multiplicité, les configurations regroupables sont exactement des étoiles de grandeur au moins trois avec au moins un agent ayant exactement un agent adjacent et, pour la détection faible de multiplicité, les configurations regroupables sont exactement les étoiles de grandeur au moins trois avec un seul agent ayant plus d'un agent lui étant adjacent. Les techniques développées dans ce chapitre peuvent être facilement adaptées pour couvrir les cas ci-haut mentionnés.

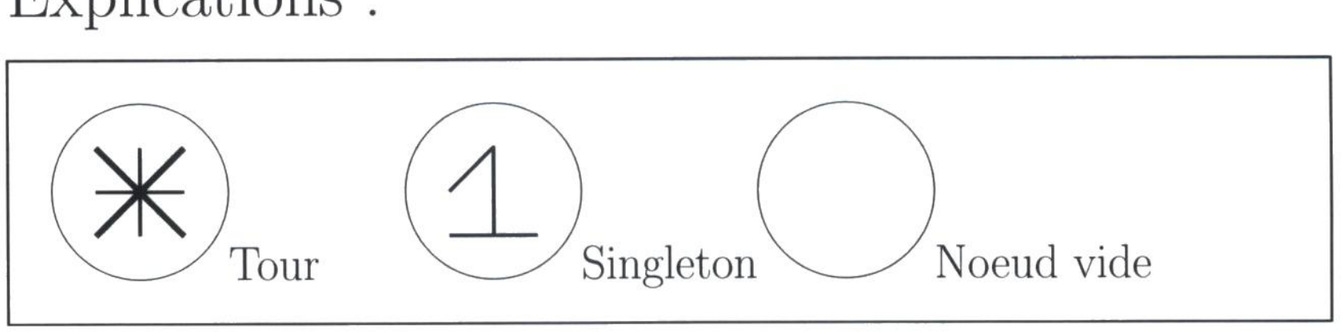

FIGURE 3.1 – Explications des symboles dans les figures du chapitre 3

Notons que d'enlever l'hypothèse de la régularité du graphe rend la solution au problème de rendez-vous impossible, même dans la classe des arbres. (Pour les graphes non-réguliers, la vision locale d'un agent voudrait, de plus, dire qu'il percevrait le degré de son propre noeud et des noeuds de son voisi-

nage immédiat). Pour observer cela, considérons les arbres suivants : T_1 est la ligne à cinq noeuds avec une feuille attachée aux extrémités, une feuille attachée aux voisins des extrémités, et deux feuilles attachées au centre (voir Fig. 3.2) ;

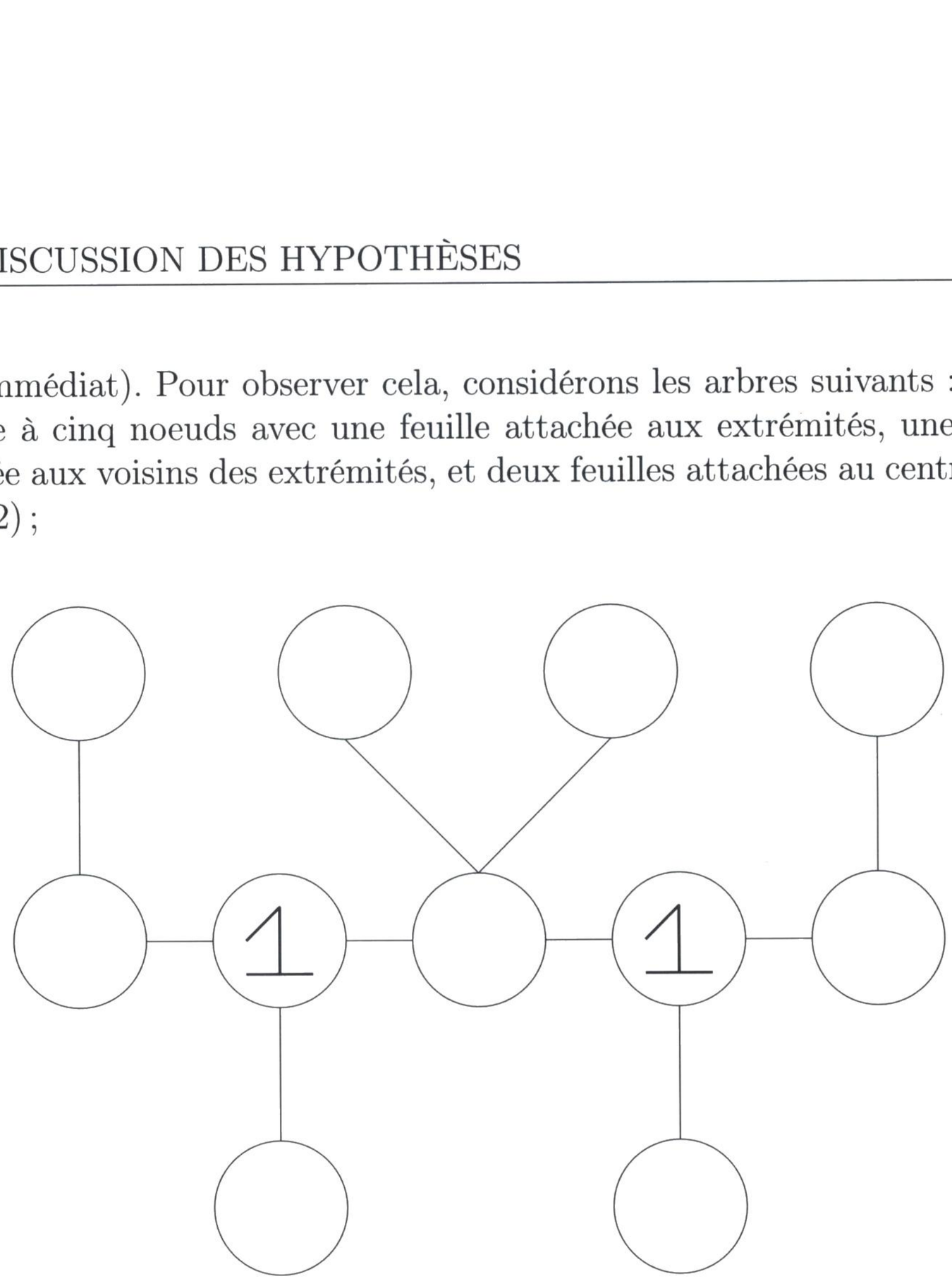

FIGURE 3.2 – L'arbre T_1 avec le profil des degrés de ligne 2-3-4-3-2

T_2 est la ligne à cinq noeuds avec trois feuilles attachées aux extrémités et une feuille attachée aux voisins des extrémités (voir Fig. 3.3).

Donc, T_1 a le profil des degrés de ligne 2-3-4-3-2 et T_2 a le profil des degrés de ligne 4-3-2-3-4. Maintenant, considérons la configuration initiale dans chacun de ces arbres consistant en deux agents localisés sur les noeuds de degré trois. Chacune de ces configurations est regroupable. L'algorithme pour la configuration de l'arbre T_1 est : si tu es sur un noeud de degré trois, va sur un noeud de degré quatre et si tu es sur un noeud de degré quatre, arrête. L'algorithme pour la configuration de l'arbre T_2 est : si tu es sur un noeud de degré trois, va sur un noeud de degré deux et si tu es sur un noeud de degré deux, arrête. Toutefois, il n'y a pas d'algorithme universel permettant d'accomplir le rendez-vous pour ces deux configurations.

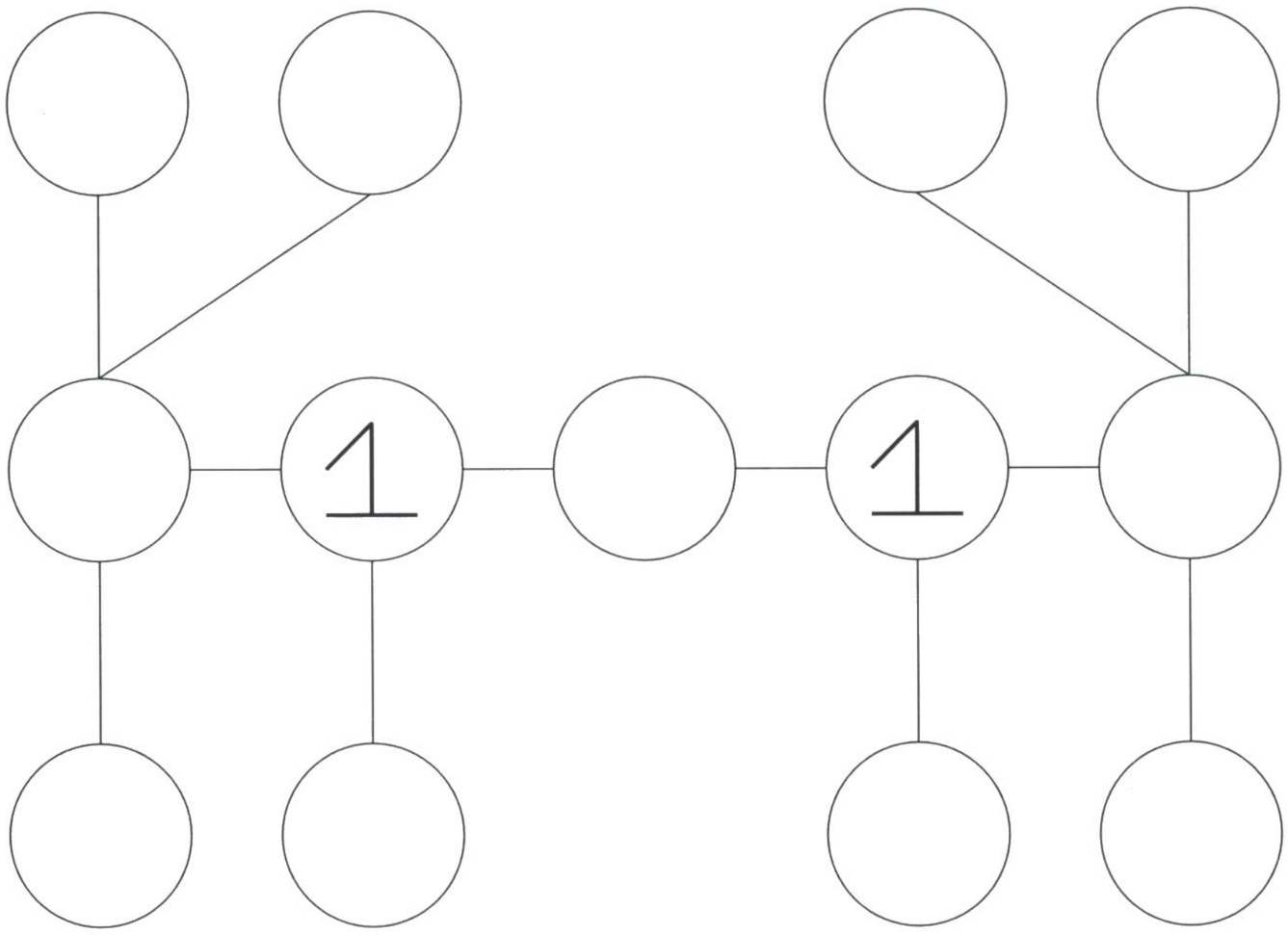

FIGURE 3.3 – L'arbre T_2 avec le profil des degrés de ligne 4-3-2-3-4

Un adversaire planifiant les actions des deux agents de façon synchrone peut les garder séparés dans l'une ou l'autre de ces configurations.

Il reste à considérer l'hypothèse que le graphe est un graphe biparti. Comme mentionné précédemment, il est possible d'étendre la solution à quelques autres graphes réguliers. Toutefois, le problème de rendez-vous dans les graphes arbitraires réguliers demeure ouvert.

3.3 Terminologie et préliminaires

Un noeud sur lequel il n'y a pas d'agents est appelé *vide*, autrement il est appelé *occupé*. Un agent étant le seul à occuper un noeud est appelé un *singleton* et plus d'un agent occupant un noeud forment une *tour*. Une configuration est une fonction sur l'ensemble des noeuds du graphe avec les valeurs dans l'ensemble $\{vide, singleton, tour\}$. Une configuration initiale ne contient pas de tour. Nous disons que les singletons sont adjacents (resp. les

tours sont adjacentes, un singleton est adjacent à une tour) si les noeuds respectifs sont adjacents.

Puisque ni les noeuds, ni les arêtes, ni les agents sont étiquetés et que le graphe est régulier de degré δ, durant l'opération d'observation, un agent situé sur un noeud v obtient l'entrée (x, S), ou la valeur x indique le statut du noeud v : soit singleton ou tour, et la valeur de S, décrivant le statut des voisins de v, est un multi-ensemble de grandeur δ contenant des éléments qui sont *vide*, *singleton* ou *tour*. Considérant ces entrées, l'agent décide d'attendre ou de bouger sur un élément de S. S'il y a plusieurs éléments de même type (*vide*, *singleton* ou *tour*), le choix du voisin particulier du type choisi par l'agent appartient à l'adversaire, puisque l'agent ne peut pas différencier deux voisins du même type.

Nous représentons ces possibilités différentes en utilisant le concept d'un adversaire qui planifie les actions des agents et choisit un voisin parmi les nombreux voisins du même type, afin que cela soit le plus nuisible possible à l'algorithme. Si nous prouvons que, pour certaines configurations initiales et pour n'importe quel algorithme hypothétique, l'adversaire peut faire des choix afin d'éviter le regroupement, il s'ensuivra que la configuration initiale donnée n'est pas regroupable, parce que, pour tout algorithme hypothétique à partir de cette configuration, certaines exécutions possibles de celui-ci empêchent le regroupement.

Une étoile est une configuration dans laquelle il y a un singleton A et tous les autres agents sont des singletons adjacent à A.

Nous considérons deux noeuds comme *équivalents* s'ils ont exactement les mêmes noeuds adjacents. Deux agents sont *similaires* s'il sont deux singletons, ou encore chacun appartient à une tour (la même ou deux tours séparées), et si le nombre de noeuds voisins vides, de singletons et de tours sont égaux. Il en résulte que ces agents similaires ont les mêmes entrées obtenues durant l'opération d'observation. Notons que les singletons et les agents localisés sur des tours, étant situés sur des noeuds équivalents, sont respectivement similaires.

Dans le reste du chapitre, nous assumons que le graphe sous-jacent est biparti régulier de degré δ. Nous allons utiliser les propositions suivantes en

omettant leurs preuves faciles.

Fait 3.3.1 *Si les noeuds a, b, c, d composent un chemin et a n'est pas adjacent à d, alors a, b, c, d sont des paires disjointes non-équivalentes.*

Fait 3.3.2 *Si les noeuds a et b ne sont pas adjacents et que chaque noeud adjacent à a est adjacent à b, alors a et b sont équivalents.*

Fait 3.3.3 *S'il existe un noeud dont tous les voisins sont équivalents, alors le graphe est biparti régulier complet.*

3.4 Le résultat d'impossibilité

Dans cette section, nous présentons notre principale contribution de ce chapitre qui est le résultat négatif suivant. Comme nous l'avons mentionné préalablement, ce résultat tient autant lorsque les agents peuvent détecter les multiplicités uniquement sur le noeud occupé que quand ils les perçoivent sur les voisins.

Théorème 3.4.1 *Une configuration qui n'est pas une étoile de grandeur au moins trois n'est pas regroupable.*

La preuve du théorème est divisée en une série de lemmes qui visent à exclure plusieurs classes de configurations de l'ensemble des configurations regroupables. Ils atteignent leur apogée en excluant toutes les configurations autres que les étoiles de grandeur au moins trois. L'idée générale utilisée afin de prouver qu'une configuration avec les propriétés données n'est pas regroupable est de considérer un algorithme hypothétique de rendez-vous et montrer que l'adversaire peut planifier les mouvements des agents d'une telle façon que la configuration résultante permute indéfiniment entre deux ou plusieurs types de configurations, chacune d'entre elles ayant au moins deux noeuds occupés.

L'objectif des trois premiers lemmes est de montrer que les configurations où tous les agents sont localisés dans plus d'une tour sont non-regroupables.

Lemme 3.4.1 *Un agent localisé dans une tour sans aucun agent adjacent ne peut pas bouger.*

Preuve: Si tous les agents sont dans la même tour, aucun d'entre eux ne peut bouger, car autrement le rendez-vous n'arriverait jamais. Dans n'importe quelle autre configuration composée d'une tour qui n'a pas d'agents adjacents, les agents dans la tour obtiennent la même entrée que dans une tour unique et, par conséquent, ne peuvent pas bouger. □

Lemme 3.4.2 *Une configuration qui est composée exactement de deux tours adjacentes n'est pas regroupable.*

Preuve: Tous les agents sont similaires et, par conséquent, peuvent soit bouger sur un voisin vide soit sur un voisin occupé par une tour. Dans le premier cas, l'adversaire bouge en premier lieu tous les agents simultanément de la tour vers un voisin vide créant ainsi deux tours non-adjacentes qui ne peuvent pas bouger selon le lemme 3.4.1. Dans le second cas, l'adversaire échange simultanément les tours de place reproduisant ainsi une configuration similaire. Dans les deux cas, le rendez-vous est empêché. □

Lemme 3.4.3 *Une configuration dont tous les agents sont localisés dans au moins deux tours n'est pas regroupable.*

Preuve: L'adversaire déplacera simultanément une tour à la fois sur un voisin vide ou sur un voisin occupé par une tour. Si de tels mouvements pouvaient accomplir le rendez-vous, l'adversaire pourrait planifier les mouvements de façon qu'à un certain moment, il n'y ait que deux tours adjacentes. Le rendez-vous est alors impossible selon le lemme 3.4.2. □

Le prochain lemme donne une condition nécessaire pour qu'un graphe biparti complet régulier soit regroupable.

Lemme 3.4.4 *Considérons une configuration dans un graphe biparti régulier complet avec la bipartition X, Y, consistant en 2 ensembles de singletons tels que les singletons dans chaque ensemble sont similaires. Cette configuration n'est pas regroupable sauf si un des ensembles est de grandeur 1 et l'autre est de grandeur d'au moins 2.*

Preuve: Puisque le graphe est biparti régulier complet, les ensembles X et Y sont de grandeur identique. Chaque ensemble d'agents similaires est inclus dans X ou dans Y. Supposons qu'il est faux de dire qu'un des ensembles de singletons est de grandeur 1 et l'autre est de grandeur 2. Il y a alors 3 cas possibles :

Cas 1. Un des ensembles est vide.
L'adversaire bouge tous les agents de l'autre ensemble sur des voisins différents. La configuration est analogue à la précédente et le rendez-vous est empêché.

Cas 2. Chacun des ensembles contient exactement un seul agent.
L'adversaire change de place les deux agents ou encore les bouge sur des noeuds voisins simultanément. La configuration est analogue à la précédente et le rendez-vous est empêché.

Cas 3. Chacun des ensembles contient au moins 2 agents.
Afin que le rendez-vous soit possible, les agents d'un des deux ensembles doivent bouger. Si les agents des deux ensembles bougent, l'adversaire fera en sorte que les singletons d'un des ensembles bougent avant l'autre. Appelons cet ensemble A. Il y a alors 2 sous-cas.

Sous-cas 3.1 Les agents de l'ensemble A se déplacent sur un voisin vide.
L'adversaire bougera tous les agents vers le même voisin vide résultant en une configuration qui contient une tour et au moins 2 singletons à distance 2 de cette tour. Selon le lemme 3.4.1, les agents sur la tour ne peuvent pas bouger. Il en résulte qu'afin que le rendez-vous soit possible, les singletons doivent bouger. L'adversaire les bougera tous vers un voisin vide créant ainsi une configuration avec tous les agents sur 2 tours adjacentes. Cette configuration n'est pas regroupable selon le lemme 3.4.2.

Sous-cas 3.2 Les agents de l'ensemble A se déplacent sur un singleton voisin.
S'il y a plus de 2 singletons dans l'autre ensemble, l'adversaire bouge tous les agents de A vers le même singleton adjacent et la configuration résultante

aura une tour et au moins 2 singletons à distance 2 de cette tour. Le reste de l'argument est comme dans le cas 3.1. S'il y a exactement 2 singletons dans l'autre ensemble, l'adversaire bouge tous les singletons de A sur ces 2 singletons créant ainsi une configuration dans laquelle tous les agents sont sur exactement 2 tours. Selon le lemme 3.4.3, cette configuration n'est pas regroupable. □

Le prochain lemme montre ce qui doit arriver lorsqu'une paire de noeuds adjacents contenant un singleton et une tour est isolée des autres agents.

Lemme 3.4.5 *Dans une configuration avec un singleton adjacent à une tour et aucun autre agent adjacent à l'un d'eux, la tour ne peut pas se déplacer et le singleton doit bouger sur la tour.*

Preuve: Si le rendez-vous peut arriver, l'adversaire peut planifier les mouvements des agents de telle façon qu'avant le dernier mouvement, il y a exactement une tour et un unique singleton adjacent à cette tour. Dans une telle configuration, soit la tour doit bouger sur le singleton ou vice-versa. Dans le premier cas, l'adversaire bouge tous les agents sauf un de la tour vers le singleton, créant une configuration identique à la précédente et le rendez-vous ne se produit pas. Il en résulte que le singleton doit bouger sur la tour. S'il y a d'autres agents dans la configuration qui ne sont ni adjacents à la tour, ni adjacents au singleton, la tour et le singleton perçoivent les mêmes entrées que dans la situation considérée précédemment et, par conséquent, leur comportement est le même. Donc, la tour ne peut pas bouger et le singleton doit bouger sur la tour. □

Une configuration est dite *stable* si elle contient une tour et un singleton ainsi qu'un voisin de ce singleton qui sont tous deux non-adjacents à la tour. Une configuration est dite *multitours* si elle contient plus d'une seule tour.

Les lemmes qui suivent montrent comment une configuration avec une tour de dimension importante évolue.

Lemme 3.4.6 *Considérons une configuration avec exactement une tour contenant au moins 4 agents. L'adversaire peut alors bouger les agents d'une telle façon que la configuration demeure stable ou devient multitours.*

Preuve: Si la tour peut se déplacer, l'adversaire peut créer une configuration multitours. Si la tour ne peut pas bouger, alors le singleton A qui rend la configuration stable devra bouger éventuellement afin de permettre le rendez-vous. Puisque A n'est pas un voisin de la tour, il doit bouger vers un noeud vide ou vers un autre singleton. S'il bouge sur un noeud vide alors, puisque A rend la configuration stable, cela veut dire qu'il existe un noeud qui n'est pas adjacent à la tour et qui est adjacent à A. Si ce dernier noeud est vide, alors l'adversaire bougera A sur ce noeud gardant ainsi la configuration stable. Si ce noeud est occupé par un singleton alors, peu importe sur quel noeud vide bougera A, la configuration demeurera stable. Finalement, si A bouge sur un autre singleton, alors la configuration passera à l'état multitours. □

Une configuration C au temps t est dite *perpétuelle* si elle est stable ou multitours au temps t et si l'adversaire peut planifier le mouvement des agents d'une telle façon que pour chaque temps $t' > t$, il existe un temps $t'' > t'$ où la configuration sera à nouveau stable ou perpétuelle. Notons qu'une configuration perpétuelle n'est pas regroupable.

Les deux prochains lemmes montrent comment une configuration composée de deux tours adjacentes évolue.

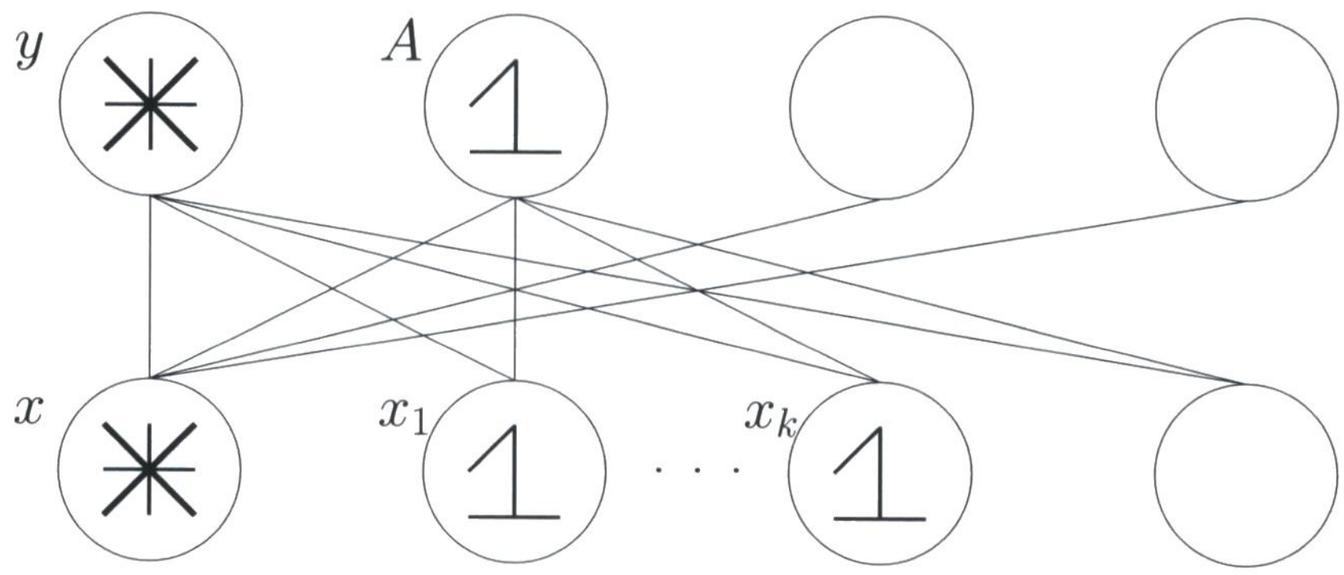

FIGURE 3.4 – La configuration avec les propriétés des lemmes 3.4.7, 3.4.8

Lemme 3.4.7 *Considérons une configuration contenant exactement 2 tours situées sur des noeuds adjacents x et y. Alors, soit la configuration est perpétuelle, soit l'adversaire peut bouger les agents d'une telle façon que la configuration doit éventuellement avoir les propriétés suivantes : (voir Fig. 3.4) :*

1. il y a exactement 2 tours adjacentes et tous les singletons sont adjacents à l'une d'elles,
2. le noeud x a exactement un singleton adjacent A,
3. le noeud contenant A est équivalent à y,
4. le noeud y a au moins deux singletons adjacents.

Preuve: Nous prouvons tout d'abord que tous les singletons sont adjacents à l'une ou l'autre des tours car sinon, la configuration sera stable après le mouvement d'une tour sur l'autre tour. Supposons qu'il existe un singleton B qui n'est pas adjacent à l'une ou l'autre des tours et que la configuration ne devient pas stable après le mouvement d'une tour sur l'autre. Alors, il existe un noeud adjacent à B qui n'est pas voisin de x et un noeud adjacent à B qui n'est pas voisin de y. Supposons, sans perte de généralité, que les agents de y bougent sur x. Alors la configuration devient stable puisque B n'est pas voisin de x et qu'il a un voisin qui n'est pas voisin de x. Contradiction (voir Fig. 3.5).

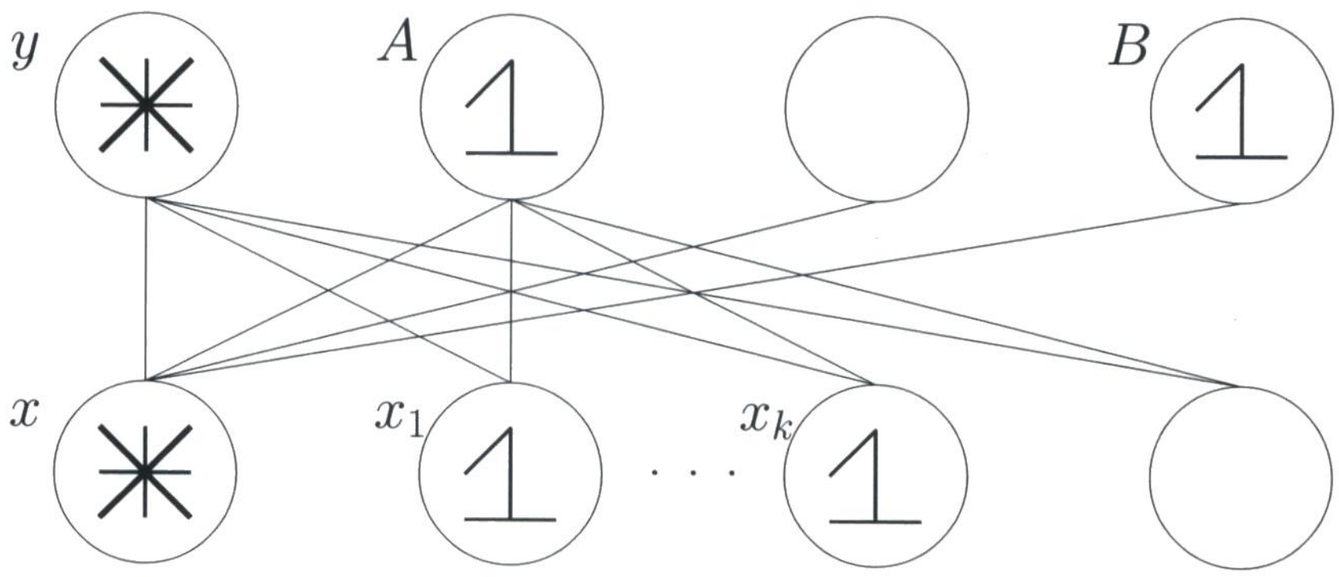

FIGURE 3.5 – Le singleton B n'est pas adjacent à une tour.

Supposons maintenant que la configuration n'est pas perpétuelle. Nous allons prouver que l'adversaire peut garantir la propriété 1. Une tour doit bouger sur l'autre afin de créer une configuration qui n'est pas multitours. Si au moins un singleton n'est pas adjacent à une tour, la configuration est stable après le mouvement. Selon le lemme 3.4.6, la configuration est perpétuelle ou deviendra multitours à nouveau. Il en résulte que la propriété 1 sera éventuellement garantie.

Affirmation 3.4.1 *Si les noeuds occupés par les singletons adjacents à y ne sont pas équivalents à x et que les noeuds occupés par les singletons adjacents à x ne sont pas équivalents à y, alors la configuration sera stable après le mouvement d'une tour sur l'autre.*

Premièrement, l'adversaire va bouger tous les singletons qui peuvent bouger sur leur tour voisine respective. Ensuite, une tour doit bouger sur l'autre. Si les agents dans les deux tours sont similaires, alors l'adversaire peut changer les deux tours de place en gardant la configuration identique (voir Fig. 3.6).

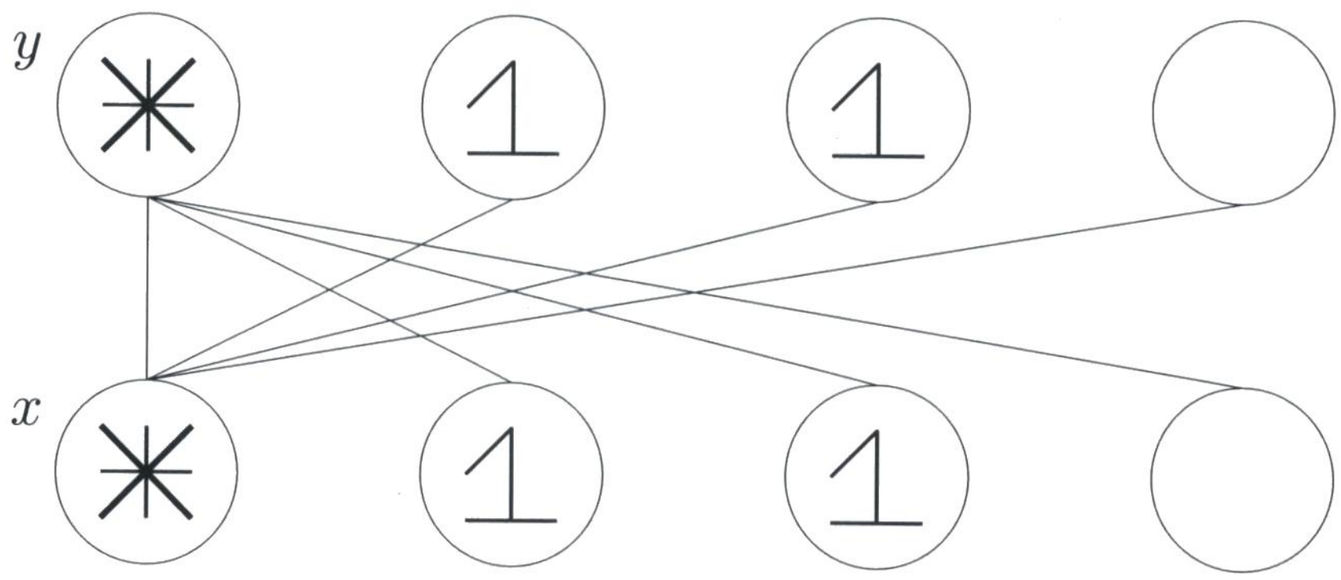

FIGURE 3.6 – Les tours sont similaires.

Donc, nous pouvons supposer que chaque tour a un nombre de singletons adjacents différents. Supposons sans perte de généralité que x a au moins un singleton adjacent A. Si y n'avait pas de singleton adjacent, tous les singletons seraient adjacents à x et selon le lemme 3.4.5, ils devraient bouger sur x passant la configuration dans l'état perpétuel (voir Fig. 3.7).

Il s'ensuit que chaque tour a au moins un singleton adjacent. Après le mouvement d'une tour sur l'autre, n'importe quel noeud contenant un singleton à distance 2 de la tour résultante rendra la configuration stable selon le fait 3.3.2, étant donné que ce noeud n'est pas équivalent au noeud contenant la tour. Cette démonstration prouve l'affirmation.

Nous prouverons maintenant que l'adversaire peut garantir la propriété 3. Si la configuration est stable après le mouvement d'une tour sur l'autre, le

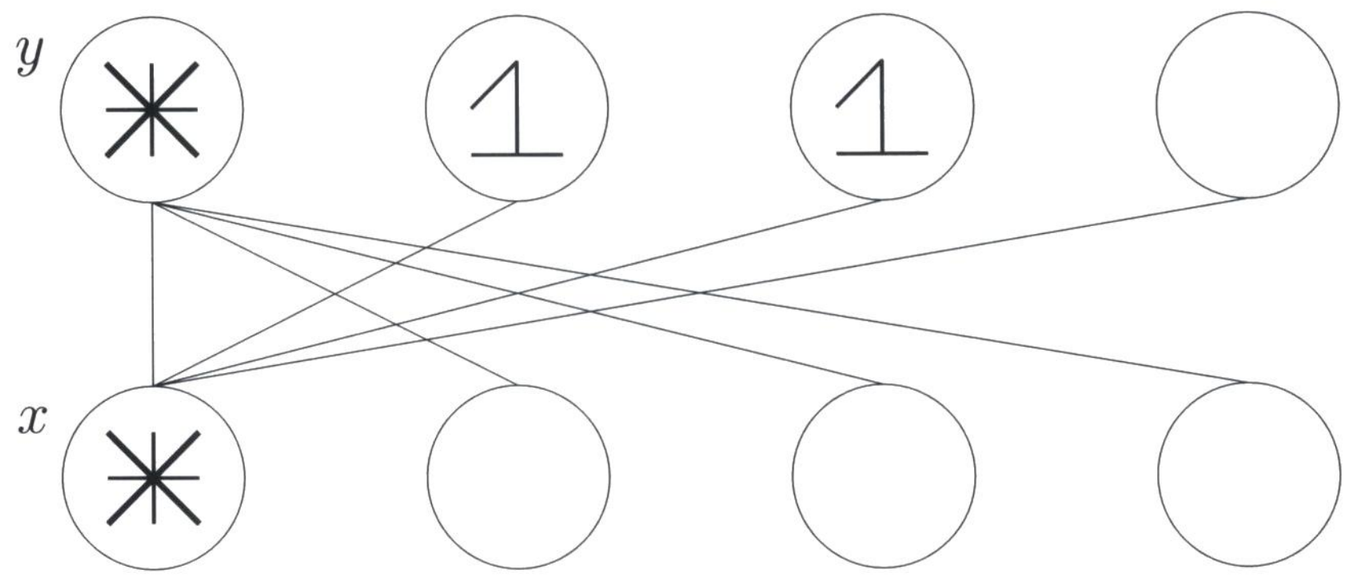

FIGURE 3.7 – x a au moins un singleton adjacent et y n'en a pas.

lemme 3.4.6 implique que la configuration va être perpétuelle ou va redevenir multitours à nouveau. Par conséquent, en vue de l'affirmation, nous pouvons assumer sans perte de généralité que le noeud contenant A est équivalent à y.

Nous prouvons finalement que l'adversaire peut garantir les propriétés 2 et 4. Comme nous l'avons observé, nous pouvons supposer que chaque noeud x et y a au moins un singleton adjacent et que les nombres de singletons adjacents doivent différer. Nous avons 2 cas :

Cas 1. A est le seul singleton adjacent à x (voir Fig. 3.8).
Il s'ensuit que y doit avoir au moins 2 singletons adjacents.

Cas 2. Il existe au moins 2 singletons A et A' adjacents à x (voir Fig. 3.9).
Puisque la configuration ne peut pas être stable après le mouvement d'une tour vers l'autre, la tour sur x doit bouger sur y. De plus, y et les noeuds contenant A et A' doivent être équivalents. Puisque, par l'hypothèse, la configuration n'est pas perpétuelle, peu importe les mouvements que feront les singletons adjacents à y, la tour ou encore A et A' devront se déplacer. Si la tour se déplace, l'adversaire peut faire en sorte que la configuration devienne multitours. Si A se déplace, alors l'adversaire peut également bouger A' puisque les noeuds sur lesquels ils sont localisés sont équivalents. Il en

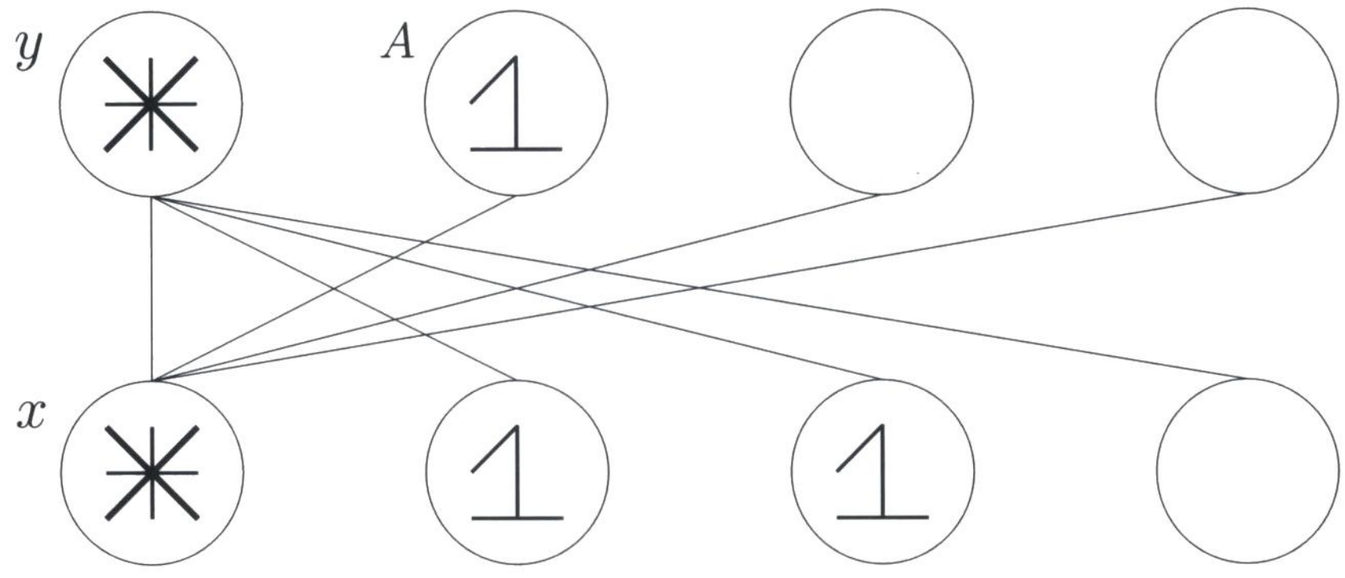

FIGURE 3.8 – A est le seul singleton adjacent à x.

résulte qu'en déplaçant A et A' simultanément, l'adversaire peut créer une configuration multitours. Puisque la configuration n'est pas perpétuelle, l'adversaire peut éventuellement créer une configuration avec exactement deux tours sur des noeuds adjacents, comme dans la condition initiale du lemme.

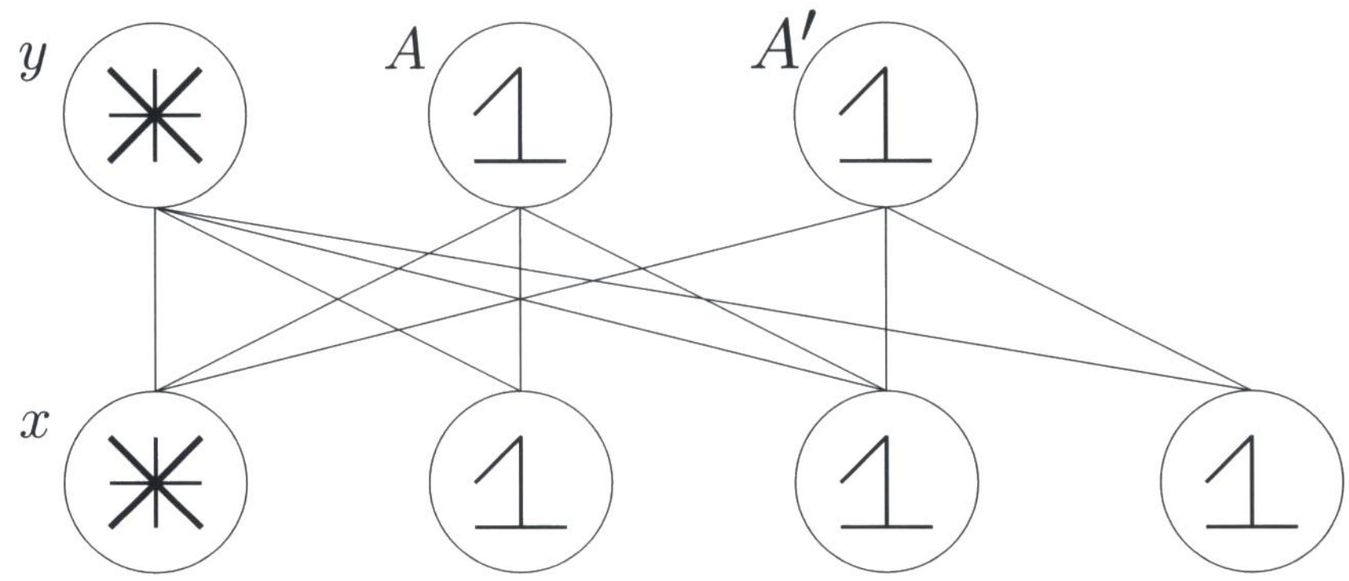

FIGURE 3.9 – Il existe au moins 2 singletons A et A' adjacents à x.

□

Pour n'importe quel algorithme de rendez-vous $\mathcal{A}$, un *parent* d'une configuration C est n'importe quelle configuration C', tel qu'il existe un mouvement d'un agent dans C' en suivant l'algorithme $\mathcal{A}$, qui résulte dans la configuration C. Le parent d'un parent de la configuration C est appelé $grand-parent$ de C. Une *branche ancestrale* d'une configuration C est une

séquence de configurations $(C = C_0, C_1, \ldots, C_s)$ tel que C_{i+1} est un parent de C_i, et C_s est une configuration initiale.

Lemme 3.4.8 *Considérons une configuration C avec exactement 2 tours situées sur des noeuds adjacents x et y. Si la configuration a les propriétés suivantes :*
1. il y a exactement 2 tours adjacentes et tous les singletons sont adjacents à l'une d'elles (voir Fig. 3.4),
2. le noeud x a exactement un singleton adjacent A,
3. le noeud contenant A est équivalent à y,
4. le noeud y a au moins deux singletons adjacents,
alors pour chaque branche ancestrale de C, il existe une configuration perpétuelle.

Preuve: Premièrement, l'adversaire bouge tous les singletons qui peuvent bouger sur leur tour adjacente. Supposons que la configuration C n'est pas perpétuelle. Il s'ensuit que, selon n'importe quel algorithme hypothétique, une tour doit bouger sur l'autre tour. Supposons que la tour sur y bouge sur x. Les seuls agents qui ont une vue différente après le mouvement de la tour sont les singletons adjacents à y et les agents formant la tour résultante. Supposons premièrement que les agents sur la tour bougent. Puisqu'il y a au moins 4 agents sur la tour, l'adversaire peut faire en sorte de recréer une configuration multitours. Nous pouvons donc assumer que les singletons adjacents à y sont les seuls à pouvoir bouger. Toutefois, si tous les singletons adjacents à y ne sont pas similaires, cela implique que la configuration est stable et alors le lemme 3.4.6 implique que la configuration sera perpétuelle, ce qui est une contradiction. Nous pouvons donc supposer que les singletons adjacents à y sont similaires. Notons que si les noeuds contenant les singletons adjacents à y sont identiques, il s'ensuit selon le fait 3.3.3 que le graphe est biparti régulier complet. Par conséquent, dans n'importe quelle branche ancestrale de C, il existe une configuration C', tel que l'adversaire pourrait bouger tous les agents dans C' afin de former exactement 2 tours, ce qui implique que C' est perpétuelle. Nous pouvons donc supposer que les singletons adjacents à y sont similaires mais que les noeuds les contenant ne sont pas équivalents.

Puisqu'il y a au moins 2 singletons adjacents à y et qu'ils ne sont pas adjacents à la tour résultante sur x après le mouvement de la tour sur y

vers x, l'adversaire va bouger ces singletons sur le même noeud et recréer à nouveau une configuration multitours (peu importe si les singletons bougent sur un noeud vide ou occupé par un agent) (voir Fig. 3.10).

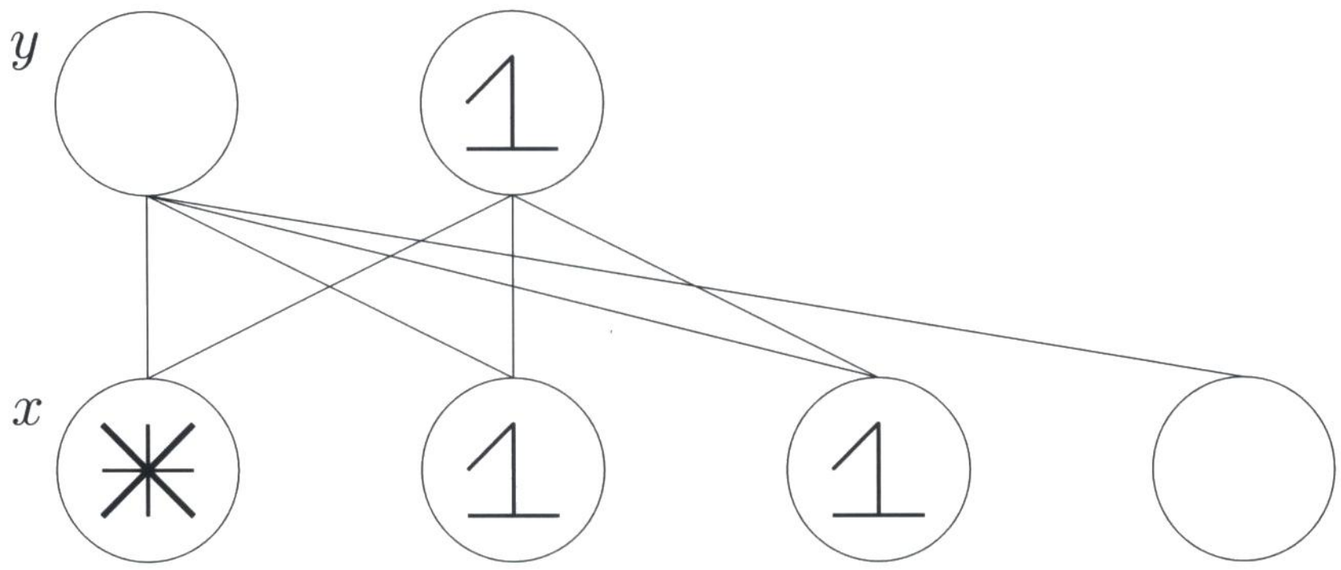

FIGURE 3.10 – La configuration après le mouvement de la tour de y vers x.

Nous avons donc une contradiction qui nous permet de supposer que la tour de x bougera sur y (voir Fig. 3.11).

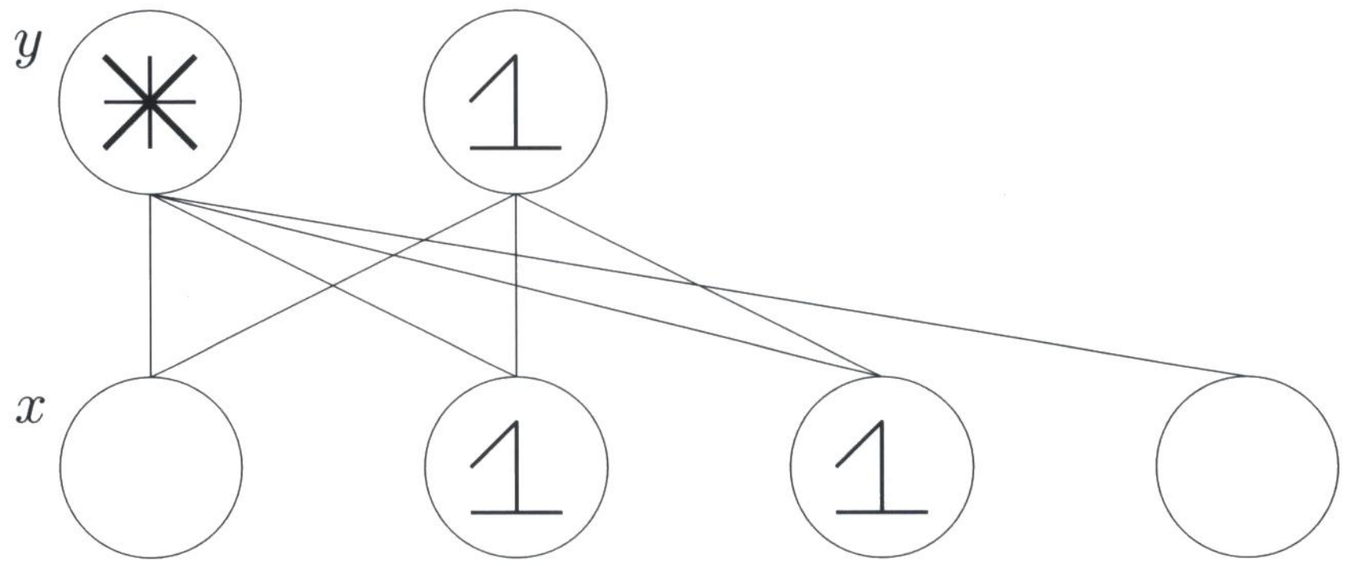

FIGURE 3.11 – La configuration après le mouvement de la tour de x vers y.

Supposons que la tour sur x a k agents. Alors l'adversaire peut bouger les agents de x vers y de deux façons différentes afin de changer la configuration : il peut bouger k agents, ce qui veut dire que x sera vide, ou il peut bouger $k-1$ agents, ce qui veut dire que x sera occupé par un singleton. Supposons qu'il y a $m \geq 2$ singletons parmi les noeuds adjacents à y. Ces singletons ne peuvent

pas bouger puisqu'ils reçoivent les mêmes entrées qu'avant le mouvement de la tour de x vers y. Le seul agent qui peut bouger après le mouvement de la tour, sans rendre la configuration multitours, est A, et afin de le faire, A doit se déplacer sur un noeud vide lorsqu'il y a m ou $m+1$ noeuds adjacents avec des singletons. Si un tel noeud n'existe pas, cela implique que l'adversaire aurait pu déplacer $k-1$ agents de x vers y et alors A ne pourrait pas se déplacer sur un noeud vide. Il en résulte qu'aucun agent ne pourrait bouger et le rendez-vous serait impossible.

Puisque après le mouvement de la tour la configuration a seulement les agents suivants : les agents sur y, le singleton A et les singletons adjacents à y, nous devons observer quel agent parmi eux a bougé afin de produire la configuration C. Considérons le temps t lorsque la configuration est égale à C et le dernier temps $t' < t$, lorsque l'agent a bougé. Considérons chaque cas possible pour ce dernier mouvement.

Cas 1. Le singleton B a bougé sur y.
Alors B est venu d'un noeud adjacent à y. Toutefois, puisque tous les singletons adjacents à y sont similaires, il s'ensuit que l'adversaire aurait pu déplacer ces singletons simultanément sur y, contrairement aux propriétés de la configuration au temps t. Il en résulte selon le lemme 3.4.7 que, puisque la configuration au temps t est multitours et que l'adversaire peut agir de telle façon que la configuration n'aura plus les 4 propriétés, le parent de C au temps t' est perpétuel (voir Fig. 3.12).

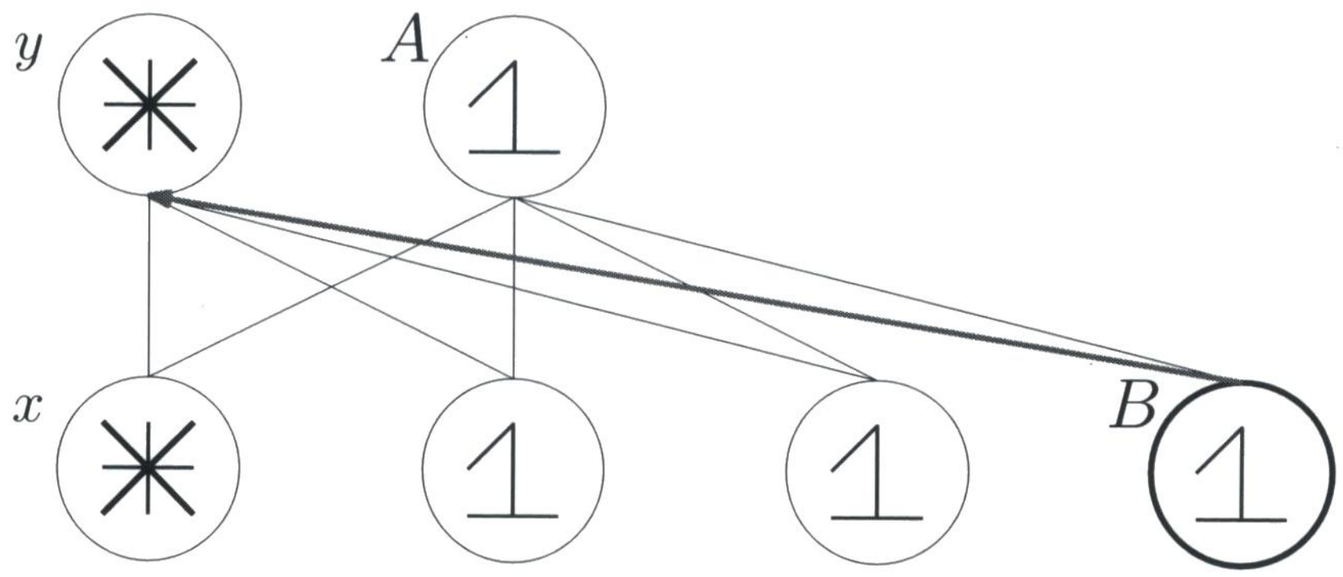

FIGURE 3.12 – Le singleton B a bougé sur y.

Cas 2. Le singleton B a bougé sur un noeud vide adjacent à y.
Il s'ensuit que les noeuds contenant A et B ne peuvent pas être équivalents ou autrement, l'adversaire aurait bougé A et B sur le même noeud résultant dans la création d'une troisième tour chez un voisin de y. Ensuite, l'adversaire aurait déplacé tous les singletons de y et la configuration ainsi produite aurait été composée de tous les agents sur les tours, résultant dans une configuration perpétuelle. Cela implique selon le fait 3.3.2 que B a un voisin qui n'est pas adjacent à x ou y. L'adversaire aurait alors pu déplacer B sur ce noeud au lieu d'un noeud adjacent à y, créant ainsi une configuration stable contrairement aux propriétés de la configuration au temps t. Il en résulte que le parent de C au temps t' est perpétuel (voir Fig. 3.13).

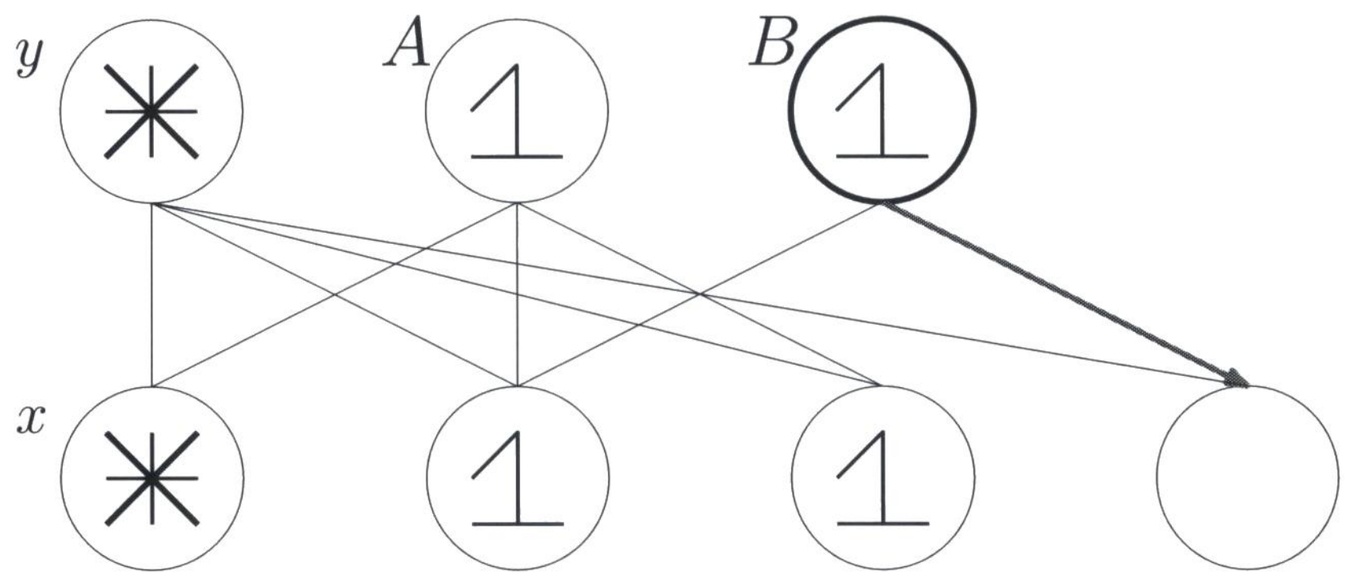

FIGURE 3.13 – Le singleton B a bougé sur un noeud vide adjacent à y.

Cas 3. Le singleton A a bougé.
Puisque A a bougé au temps t, cela implique qu'il était adjacent à y avant le mouvement. Il s'ensuit que A et les singletons adjacents à y étaient similaires et alors, l'adversaire aurait pu les déplacer sur la position actuelle de A, résultant dans la création d'une troisième tour au temps t, avec tous les agents sur les tours. Il s'ensuit que le parent de C au temps t' est perpétuel (voir Fig. 3.14).

Cas 4. Le singleton B a bougé sur x.
Le noeud x contient $1 \leq k-1$ agents avant que B se déplace. Supposons que $k = 2$. Cela implique qu'au temps t', A pourrait se déplacer sur un noeud vide (A n'a pas de tour adjacente et A a exactement $m+1$ singletons adjacents). Alors l'adversaire déplacerait A sur un noeud vide en même temps que B se

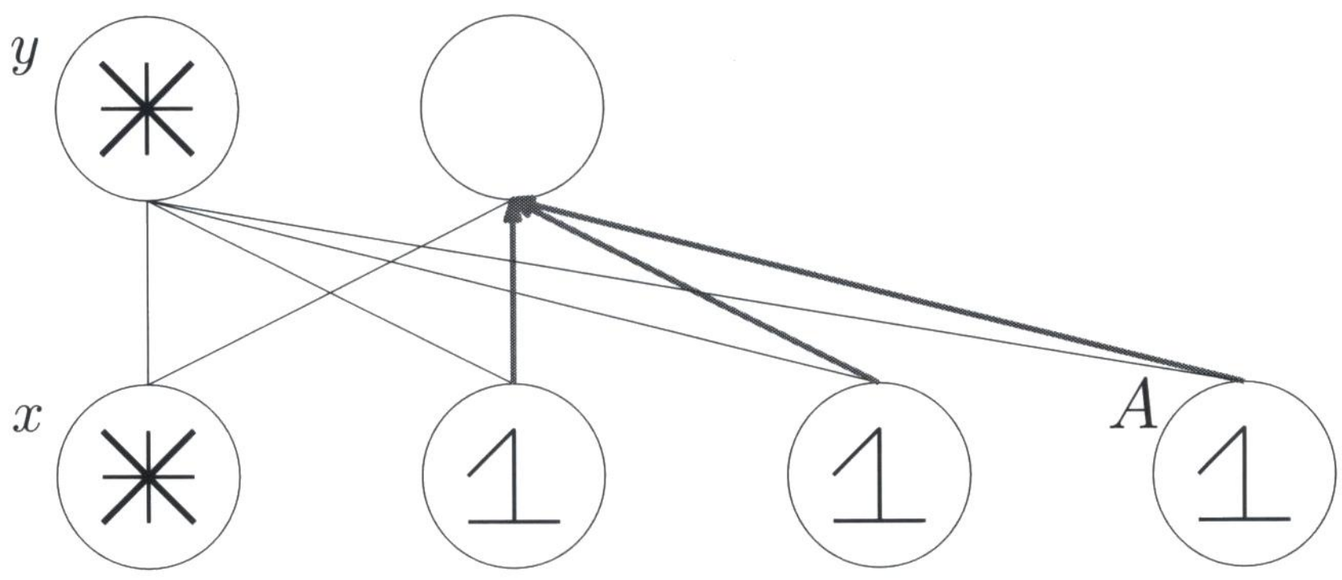

FIGURE 3.14 – Le singleton A a bougé.

déplacerait sur x. Seulement, y aurait des singletons adjacents au temps t, puisque A se serait déplacé. Il s'ensuit que tous les singletons se déplaceraient alors sur y et la configuration serait perpétuelle (voir Fig. 3.15).

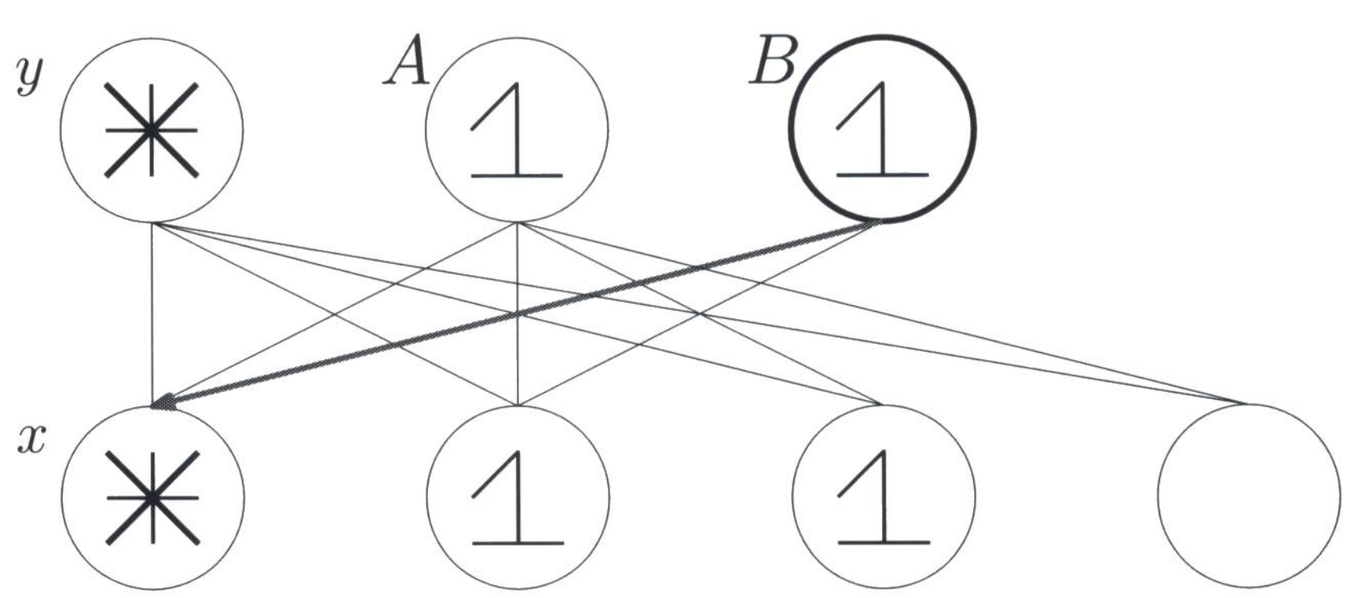

FIGURE 3.15 – Le singleton B a bougé sur x.

Nous pouvons donc supposer que $k > 2$. Les noeuds contenant A et B au temps t' n'étaient pas équivalents, ou autrement l'adversaire aurait bougé les deux agents sur x rendant la configuration perpétuelle. Il en résulte que B a un voisin qui n'est pas voisin de y. L'agent adjacent a y qui voit B peut se déplacer. Toutefois, avant que B bouge sur le noeud v adjacent à x et après que B se soit déplacé de v vers x au temps t, les singletons adjacents à y ne pouvaient pas se déplacer. Il s'ensuit que l'adversaire peut ordonner l'arrivée de B sur v et son départ de v vers x avant que les singletons adjacents à

y voient le singleton B sur v. Il en résulte que nous n'avons seulement qu'à considérer le temps $t'' < t'$ avant que B se déplace sur v. (La configuration C' au temps t'' est un grand-parent de C.)

Cas 4.1. Le singleton B est adjacent à y au temps t''.
Alors, B et tous les m autres singletons adjacents à y sont similaires et l'adversaire pourrait déplacer tous ces singletons sur v. Il en résulte qu'une troisième tour serait créée et que l'ancêtre C' de C est perpétuel.

Cas 4.2. Le singleton B n'est pas adjacent à y au temps t''.
Alors B n'est adjacent à aucun agent. Tous les autres agents ont la même entrée au temps t qu'au temps t'' et alors, l'adversaire aurait pu bouger les agents de x vers y au temps t'' résultant par une configuration stable. Il en résulte que l'ancêtre C' de C est perpétuel. □

Le lemme qui suit est crucial pour les considérations futures. Il implique que si une configuration initiale évolue pour devenir soit stable soit multitours, alors cette configuration initiale n'est pas regroupable.

Lemme 3.4.9 *Soit C, une configuration qui est stable ou multitours. Dans chaque branche ancestrale de C, il existe une configuration C' tel que l'adversaire peut planifier les mouvements des agents d'une telle manière que C' devient perpétuelle.*

Preuve: Supposons premièrement que la configuration C est multitours. Si C n'est pas perpétuelle, alors après certains déplacements des agents, elle ne peut plus être multitours. Par conséquent, certains agents doivent se déplacer. Si certains singletons se déplacent, le nombre de tours ne diminue pas et la configuration demeure multitours. Si les agents dans une tour se déplacent, alors s'il y a plus de deux tours dans C, la configuration demeure tout de même multitours. S'il y a exactement deux tours dans C, alors afin de permettre de créer une configuration qui n'est plus multitours, les tours doivent tout d'abord être adjacentes et alors dans chaque branche ancestrale de C, il existe une configuration perpétuelle, selon les lemmes 3.4.7 et 3.4.8.

Supposons que C est stable. Si C est également multitours, les arguments ci-haut mentionnés s'appliquent. Nous pouvons donc supposer que la configuration a une unique tour située sur un noeud v et un singleton qui n'est

pas adjacent à v, qui a un voisin w qui n'est pas adjacent à v également. Pour que la configuration ne soit plus stable, A ou la tour doivent bouger.

Cas 1. Le singleton A se déplace.
Le singleton A peut bouger sur un noeud vide ou sur un singleton. Si A bouge sur un singleton, alors une nouvelle tour est créée et la configuration devient multitours. Si A bouge sur un noeud vide, l'adversaire le déplacera sur w si w est vide. Dans ce cas, la configuration demeure stable. Si w est occupé par un singleton, peu importe sur quel noeud vide A se déplacera, la configuration demeurera stable.

Cas 2. Les agents dans la tour se déplacent.
Puisque le singleton A et le noeud v ne sont pas adjacents et que la configuration est stable, il existe un noeud v' adjacent à v qui n'est pas adjacent à A. Les agents dans la tour peuvent bouger sur un singleton ou sur un noeud vide.

Cas 2.1. Les agents dans la tour se déplacent sur un singleton.
S'il y avait exactement un seul singleton adjacent, le lemme 3.4.5 implique que la tour ne pourrait pas se déplacer. Nous pouvons donc assumer qu'il y a au moins 2 singletons adjacents à la tour. L'adversaire peut déplacer des agents de la tour vers ces 2 singletons créant ainsi une configuration multitours.

Cas 2.2. Les agents dans la tour se déplacent sur un noeud vide.
Si A et la tour sont à distance supérieure à 2, alors après le mouvement de la tour, la configuration demeure stable. Nous pouvons donc supposer que la tour et le singleton A sont à distance 2 exactement. Dans ce cas, si v' est vide, alors la tour se déplacera sur v' et la configuration restera stable. Si v' est occupé par un singleton, alors si la tour se déplace sur un noeud z adjacent à A, il en résultera que la configuration demeurera stable puisqu'il existe un noeud adjacent à v' qui n'est pas adjacent à z. □

Une configuration est *connexe* si le sous-graphe induit par les noeuds occupés est connexe. Une configuration est *linéaire* si elle contient 4 noeuds occupés a, b, c, d tel que b est adjacent à a et c, et que d est adjacent à c mais pas à a.

Lemme 3.4.10 *Si une configuration non-connexe qui n'a pas 3 agents est obtenue à partir d'une configuration initiale regroupable, alors l'adversaire peut bouger les agents d'une telle manière que la configuration demeure non-connexe ou devient linéaire.*

Preuve: Selon le lemme 3.4.4, le graphe ne peut pas être complet biparti parce que tous les agents seraient isolés et le rendez-vous serait alors impossible. Si la configuration contient une tour, il y a alors trois possibilités : soit elle est stable, soit elle est multitours, ou soit chaque singleton est sur des noeuds équivalents à celui contenant la tour. Dans les deux premiers cas, le rendez-vous est impossible selon le lemme 3.4.9. Dans le dernier cas, il doit y avoir un unique singleton, car autrement, l'adversaire pourrait créer une autre tour. Puisque la configuration n'est pas stable, les noeuds contenant la tour et les singletons sont équivalents. S'il y a plus de 2 agents dans la tour, l'adversaire aurait pu déplacer les agents afin de créer une configuration multitours. Donc, il y a exactement 3 agents : 2 dans la tour et un singleton. Contradiction.

Nous pouvons donc supposer que la configuration ne contient aucune tour. À un certain point, 2 composantes connexes de la configuration doivent être à distance 2, autrement le rendez-vous est impossible. Considérons 2 singletons A et A' a distance 2, chacun faisant partie d'une composante connexe différente. Sans perte de généralité, un de ces agents doit se déplacer, afin d'accomplir le rendez-vous. Si A et A' se déplacent sur un singleton, alors la configuration devient stable et le rendez-vous est impossible. Par conséquent, au moins un des agents A ou A' doit se déplacer sur un noeud vide.

Si A et A' sont similaires, il y a deux possibilités. Si tous les singletons sont localisés sur des noeuds équivalents, alors l'adversaire bougera tous les singletons simultanément sur des noeuds vides différents et les déplacera à nouveau sur leurs points de départ initiaux respectifs. La configuration restera non-connexe. S'il y a des noeuds non-équivalents contenant des singletons, alors il y a deux cas.

Cas 1. Il y a un noeud adjacent à tous les singletons.
Il y a au plus δ singletons et l'adversaire peut déplacer tous les singletons simultanément sur des noeuds différents vides et les ramener au point de

départ par la suite. La configuration demeure non-connexe.

Cas 2. A et A' sont adjacents à un noeud x et il y a un singleton A'' qui n'est pas adjacent à x
Puisque A'' ne peut pas être un voisin de A et A' en même temps (autrement, A et A' seraient dans la même composante connexe), A'' a un voisin qui n'est pas adjacent à x. L'adversaire peut bouger A et A' sur x et la configuration devient stable. Selon le lemme 3.4.9, le rendez-vous est impossible.

Si A et A' ne sont pas similaires, alors soit A ou A' a un singleton adjacent. Afin de former éventuellement une configuration connexe, soit A ou A' doit se déplacer sur le noeud vide x. Sans perte de généralité, assumons que A' a un singleton adjacent. Il y a trois cas.

Cas 1. A n'a pas de singleton adjacent et il peut bouger sur un noeud vide.
L'adversaire déplacera A sur un noeud qui n'est pas adjacent à A'. Après ce déplacement, soit la configuration reste non-connexe ou A a un singleton adjacent. Si la configuration est maintenant connexe et que A a un singleton adjacent, alors la configuration a un diamètre de 3 et devient linéaire.

Cas 2. A a un singleton adjacent et il peut bouger sur un noeud vide.
Alors le singleton adjacent à A n'est pas adjacent à A'. Suite au déplacement de A sur x, la configuration sera non-connexe ou linéaire.

Cas 3. A' peut bouger sur un noeud vide.
Puisque A' a un singleton adjacent qui n'est pas adjacent à A, suite au déplacement de A' sur x, la configuration sera non-connexe ou linéaire. □

Le prochain lemme montre que le rendez-vous ne peut pas être accompli en passant par une configuration linéaire.

Lemme 3.4.11 *Une configuration linéaire obtenue à partir d'une configuration initiale regroupable n'est pas regroupable.*

Preuve: Supposons tout d'abord que la configuration est non-connexe. Pour que le rendez-vous soit possible, la configuration devra éventuellement être connexe, et alors, selon le lemme 3.4.10 elle va devenir connexe ainsi que linéaire. Supposons que la configuration possède une tour sur le noeud v. Nous pouvons supposer que cette tour est unique, sinon le lemme 3.4.9 montre que la configuration n'est pas regroupable. Considérons une ligne a, b, c, d de noeuds occupés qui rendent la configuration linéaire. Si v n'est pas adjacent à aucun noeud de la ligne, la configuration est stable et n'est pas regroupable selon le lemme 3.4.9. Donc, nous pouvons supposer sans perte de généralité que v est adjacent à a ou c. Puisque b et d ne sont pas équivalents, v ne peut pas être équivalent aux deux. Donc v fait partie d'une ligne de noeuds occupés. Il en résulte qu'un des noeuds de cette ligne est à distance 2 de v et que ce noeud n'est pas équivalent à v. La configuration sera donc stable et elle n'est pas regroupable selon le lemme 3.4.9.

Dans le reste de la preuve, nous pouvons assumer que la configuration ne contient pas de tours. Supposons que les singletons A, B, C, D sont respectivement localisés sur les noeuds a, b, c, d de la ligne qui rend la configuration linéaire. Selon le fait 3.3.1, tous les noeuds contenant ces singletons ne sont pas équivalents. Afin d'accomplir le rendez-vous, certains de ces singletons doivent se déplacer. Le rôle de A et D ainsi que le rôle de B et C sont les mêmes, donc nous allons argumenter seulement pour A et B. Il y a 4 cas.

Cas 1. A se déplace sur un singleton.
L'adversaire déplace A sur B. Puisque b et d ne sont pas équivalents, la configuration devient stable et le rendez-vous est impossible.

Cas 2. A se déplace sur un noeud vide.
S'il existe un singleton adjacent à B qui n'est pas adjacent à D, alors la configuration reste linéaire. Supposons qu'un tel singleton n'existe pas. Considérons un noeud α adjacent à A qui n'est pas adjacent à C. Si α est vide, l'adversaire déplacera A sur α, autrement, il bouge A sur un autre noeud vide. Après ce mouvement, il y a deux possibilités : α et B sont dans la même composante connexe ou pas. Dans le premier cas, la configuration reste linéaire et dans le second cas, elle est non-connexe et alors, selon le lemme 3.4.10, elle restera non-connexe ou redeviendra à nouveau linéaire. Il

en résulte que le rendez-vous sera impossible.

Cas 3. B se déplace sur un singleton.
L'adversaire déplace B sur A. Puisque a et c ne sont pas équivalents, la configuration devient stable et le rendez-vous est impossible.

Cas 4. B se déplace sur un noeud vide.
S'il existe un singleton adjacent à A qui n'est pas adjacent à C, alors la configuration reste linéaire. Supposons qu'un tel singleton n'existe pas. Considérons un noeud β adjacent à B qui n'est pas adjacent à D. Si β est vide, l'adversaire déplacera B sur β, autrement, il bouge B sur un autre noeud vide. Après ce mouvement, il y a deux possibilités : β et A sont dans la même composante connexe ou non. Dans le premier cas, la configuration reste linéaire et dans le second cas, elle est non-connexe et alors, selon le lemme 3.4.10, elle restera non-connexe ou redeviendra à nouveau linéaire. Il en résulte que le rendez-vous sera impossible. □

Nous sommes maintenant prêts à prouver trois corollaires qui, réunis, excluent toutes les configurations initiales autres que les étoiles de grandeur au moins 3.

Corollaire 3.4.1 *Une configuration initiale non-connexe n'est pas regroupable.*

Preuve: Considérons tout d'abord une configuration initiale non-connexe avec 3 agents. Si tous les agents sont isolés, alors l'adversaire peut planifier leurs mouvements afin de les garder toujours séparés. Si un agent est isolé et que les deux autres sont adjacents, alors tous les noeuds les contenant ne sont pas équivalents. Il s'ensuit que l'adversaire peut planifier les mouvements des agents afin d'en garder un constamment isolé des autres agents. Cela prouve qu'une configuration initiale non-connexe avec exactement 3 agents n'est pas regroupable. Nous pouvons donc supposer que la configuration n'a pas 3 agents. Selon les lemmes 3.4.10 et 3.4.11, la configuration n'est pas regroupable. □

Corollaire 3.4.2 *Une configuration initiale de diamètre différent de 2 n'est pas regroupable.*

Preuve: Si le diamètre est de 1, il y a exactement 2 singletons adjacents. Ils sont similaires et, par conséquent, l'adversaire fera en sorte de les bouger simultanément sur des noeuds vides ou de les inter-changer. Dans les deux cas, le rendez-vous est impossible. Si le diamètre est supérieur à 2, alors la configuration doit être linéaire et le rendez-vous est impossible selon le lemme 3.4.11. □

Corollaire 3.4.3 *Une configuration initiale de diamètre 2 contenant un cycle d'agents n'est pas regroupable.*

Preuve: La seule configuration initiale possible de diamètre 2 avec un cycle est une configuration dont les agents forment un graphe biparti régulier complet avec au moins 2 agents dans chaque ensemble de la bipartition. Les agents dans chaque ensemble sont similaires. Si les agents dans au moins un ensemble bougent sur des singletons, alors l'adversaire peut faire en sorte de créer une configuration multitours. Si tous les agents bougent sur des noeuds vides, alors l'adversaire peut faire en sorte de créer une configuration multitours ou encore une configuration non-connexe sans tour. Le rendez-vous est impossible selon le lemme 3.4.9 et le corollaire 3.4.1. □

Nous pouvons finalement prouver le principal résultat négatif.

Preuve du théorème 3.4.1. Considérons une configuration initiale qui n'est pas une étoile avec au moins 3 agents. Si la configuration est non-connexe, alors elle n'est pas regroupable selon le corollaire 3.4.1. Si la configuration est connexe avec un diamètre différent de 2, alors elle n'est pas regroupable selon le corollaire 3.4.2. Nous pouvons donc assumer qu'elle a le diamètre 2. Puisque la configuration n'est pas une étoile, elle doit contenir un cycle d'agents et alors, elle n'est pas regroupable selon le corollaire 3.4.3.

3.5 L'algorithme

Le théorème 3.4.1 laisse seulement les étoiles de grandeur au moins 3 comme candidates pour les configurations regroupables. L'algorithme qui

suit, formulé pour un agent A, accomplit le rendez-vous pour toutes ces configurations et fonctionne également pour la détection faible de multiplicité, autrement dit, lorsqu'un agent peut seulement détecter s'il est seul ou pas sur le noeud où il est localisé.

Algorithme Regrouper-Tout

Si A est dans une tour **ou** a plus d'un seul voisin occupé
alors A ne bouge pas
sinon A bouge sur son seul voisin occupé.

Théorème 3.5.1 *L'algorithme Regrouper-Tout accomplit le rendez-vous pour toutes les configurations initiales regroupables.*

Preuve: En vue du théorème 3.4.1, il suffit de prouver que l'algorithme Regrouper-Tout accomplit le rendez-vous pour toutes les étoiles de grandeur au moins 3. Chacune de ces étoiles a exactement un singleton B avec plus d'un voisin occupé et tous les autres singletons sont adjacents à B. Puisque ni B, ni une tour ne peut bouger, tous les singletons autres que B vont se déplacer sur le noeud occupé par B et s'arrêteront. □

Chapitre 4

Rendez-vous asynchrone des agents mobiles anonymes avec vision restreinte dans la ligne infinie

Dans ce chapitre, nous étudions le problème de rendez-vous dans un graphe qui est une ligne infinie, c'est-à-dire dont chaque noeud a le degré 2. Le modèle utilisé est une généralisation du modèle du chapitre 3.

4.1 Problème et modèle

Le modèle utilisé dans ce chapitre diffère du modèle du chapitre 3 dans un seul aspect. Lors de l'observation, l'agent perçoit tous les noeuds à distance au plus d du noeud sur lequel il se trouve, où d est un entier positif appelé le rayon de vision. Pour chacun de ces noeuds, il obtient l'information si le noeud est vide, s'il y a un seul agent, ou s'il y a plus d'un agent qui l'occupe. Les agents ont donc la capacité forte de détection de multiplicité.

Comme auparavant, une configuration d'agents est dite *regroupable* s'il existe un algorithme qui regroupe tous les agents de la configuration sur un

noeud et les immobilise à partir de là, quelles que soient les actions de l'adversaire asynchrone. (L'algorithme peut même être adapté pour regrouper cette configuration spécifique.) Nous supposons que, dans une configuration initiale, il y a au plus un agent par noeud, c'est-à-dire, il n'y a pas de tour. Un algorithme de rendez-vous est *universel* s'il rassemble toutes les configurations regroupables. Nous étudions l'existence d'algorithmes universels de rendez-vous pour la ligne infinie.

4.2 Résultats

Nous observons tout d'abord que si la vision des agents n'est pas restreinte, il existe un algorithme universel de rendez-vous pour la ligne infinie. D'autre part, il résulte du chapitre 3 que pour le rayon de vision $d = 1$, il y a un algorithme universel pour tous les graphes bipartis réguliers. En effet, appliqué à la ligne, notre résultat du chapitre 3 implique que la seule configuration regroupable pour $d = 1$ est un segment de trois singletons, et donc l'algorithme de rendez-vous pour cette configuration est universel. Par conséquent, il est naturel de se demander s'il existe des algorithmes universels de rendez-vous pour des rayons de vision supérieurs. Le résultat principal de ce chapitre est la réponse négative à cette question : nous montrons qu'un algorithme universel de rendez-vous pour la ligne infinie n'existe pas pour n'importe quel rayon de vision $d > 1$. Notre résultat reste valable pour les anneaux de taille au moins $7d + 8$.

4.3 Notions et résultats préliminaires

Nous considérons une ligne infinie de noeuds de degré 2. Un noeud dans lequel il n'y a pas d'agents est appelé *vide*, sinon il est appelé *occupé*. Un seul agent occupant un noeud est appelé un *singleton* et plus d'un agent occupant un noeud forment une *tour*. Un noeud vide est noté 0, un singleton est noté 1, et une tour est notée $*$. Le nombre total d'agents est fini mais inconnu. Une configuration est une fonction sur l'ensemble des noeuds de la ligne avec des valeurs dans l'ensemble $\{0, 1, *\}$. Une configuration initiale ne contient pas de tours. Nous disons que les singletons sont adjacents (resp. les tours sont adjacentes, un singleton est adjacent à une tour), si les

noeuds respectifs sont adjacents. Étant donné que la position d'une configuration dans la ligne n'est pas significative, une configuration peut être identifiée avec une séquence finie de termes $\{0, 1, *\}$, de sorte que le premier et le dernier terme de la séquence ne sont pas des zéros, en utilisant la convention que tous les noeuds en dehors de cette séquence sont vides. Dans une configuration $(c_1, \ldots, c_r)$, les termes c_1 et c_r sont appelés *extrémités*. Une configuration est *symétrique* si les séquences correspondantes ont un axe de symétrie. Il existe deux types de symétrie dans une configuration : *noeud-symétrique* si l'axe passe par un noeud, et *arête-symétrique* si l'axe passe par une arête. Par exemple, la configuration $(*, 1, 0, 1, 0, 1, *)$ est noeud-symétrique, la configuration $(*, 1, 0, 1, 1, 0, 1, *)$ est arête-symétrique, et la configuration $(*, 1, 0, 1, 0, 0, 1, *)$ n'est pas symétrique. Le *diamètre* d'une configuration est la distance maximale entre n'importe quel pair d'agents.

Une *vue* d'un agent A est l'ensemble des deux séquences

$$\{(a_d, \ldots, a_1, \underline{1}, b_1, \ldots, b_d), (b_d, \ldots, b_1, \underline{1}, a_1, \ldots, a_d)\}$$

si l'agent est un singleton, et l'ensemble des deux séquences

$$\{(a_d, \ldots, a_1, \underline{*}, b_1, \ldots, b_d), (b_d, \ldots, b_1, \underline{*}, a_1, \ldots, a_d)\}$$

si l'agent est dans une tour. Les séquences sont de longueur $2d + 1$, le singleton central ou la tour centrale est souligné par souci de clarté, et a_i et b_i proviennent de l'ensemble $\{0, 1, *\}$. Nous définissons la vue comme une paire de séquences dont l'une est l'image miroir de l'autre, afin de saisir la particularité du modèle que l'agent ne peut pas distinguer sa gauche de sa droite. Une vue est symétrique si $a_i = b_i$, pour tout $i = 1, \ldots, d$. Les séquences $(a_1, \ldots, a_d)$ et $(b_1, \ldots, b_d)$ sont appelées les deux *côtés* de la vue de l'agent. Nous utilisons 0^k pour désigner une séquence de k noeuds vides consécutifs.

Comme au chapitre 3, les caractéristiques de notre modèle impliquent qu'un algorithme de rendez-vous donné peut avoir de nombreuses exécutions. Cela est dû à deux particularités du modèle. Tout d'abord, puisque les actions d'observation, de calcul et de déplacement de chaque agent sont asynchrones, des ordres différents de ces actions peuvent potentiellement conduire à des exécutions différentes. Deuxièmement, toute action de déplacement est basée sur la vue de l'agent obtenue dans l'action d'observation précédente. Si la vue est symétrique et l'algorithme prescrit de se déplacer quand cette vue

est donnée, l'agent ne peut pas distinguer sur lequel de ses deux voisins il doit se déplacer, et chaque choix conduit à une exécution potentiellement différente.

Comme auparavant, nous saisissons ces différentes possibilités en utilisant le concept d'un *adversaire* qui planifie les actions d'observation, de calcul et de déplacement et décide de bouger l'agent sur un voisin parmi les deux voisins de la vue symétrique, à chaque fois de la manière la plus nuisible à l'algorithme. Si nous prouvons que, pour une configuration initiale et pour tout algorithme hypothétique, l'adversaire peut faire les choix ci-dessus afin d'empêcher le rendez-vous, il s'ensuivra que la configuration initiale donnée n'est pas regroupable, parce que, pour n'importe quel algorithme hypothétique à partir de cette configuration, certaines exécutions possibles de celui-ci empêchent le rendez-vous.

Nous allons utiliser le fait suivant qui est prouvé de manière analogue comme pour les anneaux dans [40].

Proposition 4.3.1 *Les configurations arête-symétriques ne sont pas regroupables, pour tout rayon de vision $d \geq 1$ et pour une vision illimitée.*

La proposition suivante montre que si la vision des agents est illimitée, alors il existe un algorithme universel de rendez-vous pour la ligne.

Proposition 4.3.2 *Il existe un algorithme universel de rendez-vous dans la ligne infinie pour les agents ayant une vision sans restriction.*

Preuve: Observons d'abord que toute configuration de 2 agents n'est pas regroupable. Ceci est prouvé comme pour le cas de l'anneau [40]. Compte tenu de la proposition 4.3.1, les configurations arête-symétriques ne sont également pas regroupables. Par conséquent, il suffit de montrer un algorithme qui regroupe toutes les configurations qui ne sont pas arête-symétriques avec au moins trois agents. Considérons une telle configuration initiale $C = (1, 0^{x_1}, 1, 0^{x_2}, 1 \ldots, 0^{x_k}, 1)$, où x_i sont des entiers non négatifs représentant le nombre de noeuds vides consécutifs entre les singletons. Définissons les *signatures* de C comme les deux suites d'entiers non négatifs $(x_1, x_2, \ldots, x_k)$

et $(x_k, x_{k-1}, \ldots, x_1)$. Notez que les signatures sont égales si, et seulement si, la configuration C est symétrique.

Considérons d'abord le cas où le diamètre de C est pair. Donc C a un noeud central v. Supposons que le nombre total d'agents est 3. L'algorithme est le suivant. S'il y a un singleton sur v, il se déplace dans l'un des deux sens (car la vue est symétrique). Puis ce singleton (qui est le seul singleton autre que les extrémités) se déplace vers l'extrémité la plus proche jusqu'à ce qu'il la rejoigne pour former une tour. Une fois que la tour est créée, l'autre extrémité se déplace vers la tour jusqu'à ce qu'elle la rejoigne, complétant le rendez-vous. Si le noeud v est vide, le protocole ci-dessus est exécuté, sauf pour le premier pas du singleton sur v.

Si le nombre total d'agents est supérieur à 3, l'agent avec la plus petite distance positive à partir de v se déplace vers v. Cela se fait jusqu'à ce qu'une tour soit formée sur v. Notez que les extrémités ne bougeront pas jusqu'à ce que la tour soit formée, donc le noeud v peut être identifié par tous les agents. Une fois que la tour est formée sur v, dans la seconde partie de l'algorithme, l'agent le plus proche de la tour se déplace vers celle-ci jusqu'à ce qu'il rejoigne la tour. De cette façon, tous les agents joindront finalement la tour sur v, complétant ainsi le rendez-vous.

Considérons maintenant le cas où le diamètre de C est impair. Donc C n'est pas symétrique, car autrement la configuration serait arête-symétrique. Soit σ la plus petite des deux signatures de C dans l'ordre lexicographique. Soit w l'extrémité de C correspondant au premier terme de σ. L'algorithme est le suivant. L'agent le plus proche de w se déplace vers w jusqu'à ce qu'il le rejoigne pour former une tour. Notez qu'à toutes les étapes de cette partie de l'exécution, la signature dont le premier terme correspond au noeud w change, mais reste inférieure à l'autre, de sorte que le même agent se déplace vers w jusqu'à ce qu'il s'y joigne. Une fois que la tour est formée sur w, la seconde partie de l'algorithme fait en sorte qu'un agent le plus proche de la tour se déplace vers celle-ci jusqu'à ce qu'il s'y joigne. De cette façon, tous les agents seront finalement dans la tour sur w, complétant ainsi le rendez-vous.

Notez que dans les deux cas, les extrémités ne se déplacent pas jusqu'à ce qu'une tour soit créée, donc le diamètre de la configuration ne change pas

jusque-là, et tous les agents savent quel cas, (diamètre pair ou impair) devrait être considéré. Une fois que la tour est créée, l'algorithme est uniforme : l'agent(s) le plus proche de la tour bouge vers celle-ci. □

4.4 Inexistance d'un algorithme universel de rendez–vous pour le rayon de vision $d > 1$

Dans cette section, nous présentons le résultat principal de ce chapitre : pour le rayon de vision $d > 1$, il n'existe pas d'algorithme universel de rendez-vous dans la ligne infinie. Nous commençons par les deux lemmes suivants qui tiennent pour n'importe quel rayon de vision.

Lemme 4.4.1 *Une configuration dont les deux tours sont aux extrémités n'est pas regroupable, pour n'importe quel rayon de vision.*

Preuve: Considérons une configuration C dont les deux extrémités sont des tours, et n'importe quel algorithme de rendez-vous hypothétique pour cette configuration. Considérons un adversaire qui planifie les cycles d'opérations Observation-Calcul-Déplacement pour les agents en rondes synchrones de manière que :

- pour chaque agent, un cycle complet est effectué dans une ronde,
- tous les agents dans une tour effectuent les cycles correspondants dans la même ronde,
- les cycles pour les agents situés dans différents noeuds sont effectués durant des rondes différentes,

à condition que le diamètre de la configuration soit plus grand que 1. Par induction sur le nombre de rondes, il s'ensuit que, aussi longtemps que le diamètre est supérieur à 1, la configuration préserve l'invariant que ses extrémités sont des tours. Puisque l'algorithme hypothétique doit regrouper la configuration, le diamètre doit finalement diminuer à zéro. En effet, les caractéristiques de l'adversaire impliquent que le diamètre peut varier au plus de 1 durant une ronde.

Considérons la première ronde t lorsque le diamètre est d'au plus 1. Comme le diamètre peut varier au plus de 1 par ronde, le diamètre à la

ronde t est exactement 1. Donc, la configuration se compose de deux tours adjacentes et le lemme découle de la proposition 4.3.1. □

Lemme 4.4.2 *Considérons un rayon de vision quelconque et un algorithme qui regroupe la configuration avec deux noeuds occupés qui sont adjacents, l'un d'entre eux occupé par un singleton et l'autre par une tour. L'algorithme doit ordonner à la tour de rester inactive et au singleton de se déplacer sur la tour.*

Preuve: Considérons un adversaire qui planifie les cycles d'opérations Observation - Calcul - Déplacement pour les agents en rondes synchrones de manière que :

- pour chaque agent, un cycle complet est effectué dans une ronde,
- tous les agents dans une tour effectuent les cycles correspondants dans la même ronde,
- les cycles pour les agents situés dans différents noeuds sont effectués durant des rondes différentes, jusqu'à l'avant dernière ronde de l'exécution.

Considérons le dernier mouvement dans l'exécution avant qu'un agent se déplace pour compléter le rendez-vous. La tour et le singleton doivent être (à nouveau) adjacents. Supposons que les instructions de l'algorithme ne sont pas comme indiquées dans le lemme. Il y a trois cas :

Cas 1. La tour et le singleton sont instruit à rester inactifs.
Cela contredit l'accomplissement du rendez-vous.

Cas 2. La tour est instruite à se déplacer sur le singleton et le singleton à rester inactif.
Dans ce cas, l'adversaire exécute les cycles de tous les agents de la tour, sauf un agent. Ils se déplacent sur le singleton, formant ainsi une tour, et l'agent restant devient un singleton. La configuration est comme avant, ce qui contredit le rendez-vous.

Cas 3. La tour est instruite à se déplacer sur le singleton et le singleton à se déplacer sur la tour.
Dans ce cas, l'adversaire exécute les cycles de tous les agents simultanément. Le singleton bouge sur le noeud occupé par la tour et la tour bouge vers

le noeud occupé par le singleton. La configuration est comme avant, ce qui contredit le rendez-vous.

La contradiction dans les trois cas conclut la preuve. □

Afin de montrer qu'il n'y a pas d'algorithme universel de rendez-vous, nous allons fixer un rayon $d > 1$ et examiner les sept configurations suivantes, dans lesquels chaque lettre $a, b, c, a', b', c', e, f, g$ désigne un singleton. (Nous utilisons différentes lettres afin de simplifier les arguments) (voir Fig. 4.1, 4.2 et 4.3).

$C = (a, b, a')$
$C_1 = (a, b, c, 0^{d-1}, c', b', a')$
$C_2 = (a, b, c, 0^{d-2}, e, f, 0, g)$
$C'_1 = (a, b, c, 0^{d-1}, e, 0^{d-1}, c', b', a')$
$C'_2 = (a, b, c, 0^{d-1}, e, f, 0, g)$
$C''_1 = (a, b, 0, c)$
$C''_2 = (a, b, c, 0, e, f)$

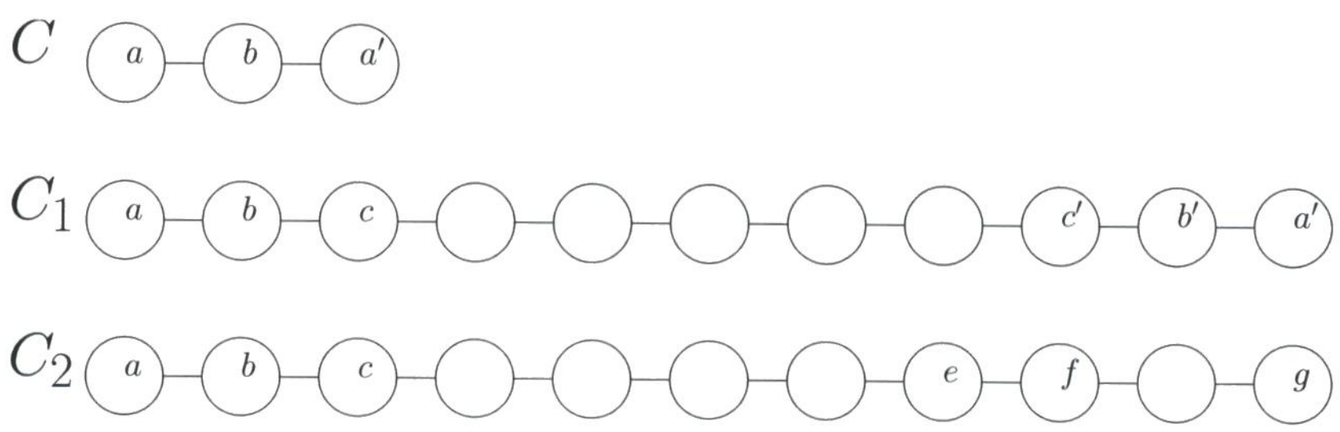

FIGURE 4.1 – Les configurations C, C_1 et C_2 avec $d = 6$

L'idée de la preuve de non-existence est la suivante. Selon le rayon de vision $d > 1$ (nous allons examiner les cas $d \geq 4$ pair, $d \geq 3$ impair et $d = 2$), nous allons montrer que dans chaque cas, trois de ces sept configurations sont regroupables, mais il n'existe pas d'algorithme unique regroupant toutes les trois. La preuve est divisée en une série de lemmes. Les quatre lemmes suivants garantissent la regroupabilité de certaines des configurations ci-dessus, en fonction du rayon de vision.

Lemme 4.4.3 *La configuration C est regroupable pour n'importe quel rayon de vision $d > 1$.*

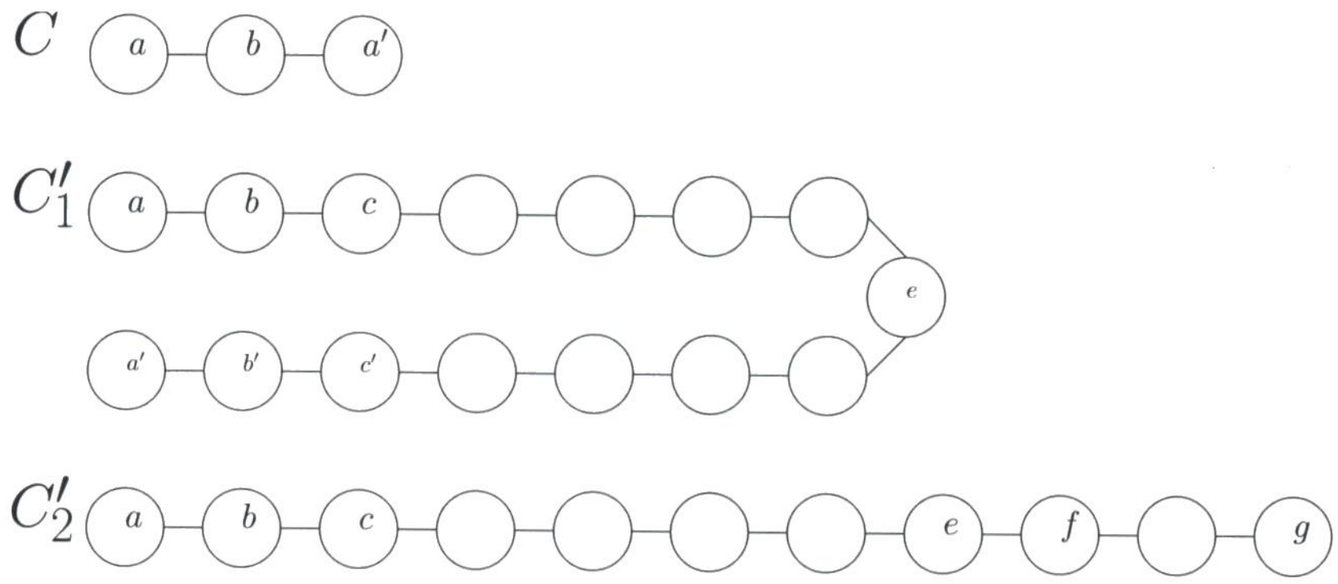

FIGURE 4.2 – Les configurations C, C'_1 et C'_2 avec $d = 5$

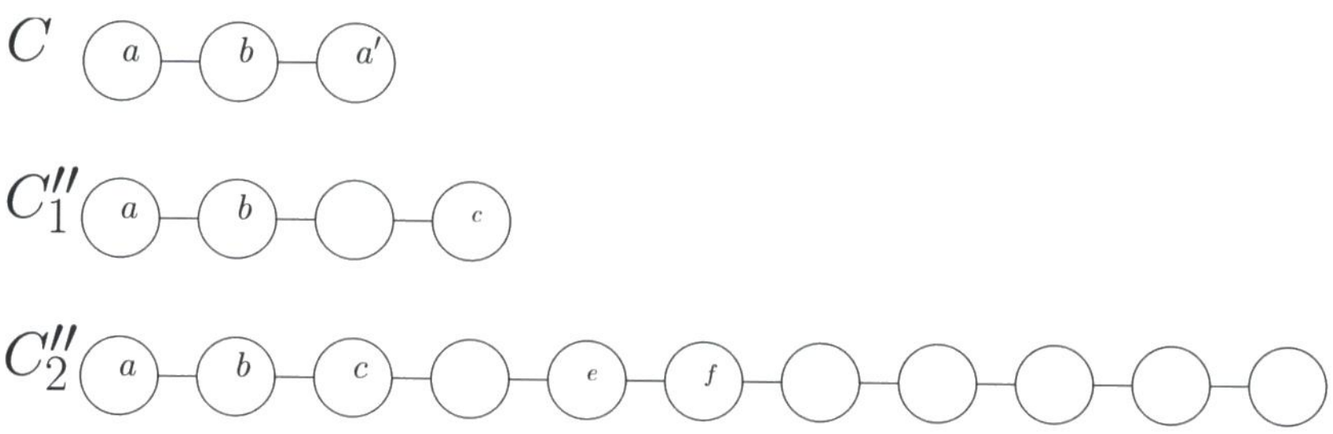

FIGURE 4.3 – Les configurations C, C''_1 et C''_2 avec $d = 2$

Preuve: L'algorithme de rendez-vous, indépendamment de d, est le suivant : un singleton adjacent à un noeud vide et à un noeud occupé bouge sur le noeud occupé ; un singleton adjacent à deux noeuds occupés ne bouge pas ; un agent dans une tour ne bouge pas. □

Lemme 4.4.4 *Les configurations C_1, C_2 sont regroupables pour les rayons de vision pairs $d \geq 4$.*

Preuve: Considérons un rayon de vision pair $d \geq 4$ et la configuration C_1. L'algorithme de rendez-vous pour cette configuration est le suivant. Le singleton c (respectivement c') s'éloigne des singletons a et b (respectivement a' et b') jusqu'à ce qu'il arrive à la distance $d/2$ de b (respectivement b'). Ils peuvent le faire parce que, pendant ce déplacement, c et c' sont les seuls singletons qui voient deux singletons adjacents d'un côté de leur vue (a et b

dans le cas de c et a' et b' dans le cas de c'), et au moins un autre singleton de l'autre côté de leur vue. Tous les autres agents restent inactifs jusqu'à ce qu'ils voient une tour. À la fin de cette partie, les singletons c et c' forment une tour au noeud central. Le reste de l'algorithme est le suivant : si un agent voit une tour et aucun agent entre lui-même et la tour, il va vers la tour ; un agent dans une tour ne bouge pas. Dans la deuxième partie, les singletons b et b' rejoingnent la tour, puis a et a' rejoignent la tour, complétant le rendez-vous.

Considérons un rayon de vision pair d et la configuration C_2. Notons que chaque singleton dans cette configuration a une vue différente. En particulier, aucun singleton ne voit une tour et f est le seul singleton qui voit un singleton unique, à une distance de 2 d'un côté de sa vue et plus d'un singleton de l'autre côté de sa vue. Initialement, le singleton f se déplace sur e formant une tour (il est le seul singleton qui peut se déplacer initialement). Le reste de l'algorithme est le suivant.

1. Un singleton qui voit une tour d'un côté de sa vue et aucun autre agent entre lui-même et la tour et qui ne voit aucun agent de l'autre côté de sa vue se déplace vers la tour.

2. Un singleton qui voit une tour d'un côté de sa vue et aucun autre agent entre lui-même et la tour et qui voit l'agent le plus proche de l'autre côté de sa vue à une distance de moins de d se déplace vers la tour.

3. Un singleton qui ne voit pas de tour et voit d'un côté de sa vue un singleton unique qui est à une distance d se déplace vers ce singleton.

4. Un agent dans une tour ne bouge pas.

Après que la tour soit formée, le singleton g se déplace vers elle, indépendamment des singletons a, b et c, joignant finalement la tour (clause 1 de l'algorithme ci-dessus). Ces trois singletons se déplacent comme suit. Le singleton c se déplace vers la tour et rejoint celle-ci (clause 2), ensuite b se déplace vers le noeud adjacent à la tour (clause 2). Ensuite a fait un mouvement vers b (clause 3), ensuite b se déplace sur la tour (clause 2), et finalement a se déplace vers la tour et la rejoint (clause 1) complétant le rendez-vous. □

Lemme 4.4.5 *Les configurations C'_1, C'_2 sont regroupables pour les rayons de vision impairs $d \geq 3$.*

Preuve: Considérons un rayon impair de vision $d \geq 3$ et la configuration C'_1.

L'algorithme de rendez-vous pour cette configuration est le suivant.

1. Un singleton qui voit d'un côté de sa vue exactement deux singletons à des distances 1 et 2 et qui voit au moins un singleton non isolé à une distance de 2 de l'autre côté de sa vue se déplace vers ce dernier singleton.

2. Un singleton qui voit d'un côté de sa vue exactement un singleton adjacent et qui voit un singleton unique, à une distance $d-1$ de l'autre côté de sa vue se déplace vers ce dernier singleton.

3. Un singleton qui voit de chaque côté de sa vue exactement deux singletons à des distances 2 et d se déplace (dans l'une des deux directions, puisque sa vue est symétrique).

4. Un singleton qui voit d'un côté de sa vue exactement 2 singletons à des distances $d-2$ et d et qui voit de l'autre côté de sa vue le singleton le plus proche à une distance de 3 se déplace vers le dernier singleton.

5. Un singleton qui voit d'un côté de sa vue exactement 2 singletons à des distances 1 et 3 et qui voit de l'autre côté de sa vue exactement 2 singletons à des distances $d-2$ et d se déplace vers les singletons à des distances 1 et 3. (Notez que pour $d = 3$ les deux côtés de la vue sont identiques, c'est-à-dire, la vue est symétrique, et dans ce cas le singleton se déplace d'un coté ou de l'autre).

6. Un singleton qui voit une tour d'un côté de sa vue et aucun autre agent entre lui-même et la tour et qui voit aucun agent de l'autre côté de sa vue se déplace vers la tour.

7. Un singleton qui voit une tour sur un côté de sa vue et aucun autre agent entre lui-même et la tour et qui voit l'agent le plus proche de l'autre côté de sa vue à une distance de moins de d se déplace vers la tour.

8. Un singleton qui ne voit pas de tour et voit d'un côté de sa vue un singleton unique qui est à une distance d se déplace vers ce singleton.

9. Un agent dans une tour ne bouge pas.

Tout d'abord, le singleton c (resp. c') se déplace à distance 2 à partir de e (clause 1 de l'algorithme). Ensuite, b (resp. b') effectue un mouvement vers c (resp. c') résultant en la configuration $(a, 0, b, 0^{d-3}, c, 0, e, 0, c', 0^{d-3}, b', 0, a')$ (clause 2). Ensuite e se déplace d'un noeud dans l'une des directions, puisque sa vue est symétrique (clause 3). Sans perte de généralité supposons que e se déplace vers c' résultant en la configuration $(a, 0, b, 0^{d-3}, c, 0, 0, e, c', 0^{d-3}, b', 0, a')$.

À ce stade, le seul singleton qui peut se déplacer est c. Si $d \neq 3$, alors la clause de l'algorithme qui le fait bouger est la clause 4 et si $d = 3$, alors les clauses de l'algorithme qui le font bouger sont la clause 4 et la clause 8, mais dans ce cas, les mouvements sont identiques. Dans les deux cas c se déplace vers e. Maintenant, le seul singleton qui peut se déplacer est c' (clause 5). Pour tout $d \neq 3$, il se déplace sur e créant une tour, et pour $d = 3$, il se déplace soit sur e soit sur b' (comme sa vue est symétrique) créant également une tour.

Une fois que la tour est créée, tous les singletons bougent vers la tour d'une manière similaire à celle décrite dans la dernière partie de la preuve du lemme 4.4.4, ils se joignent à la tour un par un sur chaque côté de celle-ci, complétant ainsi le rendez-vous.

Ensuite, considérons un rayon impair de vision $d \geq 3$ et la configuration C_2'. Notez que chaque singleton dans cette configuration a une vue différente. En particulier, aucun singleton ne voit une tour et f est le seul singleton qui voit un singleton unique, à une distance de 2 d'un côté de sa vue et un singleton unique sur l'autre côté de sa vue. Initialement, le singleton f se déplace sur e formant une tour (il est le seul singleton qui peut se déplacer initialement). Le reste de l'algorithme est le suivant.

1. Un singleton qui voit une tour d'un côté de sa vue et aucun autre agent entre lui-même et la tour et qui ne voit aucun agent de l'autre côté de sa vue se déplace vers la tour.

2. Un singleton qui voit une tour d'un côté de sa vue et aucun autre agent entre lui-même et la tour et qui voit l'agent le plus proche de l'autre côté de sa vue à une distance de moins de d se déplace vers la tour.

3. Un singleton qui ne voit pas de tour et voit d'un côté de sa vue un singleton unique qui est à une distance d se déplace vers ce singleton.

4. Un agent dans une tour ne bouge pas.

Après que la tour soit formée, le singleton g se déplace vers elle, indépendamment des singletons a, b et c, joignant finalement la tour (clause 1 de l'algorithme ci-dessus). Ces trois singletons se déplacent comme suit. Le singleton c se déplace vers le noeud à côté de la tour (clause 2), ensuite b fait un mouvement vers c (clause 3), puis c rejoint la tour (clause 2). Ensuite b se déplace vers le noeud à distance 2 de la tour (clause 2), ensuite a fait un mouvement vers b (clause 3), puis b fait un mouvement vers la tour (clause 2), ensuite a fait un mouvement vers b (clause 3), puis b se déplace sur la tour (clause 2) et enfin, a se déplace vers la tour et rejoint celle-ci (clause 1) complétant le rendez-vous. □

Lemme 4.4.6 *Les configurations C_1''', C_2''' sont regroupables pour le rayon de vision $d = 2$.*

Preuve: Considérons le rayon de vision $d = 2$ et la configuration C_1'''. L'algorithme de rendez-vous pour cette configuration est le suivant. Au début, tous les noeuds ont des vues différentes et c se déplace vers b, pendant que a et b sont inactifs. Maintenant un segment (a, b, c) est formé et tous les noeuds ont des vues différentes de celles de la configuration C_1'''. Le reste de l'algorithme est le même que pour la configuration C_1.

Enfin, considérons le rayon de vision $d = 2$ et la configuration C_2'''. L'algorithme de rendez-vous pour cette configuration est le suivant.

1. Un singleton qui est à une extrémité d'un segment de longueur d'au plus 2 et voit un noeud occupé derrière un voisin vide, se déplace vers ce voisin vide.

2. Un singleton qui voit 4 autres singletons se déplace vers l'un des voisins.

3. Un singleton qui voit une tour adjacente et voit un noeud vide derrière elle se déplace sur la tour.

4. Un singleton qui voit un singleton adjacent et une tour adjacente se déplace sur la tour.

5. Un singleton qui voit une tour adjacente et un singleton derrière elle d'un côté, et les noeuds vides de l'autre côté, se déplace sur la tour.

6. Un agent qui ne satisfait aucune des conditions 1 – 5 ne bouge pas.

Au début, seul le singleton e satisfait une des conditions 1 à 5, à savoir la condition 1. Il se déplace vers c. Alors seulement le singleton f satisfait une des conditions 1 à 5, à savoir la condition 1. Il se déplace vers e, formant un segment (a, b, c, e, f). Maintenant c est le seul agent qui satisfait une des conditions 1 à 5, à savoir la condition 2. S.p.d.g. il se déplace sur e formant une tour. Maintenant, la configuration est $(a, b, 0, *, f)$. Le singleton f satisfait la condition 3 et peut se déplacer sur la tour à tout moment. Le singleton b satisfait la condition 1 et peut se déplacer vers la tour à tout moment. L'adversaire peut planifier les mouvements de b et f dans un ordre arbitraire. Après le mouvement de b le singleton a satisfait la condition 1 et se déplace vers b. (Le mouvement de f et les mouvements de a et b sont indépendants et peuvent être programmés arbitrairement.) Après le mouvement de a, le singleton b satisfait la condition 4 et se déplace sur la tour. Alors a satisfait la condition 1 et peut se déplacer vers la tour. Les singletons a et f peuvent se déplacer sur la tour dans un ordre arbitraire, soit par la condition 3 ou la condition 5. Ceci termine le rendez-vous. □

Les trois prochains lemmes montrent l'inexistence d'un algorithme universel de rendez-vous pour les différentes classes de rayons de vision.

Lemme 4.4.7 *Il n'y a pas d'algorithme universel pour les rayons de vision pairs $d \geq 4$ (voir Fig. 4.1).*

Preuve: Supposons qu'il existe un algorithme universel de rendez-vous $\mathcal{A}$ pour un rayon de vision pair $d \geq 4$. En particulier, cet algorithme doit regrouper les configurations C, C_1 et C_2. Considérons la configuration C. L'algorithme doit déplacer au moins un des agents sinon le rendez-vous ne se produirait jamais. Il y a 3 mouvements possibles dans C que les agents peuvent faire. (Notons que a et a' ont la même vue dans C.) Le déplacement α, lorsque b se déplace sur l'un des singletons adjacents (soit a ou a'), le déplacement β, lorsque a se déplace sur b (resp. a' se déplace sur b) et le déplacement γ, lorsque a s'éloigne de b (resp. a' s'éloigne de b).

Maintenant, considérons la configuration C_1. Les singletons b et b' dans C_1 ont la même vue que le singleton b dans C. Les singletons a et a' dans C_1 ont la même vue que les singletons a et a' dans C. Considérons le comportement suivant de l'adversaire si le déplacement α est exécuté dans C_1. L'adversaire planifie les mouvements de b sur a et b' sur a' de manière synchrone, créant une configuration dans laquelle les tours sont aux extrémités. Selon le lemme 4.4.1, le rendez-vous est impossible. Maintenant, considérons le comportement suivant de l'adversaire si le mouvement β est exécuté dans C_1. L'adversaire planifie les mouvements de a sur b et a' sur b' de manière synchrone, créant une configuration dans laquelle les tours sont aux extrémités. À nouveau, selon le lemme 4.4.1, le rendez-vous est impossible. Par conséquent, nous pouvons restreindre l'attention au mouvement γ pour la configuration C.

Considérons maintenant la configuration C_2. Le singleton a dans C_2 a la même vue que les singletons a et a' dans C. Considérons le comportement suivant de l'adversaire si le mouvement γ est exécuté dans C_2. L'adversaire planifie tout d'abord le mouvement du singleton a de s'éloigner de b. Il en résulte une configuration arête-symétrique $(a, 0, b, c, 0^{d-2}, e, f, 0, g)$. Selon la proposition 4.3.1, le rendez-vous est impossible.

Il s'ensuit que les mouvements α et β exécutés dans C_1 conduisent à une configuration non regroupable et le mouvement γ exécuté dans C_2 conduit à une configuration non regroupable, sous certains comportements de l'adversaire. Cela implique que l'algorithme $\mathcal{A}$ ne peut pas regrouper les configurations C, C_1 et C_2, bien que ces configurations soient regroupables (chacune par un algorithme dédié), selon les lemmes 4.4.3 et 4.4.4. Ceci contredit l'hypothèse que $\mathcal{A}$ est universel. □

Lemme 4.4.8 *Il n'y a pas d'algorithme universel pour les rayons de vision impairs $d \geq 3$ (voir Fig. 4.2).*

Preuve: Supposons qu'il existe un algorithme universel de rendez-vous $\mathcal{A}$ pour un rayon de vision impair $d \geq 3$. En particulier, cet algorithme doit regrouper les configurations C, C'_1 et C'_2. Considérons la configuration C. L'algorithme doit déplacer au moins un des agents sinon le rendez-vous ne se produirait jamais. Il y a 3 mouvements possibles dans C que les agents peuvent faire. (Notons que a et a' ont la même vue dans C.) Le déplacement α, lorsque b se déplace sur l'un des singletons adjacents (soit a ou a'), le déplacement β, lorsque a se déplace sur b (resp. a' se déplace sur b) et le déplacement γ, lorsque a s'éloigne de b (resp. a' s'éloigne de b).

Maintenant, considérons la configuration C_1. Les singletons b et b' dans C_1 ont la même vue que le singleton b dans C. Les singletons a et a' dans C_1 ont la même vue que les singletons a et a' dans C. Considérons le comportement suivant de l'adversaire si le déplacement α est exécuté dans C'_1. L'adversaire planifie les mouvements de b sur a et b' sur a' de manière synchrone, créant une configuration dans laquelle les tours sont aux extrémités. Selon le lemme 4.4.1, le rendez-vous est impossible. Maintenant, considérons le comportement suivant de l'adversaire si le mouvement β est exécuté dans C'_1. L'adversaire planifie les mouvements de a sur b et a' sur b' de manière synchrone, créant une configuration dans laquelle les tours sont aux extrémités. À nouveau, selon le lemme 4.4.1, le rendez-vous est impossible. Par conséquent, nous pouvons restreindre l'attention au mouvement γ pour la configuration C.

Considérons maintenant la configuration C'_2. Le singleton a dans C'_2 a la même vue que les singletons a et a' dans C. Considérons le comportement suivant de l'adversaire si le mouvement γ est exécuté dans C'_2. L'adversaire planifie tout d'abord le mouvement du singleton a de s'éloigner de b. Il en résulte une configuration arête-symétrique $(a, 0, b, c, 0^{d-1}, e, f, 0, g)$. Selon la proposition 4.3.1, le rendez-vous est impossible.

Il s'ensuit que les mouvements α et β exécutés dans C'_1 conduisent à une configuration non regroupable et le mouvement γ exécuté dans C'_2 conduit à

une configuration non regroupable, sous certains comportements de l'adversaire. Cela implique que l'algorithme $\mathcal{A}$ ne peut pas regrouper les configurations C, C'_1 et C'_2, bien que ces configurations soient regroupables (chacune par un algorithme dédié), selon les lemmes 4.4.3 et 4.4.5. Ceci contredit l'hypothèse que $\mathcal{A}$ est universel. □

Lemme 4.4.9 *Il n'y a pas d'algorithme universel pour le rayon de vision $d = 2$ (voir Fig. 4.3).*

Preuve: Supposons qu'il existe un algorithme de rendez-vous universel $\mathcal{A}$ pour le rayon de vision $d = 2$. En particulier, cet algorithme doit regrouper les configurations C, C''_1 et C''_2.

Considérons la configuration C''_1. Au moins un des agents doit se déplacer sinon le rendez-vous ne se produira jamais. Il y a 6 mouvements possibles en C''_1 : chacun des agents peut se déplacer d'un côté ou de l'autre. Parmi ces 6 mouvements, trois mènent à une configuration non regroupable. Nous analysons ces mouvements ci-dessous.

1. a s'éloigne de b.
Cela conduit à la configuration $(a, 0, b, 0, c)$ où a et c ont la même vue. Si b se déplace dans cette configuration, on obtient (s.p.d.g.) la configuration $(a, 0, 0, b, c)$ où b et c ont la même vue. Si l'algorithme leur ordonne de se déplacer sur le singleton adjacent, l'adversaire les déplace de manière synchrone et ils échangent leurs positions reproduisant la même configuration. Si l'algorithme leur ordonne de s'éloigner l'un de l'autre, l'adversaire planifie tout d'abord le cycle d'Observation-Calcul-Déplacement de b revenant à nouveau à la configuration $(a, 0, b, 0, c)$. Cela montre que b ne peut pas se déplacer dans la configuration $(a, 0, b, 0, c)$. Par conséquent, dans cette configuration, les singletons a et c doivent se déplacer. Il y a deux possibilités : ils peuvent soit s'éloigner de b ou bouger vers b. S'ils s'éloignent de b, l'adversaire les déplace simultanément et donc a, b et c ont des vues vides qui empêchent le rendez-vous (l'adversaire peut alors déplacer a et c de plus en plus loin de b). L'autre possibilité est que a et c bougent vers b. Dans ce cas, l'adversaire planifie le cycle d'Observation-Calcul-Déplacement de c en premier créant ainsi à nouveau la configuration C''_1. C'est pourquoi nous concluons qu'aucune des possibilités ci-dessus ne garantit le rendez-vous. Il s'ensuit que a ne peut pas s'éloigner de b dans la configuration C''_1.

2. b bouge sur a.
Cela conduit à une configuration contenant une tour et le singleton c à une distance de 3 de celle-ci. Il s'agit d'une configuration non regroupable puisque l'adversaire peut planifier l'éloignement du singleton et de la tour.

3. c s'éloigne de b.
Cela conduit à la configuration $(a, b, 0, 0, c)$ et l'argument similaire à celui du cas 1 (ci-dessus) s'applique.

Par conséquent, dans la configuration C_1'', nous pouvons restreindre l'attention aux autres mouvements possibles. Si dans la configuration C_1'' ni a ni c ne se déplacent vers b, alors le seul mouvement possible est celui de b vers le noeud vide adjacent et ce mouvement créerait perpétuellement une configuration équivalente à C_1''. Nous pouvons donc supposer que soit a se déplace sur b soit c se déplace vers b. Notez que le singleton b peut se déplacer sur le noeud vide adjacent, mais dans ce cas *les deux mouvements* (a se déplace sur b et c se déplace vers b) doivent se produire. En effet, si un seul de ces mouvements se produit, l'adversaire pourrait planifier les mouvements de b de sorte que a et c soient toujours inactifs, créant perpétuellement une configuration équivalente à C_1''. Notons par λ le mouvement de a sur b, par μ le mouvement de c sur b et par ν le mouvement de b sur le noeud vide adjacent dans la configuration C_1''.

Maintenant, considérons la configuration C. L'algorithme doit déplacer au moins un des agents sinon le rendez-vous ne se produirait jamais. Il y a 3 mouvements possibles dans C que les agents peuvent faire. (Notez que a et a' ont la même vue dans C.). Le mouvement α, lorsque b se déplace sur l'un des singletons adjacents (soit a ou a'), le mouvement β, lorsque a se déplace sur b (resp. a' se déplace sur b) et le mouvement γ, lorsque a s'éloigne de b (resp. a' s'éloigne de b). Remarquons que le mouvement γ dans la configuration C conduit à une configuration non regroupable. En effet, l'adversaire peut planifier les mouvements de a et a' simultanément, créant ainsi la configuration $(a, 0, b, 0, a')$. L'argument est alors le même que dans le cas 1 ci-dessus. Par conséquent, nous pouvons restreindre l'attention aux mouvements α et β dans la configuration C.

Maintenant, considérons la configuration C_2''. Le singleton f dans C_2'' a la même vue que le singleton a dans C_1''. Le singleton e dans C_2'' a la même

vue que le singleton b dans C_1''. Le singleton a dans C_2'' a la même vue que les singletons a et a' dans C. Le singleton b dans C_2'' a la même vue que le singleton b dans C.

Considérons le mouvement λ dans la configuration C_1'' et les mouvements α et β dans la configuration C. Puisque f dans C_2'' a la même vue que a dans C_1'' et a et b dans C_2'' ont la même vue que (respectivement) a et b dans C, le mouvement λ effectué en C_2'' créerait une tour à une extrémité et les mouvements α et β effectués en C_2'' créeraient une tour à l'autre extrémité. Selon le lemme 4.4.1, le rendez-vous est impossible. Cela implique que le mouvement λ dans la configuration C_1'' est impossible. Également, le mouvement ν dans cette configuration est impossible (rappelons que ν implique à la fois λ et μ en C_1''). Il s'ensuit que le seul mouvement possible dans la configuration C_1'' est le mouvement μ.

Dans la configuration C_2'', les mouvements α ou β doivent se produire. L'adversaire peut planifier ces mouvements d'abord, créant une tour à une extrémité. La configuration qui en résulte est soit $D_1 = (*, 0, c, 0, e, f)$ soit $D_2 = (*, c, 0, e, f)$. Maintenant ni e ni f ne peuvent se déplacer, car les mouvements seraient λ et ν. Considérons le singleton c. Il y a trois cas.

Cas 1. c bouge vers la tour.
Après ce mouvement dans la configuration D_2 la tour ne voit aucun autre agent et ne peut donc pas se déplacer ; après ce déplacement dans la configuration D_1, le singleton c est adjacent à la tour, selon le lemme 4.4.2, il rejoint la tour, qui alors ne voit pas d'autre agent et ne peut donc pas se déplacer. Dans D_1 et D_2 après le mouvement de c, chacun des singletons e et f ne voient que l'autre et, par conséquent le rendez-vous est impossible.

Cas 2. c bouge vers e.
L'adversaire planifie ce mouvement en premier. Puisque e a la même vue que b dans la configuration C et f a la même vue que a dans la configuration C, les mouvements α ou β peuvent être programmés par l'adversaire pour créer une tour à l'autre extrémité, ce qui interdit le rendez-vous selon le lemme 4.4.1.

Cas 3. c ne bouge pas.
Comme aucun singleton ne se déplace dans ce cas, la tour doit finalement

devenir adjacente à c. Selon le lemme 4.4.2 la tour ne peut pas bouger sur c et donc le rendez-vous est impossible.

Cela implique que l'algorithme $\mathcal{A}$ ne peut pas regrouper les configurations C, C_1'' et C_2'', bien que ces configurations soient regroupables (chacun par un algorithme dédié), selon les lemmes 4.4.3 et 4.4.6. Ceci contredit l'hypothèse que $\mathcal{A}$ est universel. □

Les lemmes ci-dessus impliquent notre résultat principal.

Théorème 4.4.1 *Il n'existe pas d'algorithme universel de rendez-vous pour la ligne infinie, pour les rayons de vision $d > 1$.*

Preuve: Les lemmes 4.4.7 et 4.4.8 montrent qu'il n'y a pas d'algorithme universel pour les rayons de vision $d > 2$. Le lemme 4.4.9 montre qu'il n'y a pas d'algorithme universel pour le rayon de vision $d = 2$. □

4.5 Le cas des anneaux

Nous avons montré que pour n'importe quel rayon de vision $d > 1$ il n'existe pas d'algorithme universel de rendez-vous pour la ligne infinie. Notre résultat reste valable pour les anneaux de taille d'au moins $7d + 8$. En effet, toutes les configurations utilisées dans notre preuve ont tout au plus 7 agents. Dans un anneau de taille d'au moins $7d+8$, l'adversaire peut toujours garantir que tous les agents ont la même vue que dans la ligne infinie : il peut planifier les mouvements des agents de telle sorte qu'ils ne se voient pas les uns les autres de «l'autre côté» de l'anneau.

Chapitre 5

Rendez-vous asynchrone de deux agents mobiles anonymes dans les graphes arbitraires

Deux agents mobiles anonymes (identiques) doivent se rencontrer dans un graphe connexe inconnu arbitraire et possiblement infini. Les agents sont modélisés comme des points, ils débutent dans des noeuds du graphe choisis par l'adversaire et la route de chacun d'eux dépend uniquement de la section déjà traversée du graphe et, dans le cas du rendez-vous probabiliste, des résultats du lancer d'une pièce de monnaie (pile ou face). Les agents ne voient aucun noeud autre que celui qu'ils visitent actuellement. La marche de chaque agent dépend également d'un adversaire asynchrone qui peut arbitrairement varier la vitesse d'un agent, l'immobiliser, ou même le faire marcher en arrière, tant que la marche de l'agent sur chaque arête est continue, ne la quitte pas et la couvre au complet. Le rendez-vous signifie que les deux agents doivent être au même moment au même endroit, soit dans un noeud soit à l'intérieur d'une arête.

Dans le scénario déterministe, nous caractérisons les positions initiales des agents pour lesquelles le rendez-vous est faisable et nous donnons un algorithme qui garantit le rendez-vous asynchrone pour chacune de ces positions dans un graphe connexe arbitraire. Dans le scénario probabiliste, nous montrons un algorithme qui accomplit le rendez-vous avec probabilité 1, pour

des positions initiales arbitraires dans un graphe connexe arbitraire. Dans les deux cas, le graphe peut être fini ou infini (énumérable).

5.1 Problème et modèle

Le réseau est modélisé par un graphe connexe non-orienté. Les noeuds du graphe n'ont pas d'étiquette, mais les ports dans chaque noeud ont des étiquettes $1, \ldots, d$, où d est le degré du noeud. Les agents sont modélisés par des points qui circulent le long des arêtes du graphe. Chaque agent conçoit sa route, qui est une séquence d'arêtes (les arêtes consécutives étant incidentes), et l'adversaire contrôle la vitesse de chaque agent, peut varier sa vitesse, l'immobiliser, ou même le faire marcher en arrière, tant que la marche de l'agent dans chaque arête est continue, ne la quitte pas et la couvre au complet. Dans cette version asynchrone du problème de rendez-vous, se rencontrer sur un noeud peut être impossible même dans un graphe avec uniquement deux noeuds, puisque l'adversaire asynchrone contrôle la vitesse des agents et peut les faire visiter un noeud à différents moments. Par conséquent, il est nécessaire de relâcher les contraintes de rendez-vous en permettant aux agents de se rencontrer sur une arête. Une telle définition du rendez-vous est naturelle, par exemple quand des robots traversent un labyrinthe ou quand des gens se promènent dans les rues d'une ville inconnue. Puisque les agents peuvent se rencontrer sur une arête, le graphe doit être représenté géométriquement sans que les arêtes se croisent. (De tels croisements permettraient une rencontre «accidentelle» entre deux agents localisés sur des arêtes différentes, ce qui ne devrait pas permettre le rendez-vous dans le graphe.) Par conséquent, nous considérons un plongement du graphe sous-jacent dans l'espace tridimensionnel Euclidien, avec les noeuds du graphe étant des points de l'espace et les arêtes étant des segments de lignes disjoints joignant ces points. Les agents sont modélisés comme des points se déplaçant dans le plongement. Ce modèle de mouvement asynchrone des agents qui doivent se rencontrer dans un graphe a été préalablement utilisé dans [9, 19, 24].

Si les noeuds du graphe ont des identificateurs uniques, alors un algorithme simple est de se rencontrer sur le noeud avec le plus petit identificateur, donc pour les graphes finis, le problème de rendez-vous se réduit au problème d'exploration d'un graphe. Toutefois, dans beaucoup d'applications, lorsque le rendez-vous est nécessaire dans un réseau avec une topologie

inconnue, les identificateurs des noeuds peuvent être inaccessibles, les agents ne sont peut-être pas capables de percevoir ces identificateurs dû à des limitations sensorielles ou les noeuds ne révèlent pas leur identificateur pour des raisons de confidentialité. Par conséquent, il est important de créer des algorithmes de rendez-vous pour les agents qui évoluent dans les graphes anonymes (les graphes avec des noeuds sans identificateurs). Notons qu'il est important qu'un agent soit en mesure d'identifier les ports localement (sur le noeud où il réside), autrement l'adversaire pourrait faire en sorte qu'un agent ne choisisse pas une arête spécifique empêchant ainsi le rendez-vous (même dans les graphes simples comme les arbres). Cela justifie l'hypothèse commune dans la littérature que nous adoptons dans ce chapitre : les ports sur un noeud de degré d sont énumérés $1, 2, \ldots, d$. L'identification des ports sur un noeud donné est fixe : chaque agent perçoit le même identificateur de ports sur les noeuds du graphe. Toutefois, il n'y a pas de cohérence entre l'identification de ces ports pour divers noeuds. Lorsqu'un agent quitte un noeud, il connaît le port par lequel il le quitte et lorsqu'il entre dans un noeud, il connaît également le port par lequel il entre ainsi que le degré du noeud. Un agent ne connaît ni la topologie du graphe ni la distance qui le sépare de l'autre agent. Les agents ne peuvent pas marquer les noeuds ou les arêtes d'une quelconque façon. Chaque agent s'immobilise lorsqu'il rencontre l'autre agent.

Contrairement à [9, 20, 24], où les agents étaient distinguables par leur identificateurs ou par les coordonnées connues de leurs positions initiales, nous supposons plutôt que les agents sont anonymes (identiques) et qu'ils exécutent le même algorithme. Afin d'accomplir le rendez-vous, de tels agents anonymes doivent briser la symétrie pour éviter de faire des mouvements identiques qui les garderaient dans des positions séparées en tout temps. Dans le scénario déterministe, cela peut être fait en exploitant différentes vues des deux agents de leur position initiale, et dans le scénario probabiliste, le résultat du lancer d'une pièce de monnaie (pile ou face) peut être utilisé. Il se révélera que, dans certains cas, même le rendez-vous déterministe, avec les agents qui partent des positions symétriques, est possible. Nous n'imposons aucune restriction sur la mémoire des agents : d'un point de vue de calcul, les agents sont modélisés par des machines de Turing.

Deux notions importantes utilisées pour décrire le mouvement des agents est la *route* d'un agent et sa *marche*. Grosso modo, l'agent choisit la route où

il se déplace et l'adversaire décrit la marche sur cette route, décidant comment l'agent se déplace. Plus spécifiquement, ces notions sont définies comme suit. L'adversaire place initialement un agent dans un noeud du graphe. La route est choisie par l'agent et est définie comme suit. L'agent choisit un des ports disponibles sur le noeud où il réside. Après avoir atteint l'autre extrémité de l'arête correspondante, l'agent lit le port par lequel il est entré ainsi que le degré du noeud où il entre. Ensuite, il choisit un port disponible où il est, et ainsi de suite, indéfiniment (jusqu'au rendez-vous). La route résultante de l'agent est la séquence correspondante d'arêtes $(e_1, e_2, \dots)$, telle que e_i est incidente à e_{i+1}. Cette séquence est un chemin (pas nécessairement simple) dans le graphe.

Nous allons maintenant décrire formellement la marche f d'un agent sur sa route. Soit $R = (e_1, e_2, \dots)$, la route d'un agent. Soit $e_i = \{v_{i-1}, v_i\}$. Soit $(t_0, t_1, t_2, \dots)$, où $t_0 = 0$, une suite croissante de nombres réels, choisie par l'adversaire, qui représente des points dans le temps. Soit $f_i : [t_i, t_{i+1}] \to [v_i, v_{i+1}]$ n'importe quelle fonction continue choisie par l'adversaire, telle que $f_i(t_i) = v_i$ et $f_i(t_{i+1}) = v_{i+1}$. Pour tous $t \in [t_i, t_{i+1}]$, nous définissons $f(t) = f_i(t)$. L'interprétation de la marche f est la suivante : au temps t l'agent est au point $f(t)$ de sa route. Cette définition générale de la marche et le fait que (contrairement à la route), elle est conçue par l'adversaire, sont des façons de formaliser les caractéristiques asynchrones du processus. Le mouvement de l'agent peut se faire à une vitesse arbitraire, l'adversaire peut même parfois immobiliser l'agent ou le déplacer d'avant en arrière, tant que la marche dans chaque arête de la route est continue et la couvre au complet. Cette définition rend l'adversaire très puissant et, par conséquent, le rendez-vous est difficile à réaliser.

Notez que la capacité de l'adversaire asynchrone de produire toute marche continue à l'intérieur des arêtes de la route déterminée par les agents implique la différence suivante significative par rapport au scénario synchrone. Alors que dans ce dernier scénario, le mouvement relatif des agents ne dépend que de leurs routes et est donc entièrement contrôlé par les agents, dans le cadre asynchrone, ce mouvement relatif est également contrôlé par l'adversaire.

Les agents avec les routes R_1 et R_2 et avec des marches f_1 et f_2 se rencontrent à l'instant t, si les points $f_1(t)$ et $f_2(t)$ sont identiques. Un rendez-vous est garanti pour les routes R_1 et R_2, si les agents utilisant ces routes

se rencontrent à un certain instant t, indépendamment de la marche choisie par l'adversaire. Un algorithme de rendez-vous exécuté par un agent dans un graphe produit la route de l'agent, connaissant son point de départ (et les résultats des lancers de pièces dans le scénario aléatoire). Nous disons que le rendez-vous asynchrone est *réalisable* à partir des positions initiales données, s'il existe des routes R_1 et R_2 à partir de ces positions qui garantissent le rendez-vous, pour n'importe quelle marche déterminée dans ces routes par l'adversaire.

Une caractéristique importante de nos algorithmes de rendez-vous est que, dans le choix des arêtes consécutives de sa route, un agent n'utilise pas la connaissance de la marche à ce jour. Ainsi la route ne dépend que du graphe et du point de départ choisi par l'adversaire (ainsi que des lancers de pièces de monnaie pour les algorithmes probabilistes), mais pas d'autres décisions de l'adversaire.

Finalement, nous désirons mentionner le problème de terminaison. Comme les agents sont identiques et qu'ils ne connaissent pas de bornes sur la taille du graphe (en fait, le graphe peut même être infini), ils ne sont pas capables de reconnaître quand le rendez-vous est impossible. Par exemple, les agents situés dans un anneau orienté, où le rendez-vous est impossible, ne peuvent pas distinguer cette situation de celle d'être dans un anneau orienté avec un noeud se distinguant par l'ajout d'une seule feuille comme voisin supplémentaire, avant de visiter ce noeud spécial. Dans ce dernier cas, le rendez-vous est possible. Ainsi, les agents ne sont jamais en mesure de dire que le rendez-vous est impossible dans la première situation et, dans ce cas, ils marchent indéfiniment sans se rencontrer. Nous allons montrer que lorsque le rendez-vous est possible, nos agents finissent toujours par se rencontrer et puis ils s'immobilisent.

5.2 Résultats

Dans le scénario déterministe, nous caractérisons les positions initiales des agents pour lesquels le rendez-vous est faisable. Il s'avère que cela est le cas si et seulement si les vues à partir des positions initiales sont différentes ou si ces positions sont reliées par un chemin dont la séquence correspondante de

numéros de ports est un palindrome. Nous fournissons un algorithme garantissant le rendez-vous asynchrone déterministe de toutes ces positions initiales dans un graphe arbitraire connexe. Dans le scénario aléatoire, nous montrons un algorithme qui réalise le rendez-vous asynchrone avec probabilité 1, pour les positions arbitraires initiales dans un graphe arbitraire connexe. Dans les deux cas, le graphe peut être fini, de taille arbitraire inconnue, ou infini dénombrable. Pour des raisons de simplicité, nous supposons que tous les degrés des noeuds sont finis, mais possiblement non-bornés.

Notre résultat dans le scénario probabiliste a les conséquences suivantes, peut-être surprenantes. Fixons une constante positive arbitraire ϵ. Nous montrons un algorithme qui garantit que deux agents asynchrones identiques équipés d'un compas, partant de positions arbitraires dans le plan, finiront par être à une distance au plus ϵ avec une probabilité de 1. Pour les agents synchrones (dans le cas du plan, cette restriction signifie que les deux agents se déplacent à une vitesse constante identique), un tel résultat découle du fait que la marche aléatoire dans un graphe infini à deux dimensions atteint un noeud de la grille avec une probabilité de 1 [29]. Notre algorithme permet d'accomplir une telle ϵ-approche avec probabilité 1 dans le cadre asynchrone (beaucoup plus difficile), c'est-à-dire quand chaque agent marche avec une vitesse arbitraire pouvant varier, et décidée par l'adversaire. Par ailleurs, notre algorithme fonctionne également dans des dimensions supérieures (par exemple, dans l'espace en 3 dimensions), tandis que la marche aléatoire dans la grille infinie de dimension > 2 ne peut pas être utilisée (il est bien connu [29] que la probabilité d'atteindre un noeud donné d'une telle grille par une marche aléatoire est strictement inférieure à 1). Au meilleur de notre connaissance, il n'existe pas de méthodes déjà connues pour accomplir une ϵ-approche des agents anonymes dans l'espace en 3 dimensions avec une probabilité de 1, même dans le scénario synchrone, tandis que notre algorithme permet de le faire dans le cadre asynchrone qui est beaucoup plus difficile.

5.3 Notions et résultats préliminaires

Nous utiliserons la notion suivante. Soit G un graphe et v le noeud de G, de degré k. La vue de v est un arbre $V(v)$ infini enraciné dans v avec

les ports étiquetés, définis récursivement comme suit. $V(v)$ a la racine x_0 correspondant à v. Pour chaque noeud $v_i, i = 1, \ldots, k$, adjacent à v, il y a un voisin x_i dans $V(v)$ tel que le numéro de port dans v correspondant à l'arête $\{v, v_i\}$ est le même que le numéro de port dans x_0 correspondant à l'arête $\{x_0, x_i\}$, et le numéro de port dans v_i correspondant à l'arête $\{v, v_i\}$ est le même que le numéro de port dans x_i correspondant à l'arête $\{x_0, x_i\}$. Le noeud x_i, pour $i = 1, \ldots, k$, est maintenant la racine de la vue de v_i.

Nos algorithmes sont basés sur la notion d'un *tunnel*, introduite dans [20] (voir Fig. 5.1). Considérons un graphe quelconque G et deux routes R_1 et R_2 débutant aux noeuds v et w, respectivement. Nous disons que ces routes forment un tunnel, s'il existe un préfixe $[e_1, e_2, \ldots, e_n]$ de la route R_1 et un préfixe $[e_n, e_{n-1}, \ldots, e_1]$ de la route R_2 pour $n \in \mathbb{N}$, pour certaines arêtes e_i dans le graphe, telles que $e_i = \{v_i, v_{i+1}\}$, où $v_1 = v$ et $v_{n+1} = w$. Intuitivement, la route R_1 a un préfixe P se terminant à w et la route R_2 a un préfixe inverse à P, se terminant à v. Par souci de simplicité, nous dirons également que les préfixes $[e_1, e_2, \ldots, e_n]$ et $[e_n, e_{n-1}, \ldots, e_1]$ forment un tunnel. La proposition qui suit a été prouvée dans [20].

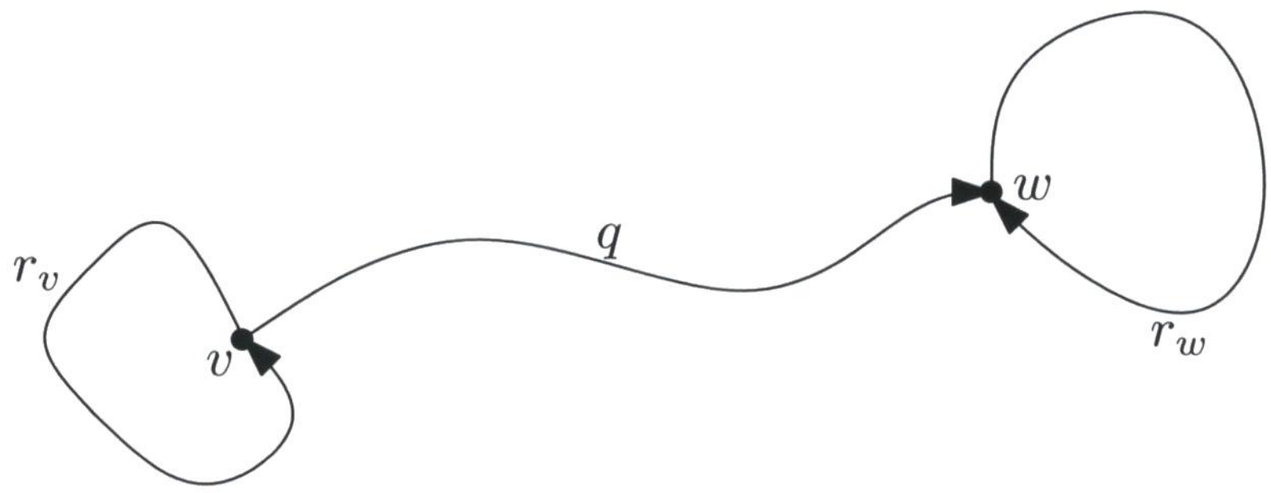

FIGURE 5.1 – Tunnel entre les routes de deux agents : notion introduite par [20].

Proposition 5.3.1 *Si les routes R_1 et R_2 forment un tunnel, alors elles garantissent le rendez-vous.*

Nous allons maintenant rappeler brièvement l'idée de l'algorithme de rendez-vous de [20], qui, par opposition à notre scénario, travaille pour les agents qui ont des étiquettes entières positives distinctes. Dans [20], les auteurs ont supposé que chaque agent connaît sa propre étiquette, mais pas celle de l'autre agent. Leur algorithme sera utilisé plus tard comme un bloc de construction pour notre algorithme de rendez-vous des agents anonymes.

Soit $G = (V, E)$ un graphe connexe dans lequel le rendez-vous doit être exécuté. Désignons par $\mathbb{N}$ l'ensemble des entiers positifs. Soit $\mathcal{S}$ l'ensemble de toutes les séquences finies de nombres entiers positifs. Soit $\mathcal{P} = \{(i, j, s', s'') | i, j \in \mathbb{N}, i < j \wedge s', s'' \in S\}$. Observons que l'ensemble $\mathcal{P}$ est dénombrable. Soit $(\varphi_1, \varphi_2, \dots)$ une énumération fixe de $\mathcal{P}$.

Pour un chemin fini r dans G, on dénote par $\overline{r}$ le chemin avec les mêmes arêtes que dans r, mais dans l'ordre inverse. Remarquons que r et $\overline{r}$ forment un tunnel. Considérons un chemin $r = (e_1, e_2, \dots, e_m)$ pour $m \in \mathbb{N}$, tel que $e_i = \{v_i, v_{i+1}\}$. Soit $s = (p_1, \dots, p_m)$, où p_i est le numéro de port du noeud v_i, correspondant à e_i. Nous disons que la séquence s de numéros de ports induit le chemin r.

L'algorithme de [20] force les routes de deux agents à former un tunnel pour toutes les combinaisons possibles des positions initiales et des étiquettes des deux agents. Par la proposition 5.3.1, cela suffit à garantir le rendez-vous. Toute configuration initiale de l'agent i placé au noeud v et de l'agent j placé au noeud w par l'adversaire correspond à un quadruple (i, j, s', s'') où s' est une séquence de ports induisant un chemin de v vers w et s'' est une séquence de ports induisant le chemin inverse de w vers v.

Chaque agent construit sa route en plusieurs phases. Au début et à la fin de chaque phase, l'agent est dans son noeud initial. Dans la phase k, la route partielle construite précédemment est étendue pendant que l'agent traite le quadruple φ_k (certaines extensions sont nulles). Cette extension garantit que les routes des agents de la configuration initiale correspondante formeront un tunnel après que les deux agents auront traité le quadruple φ_k. Lorsque l'agent avec l'étiquette l exécute le quadruple $\varphi_k = (i, j, s', s'')$, rien ne se

passe si $l \neq i$ et $l \neq j$. Si $l = i$, l'agent i essaie d'étendre sa route afin de garantir le rendez-vous avec l'agent j sous l'hypothèse qu'un chemin q de v vers w correspond à la séquence s' des ports et le chemin inverse $\overline{q}$ correspond à la séquence s''. Pour ce faire, l'agent tente d'abord de suivre le chemin induit par la séquence s' des ports. Cette tentative est considérée réussie si les conditions suivantes sont remplies :
• À chaque noeud consécutif du chemin parcouru, les ports dont les numéros correspondent à la séquence s' sont disponibles,
• Le chemin inverse correspond à la séquence s'' des ports.
Lorsque la tentative est réussie, l'agent est au noeud w et a déjà traversé la route $r_v \frown q$, où r_v est la route parcourue par celui-ci dans les $k-1$ premières phases. (Le symbole $\frown$ représente la concaténation.) Maintenant, il simule les $k-1$ premières phases de l'exécution de l'algorithme par l'agent avec l'étiquette j à partir du noeud w. L'effet de cette simulation est le chemin r_w. Il en résulte que l'agent a parcouru la route $r_v \frown q \frown r_w$ et est à nouveau sur w. Maintenant, l'agent retourne vers le noeud v en utilisant le chemin $\overline{q}$, il traverse $\overline{r_v}$ pour revenir à v, traverse q à nouveau allant ainsi à w, utilise $\overline{r_w}$ pour revenir à w et revient à v en passant par le chemin $\overline{q}$.

Si $l = j$, les actions ci-dessus sont effectuées avec les rôles de i et j inversés et les rôles de s' et s'' inversés également. Pour résumer, après que les deux agents avec des étiquettes i et j ont exécuté le quadruple $\varphi_k = (i, j, s', s'')$, où s' est la séquence des numéros de ports induisant q et s'' est la séquence des numéros de ports induisant $\overline{q}$, alors l'agent i traverse la route $\rho = r_v \frown q \frown r_w \frown \overline{q} \frown \overline{r_v} \frown q \frown \overline{r_w} \frown \overline{q}$, et l'agent j traverse la route $\rho' = r_w \frown \overline{q} \frown r_v \frown q \frown \overline{r_w} \frown \overline{q} \frown \overline{r_v} \frown q$. Par construction, la partie $r_v \frown q \frown r_w \frown \overline{q} \frown \overline{r_v} \frown q \frown \overline{r_w}$ de ρ et la partie $r_w \frown \overline{q} \frown r_v \frown q \frown \overline{r_w} \frown \overline{q} \frown \overline{r_v}$ de ρ' forment un tunnel, ce qui garantit le rendez-vous.

Pour construire le chemin inverse $\overline{q}$, $\overline{r_v}$ et $\overline{r_w}$, lorsque l'agent traverse l'un des chemins q, r_v, ou r_w, à chaque fois qu'il atteint un nouveau noeud, il mémorise le numéro de port d'entrée de l'arête d'où il arrive. Le chemin inverse respectif est obtenu en prenant la séquence de numéros de ports d'entrée dans l'ordre inverse.

Un chemin fini r dans G est appelé un *palindrome*, si r et $\overline{r}$ sont induits par la même séquence de numéros de ports. (Une condition équivalente est que la séquence de tous les ports rencontrés lors de la traversée de cette route

est identique à sa séquence inverse, c'est-à-dire cette séquence lue de la fin au début.)

5.4 Rendez-vous déterministe

Dans cette section, nous caractérisons les positions initiales des agents pour lesquels le rendez-vous déterministe est faisable. Plus précisément, nous montrons que ce sont des positions initiales pour lesquelles les vues sont différentes ou qui sont reliées par un chemin qui est un palindrome. Par ailleurs, nous fournissons un algorithme garantissant le rendez-vous asynchrone déterministe de toutes ces positions initiales dans un graphe connexe quelconque.

La difficulté principale dans la conception de l'algorithme est que, contrairement à [20], les agents n'ont pas d'étiquettes qui leur permettent de briser la symétrie. Toutefois, la symétrie peut être brisée par l'analyse des vues des agents, si ces vues sont différentes. Même quand elles sont différentes, les agents ne peuvent pas savoir à quels moments ces vues divergent et doivent donc les explorer pour trouver la première différence. Ainsi, l'algorithme procède par époques : à chaque époque consécutive de chaque agent, l'agent explore sa vue plus profondément, et crée un code de cette vue tronquée, en le traitant ultérieurement comme son étiquette temporaire et en appliquant la procédure décrite dans la section précédente pour une liste restreinte de quadruples. Si les vues sont différentes, un tunnel sera éventuellement créé après une certaine époque avec un indice suffisamment élevé. L'algorithme offre une fonctionnalité supplémentaire permettant la création d'un tunnel lorsque les agents ont des vues identiques, mais que leurs positions initiales sont reliées par un chemin qui est un palindrome. Ci-dessous, nous donnons une description détaillée de l'algorithme.

Algorithme Deterministe-RV

Nous présentons un algorithme pour un agent dont la position initiale est localisée sur le noeud v. (Le nom du noeud est à titre descriptif uniquement puisque les noeuds n'ont pas d'étiquettes). L'algorithme procède par époques

numérotées par des entiers consécutifs $1, 2, \ldots$. Au début et à la fin de chaque époque, l'agent est dans le noeud v. À l'époque n l'agent effectue tout d'abord une fouille en profondeur, de profondeur n, quittant un noeud visité par tous les ports dont le numéro est au plus n, dans l'ordre croissant. À la fin de la fouille, l'agent est de retour dans v. Maintenant, l'agent obtient le code $C(v, n)$ défini comme la séquence des numéros de ports qu'il a visités durant sa fouille sous la restriction que tous les numéros de ports supérieurs à n (par lequel l'agent peut avoir pénétré un noeud) sont remplacées par 0. Le code $C(v, n)$ est une suite de nombres entiers de l'ensemble $\{0, 1, \ldots, n\}$ de longueur au plus $4(n+1)^{n+1}$. Remarquez que si les vues $\mathcal{V}(v)$ et $\mathcal{V}(w)$ sont différentes, alors $C(v, n) \neq C(w, n)$ pour un certain n entier.

Considérons maintenant l'ensemble suivant de quadruples. Un quadruple (c_1, c_2, s_1, s_2) appartient à cet ensemble si c_1, c_2 sont des séquences de nombres $\{0, 1, \ldots, n\}$ de longueur au plus $4(n+1)^{n+1}$, s_1 et s_2 sont des séquences de nombres $1, 2, \ldots, n$ de longueur au plus n et soit $c_1 \neq c_2$, soit $c_1 = c_2$ et $s_1 = s_2$. La liste de tous les quadruples ordonnés par ordre lexicographique est notée par Q_n. Soit $\psi_{n,i}$ le i-ième quadruple dans la liste.

Après avoir obtenu le code $C = C(v, n)$, le reste de l'époque n est consacré à traiter les quadruples de la liste Q_n dans l'ordre et procède par étapes. L'étape i est consacrée au traitement du quadruple $\psi_{n,i} = (c_1, c_2, s_1, s_2)$. Au début et à la fin de chaque étape, l'agent est sur le noeud v. Si C est différent de c_1 et de c_2, rien ne se passe à l'étape i. Si $C = c_1$, l'agent traite le quadruple $\psi_{n,i}$ de la même manière que cela a été fait pour le quadruple φ_k dans la section précédente. Soit r_v la route parcourue par l'agent dans toutes les époques antérieures et dans la partie précédente de l'époque actuelle. La route r_v commence et se termine sur le noeud v. L'agent essaie d'étendre la route r_v pour garantir le rendez-vous avec l'autre agent sous l'hypothèse que :
(a) cet agent se situe sur le noeud w,
(b) $c_2 = C(w, n)$,
(c) un chemin q de v vers w est induit par la séquence de ports s_1,
(d) le chemin inverse $\overline{q}$ est induit par la séquence de ports s_2.
L'agent essaie d'abord de suivre le chemin induit par la séquence de ports s_1. Comme précédemment, cette tentative est considérée réussie si :
(1) au niveau des noeuds consécutifs du chemin parcouru, les ports avec les numéros de la séquence s_1 sont disponibles,

(2) le chemin inverse correspond à la séquences s_2.
Lorsque la tentative est réussie, l'agent est sur le noeud w et a déjà traversé la route $r_v \frown q$. Il simule ensuite les époques précédentes et la partie antérieure de l'époque actuelle de l'exécution de l'algorithme par un agent avec le code $c_2 = C(w, n)$ à partir de w. Remarquez que, pour réaliser cette simulation, il est nécessaire de calculer tous les codes de $C(w, m)$ pour $m < n$, où $c_2 = C(w, n)$, car un agent avec le code c_2 durant l'époque n aurait le code $C(w, m)$ durant une époque $m < n$. Cependant, tous les codes $C(w, m)$ pour $m < n$ peuvent être déduits à partir de $C(w, n)$. Cela se fait comme suit. À partir du code $C(w, n)$ reconstruire l'arbre T_n correspondant à la fouille en profondeur restreinte en fonction de la profondeur n. Maintenant, construisons un sous-arbre T_m de T_n, en coupant tous les sous-arbres enracinés aux enfants u du noeud z de sorte que le port dans z qui correspond à l'arête $\{z, u\}$ est plus grand que m. Dans cet arbre élagué, remplacer tous les numéros de ports supérieurs à m par 0. Maintenant, le code $C(w, m)$ est la séquence des numéros de ports rencontrés lors de la visite de T_m en profondeur d'abord, dans l'ordre croissant des numéros de ports.

L'effet de cette simulation est le chemin r_w. Il en résulte que l'agent traverse la route $r_v \frown q \frown r_w$ et est à nouveau sur w. Maintenant, l'agent remonte à v en utilisant le chemin $\overline{q}$, il parcourt $\overline{r_v}$ revenant à v, traverse q à nouveau pour atteindre w, utilise $\overline{r_w}$ revenant à w et se rétracte à v en utilisant le chemin $\overline{q}$.

Si $C = c_2$, les actions ci-dessus sont effectuées avec les rôles de c_1 et c_2 inversés et les rôles de s_1 et s_2 inversés. Notez que dans notre algorithme, le quadruple traité peut être de la forme (c, c, s_1, s_2) (qui ne pourrait pas se produire dans l'algorithme de [11], où les étiquettes sont toujours assumées comme étant différentes). Dans ce cas, il n'y a pas d'ambiguïté en ce qui concerne les actions de l'agent, car alors l'égalité $s_1 = s_2$ est satisfaite par la définition des quadruples.

Après avoir terminé toutes les étapes de l'époque n, l'agent commence l'époque $n+1$, jusqu'à ce qu'il y ait rendez-vous ou indéfiniment, si le rendez-vous est impossible.

Lemme 5.4.1 *Si les vues des positions initiales des agents sont différentes ou si les positions initiales sont reliées par un chemin qui est un palindrome,*

l'algorithme déterministe-RV garantit le rendez-vous asynchrone.

Preuve: Supposons d'abord que les vues $\mathcal{V}(v)$ et $\mathcal{V}(w)$ des positions initiales v et w des agents sont différentes. Considérons le plus petit n, tel que $C(v,n) \neq C(w,n)$ et il existe un chemin q de longueur au plus n entre v et w, tel que tous les numéros de ports sur q sont au plus n. Soit $c_1 = C(v,n)$ et $c_2 = C(w,n)$. Alors $c_1 \neq c_2$. Soit s_1 la séquence des numéros de ports induisant q et s_2 la séquence des numéros de ports induisant $\overline{q}$. Ainsi, le quadruple (c_1, c_2, s_1, s_2) apparaît dans la liste Q_n. Supposons que $\psi_{n,i} = (c_1, c_2, s_1, s_2)$. Nous montrons qu'avant la fin de l'étape i de l'époque n par les deux agents, un tunnel est formé et, par conséquent, le rendez-vous asynchrone est garanti.

Soit r_v (resp. r_w) la route de l'agent qui commence à v (resp. de l'agent qui commence à w) après avoir terminé toutes les époques $1, \ldots, n-1$ et toutes les étapes de $1, \ldots, i-1$ de l'époque n. A l'issue de l'étape i de l'époque n, l'agent débutant sur v a traversé la route $\rho = r_v \frown q \frown r_w \frown \overline{q} \frown \overline{r_v} \frown q \frown \overline{r_w} \frown \overline{q}$ et l'agent débutant sur w a parcouru la route $\rho' = r_w \frown \overline{q} \frown r_v \frown q \frown \overline{r_w} \frown \overline{q} \frown \overline{r_v} \frown q$. Par construction, la partie $r_v \frown q \frown r_w \frown \overline{q} \frown \overline{r_v} \frown q \frown \overline{r_w}$ de ρ et la partie $r_w \frown \overline{q} \frown r_v \frown q \frown \overline{r_w} \frown \overline{q} \frown \overline{r_v}$ de ρ' forment un tunnel.

Supposons maintenant que les vues $\mathcal{V}(v)$ et $\mathcal{V}(w)$ des positions initiales v et w des agents sont identiques, mais ces positions initiales sont reliées par un chemin q qui est un palindrome. Soit s est la séquence des numéros de ports qui induit ce chemin et qui induit le chemin inverse $\overline{q}$. Soit n le plus grand des deux nombres entiers : la longueur du chemin q et le plus grand terme de la séquence s. Soit $c = C(v,n) = C(w,n)$. Ainsi, le quadruple (c, c, s, s) apparaît dans la liste Q_n. Supposons que $\psi_{n,j} = (c, c, s, s)$. Nous montrons qu'à l'issue de l'étape j de l'époque n par les deux agents, un tunnel est formé et, par conséquent, le rendez-vous asynchrone est garanti.

Soit r_v (resp. r_w) la route de l'agent qui commence à v (resp. de l'agent qui commence à w) après avoir terminé toutes les époques $1, \ldots, n-1$ et toutes les étapes de $1, \ldots, j-1$ de l'époque n. À l'issue de l'étape j de l'époque n, un agent a traversé la route $r_v \frown q \frown r_w \frown \overline{q} \frown \overline{r_v} \frown q \frown \overline{r_w} \frown \overline{q}$ et l'autre agent a parcouru la route $r_w \frown \overline{q} \frown r_v \frown q \frown \overline{r_w} \frown \overline{q} \frown \overline{r_v} \frown q$. Comme précédemment, ces routes forment un tunnel. □

Le lemme suivant montre que si les positions initiales des agents ne satisfont pas la condition du lemme 5.4.1, le rendez-vous asynchrone ne peut pas être garanti.

Lemme 5.4.2 *Si les vues à partir des positions initiales des agents sont identiques et les positions initiales ne sont pas connectées par un chemin qui est un palindrome, alors le rendez-vous asynchrone ne peut pas être garanti.*

Preuve: Considérons deux agents débutant dans les noeuds v et w, tels que $\mathcal{V}(v) = \mathcal{V}(w)$. Considérons un algorithme arbitraire garantissant le rendez-vous asynchrone. Il produit les routes $R(v)$ et $R(w)$, respectivement. Considérons un adversaire qui déplace ces agents le long de leurs routes à vitesse constante identique. Une rencontre des agents pourrait se produire la première fois qu'ils se rendent sur un certain noeud u en même temps, ou quand ils traversent la même arête e dans le sens opposé simultanément (dans ce cas, le rendez-vous aurait lieu au milieu de cette arête). La première situation est impossible parce que cela impliquerait que deux arêtes distinctes incidentes à u ont le même numéro de ports sur u (compte tenu de $\mathcal{V}(v) = \mathcal{V}(w)$). Par conséquent, on peut supposer que la seconde situation se produit. Soit $\pi = (e_1, \ldots, e_k)$ et $\pi' = (e'_1, \ldots, e'_k)$ les parties des routes $R(v)$ et $R(w)$, respectivement, avant que les agents entrent dans l'arête e. Compte tenu de $\mathcal{V}(v) = \mathcal{V}(w)$, la séquence de ports rencontrées par les agents quand ils traversent cette partie de leurs routes est la même et les ports aux deux extrémités de l'arête e sont identiques. Cela implique que le chemin $\pi \frown \{e\} \frown \overline{\pi'}$ reliant v et w est un palindrome. □

Les lemmes 5.4.1 et 5.4.2 impliquent le résultat principal concernant le rendez-vous asynchrone déterministe pour les agents anonymes.

Théorème 5.4.1 *Le rendez-vous asynchrone déterministe des agents anonymes est réalisable si et seulement si les vues à partir des positions initiales des agents sont différentes ou si les positions initiales sont reliées par un chemin qui est un palindrome. Si cette condition est satisfaite, l'algorithme déterministe-RV garantit le rendez-vous asynchrone des agents.*

5.5 Rendez-vous probabiliste

Dans cette section, nous montrons un algorithme aléatoire qui permet d'obtenir le rendez-vous asynchrone avec une probabilité de 1, pour les positions initiales arbitraires dans un graphe arbitraire connexe.

Algorithme Aléatoire-RV Nous présentons l'algorithme pour un agent dont la position initiale est sur le noeud v (Comme avant, le nom du noeud est pour des fins descriptives uniquement). L'algorithme procède par époques numérotées par des entiers consécutifs $1, 2, \ldots$. Au début et à la fin de chaque époque, l'agent est localisé sur le noeud v. Au début de l'époque n, l'agent a un code $C(v, n-1)$ qui est une séquence binaire de longueur $n-1$. L'agent commence l'époque n en choisissant un bit au hasard avec une probabilité $\frac{1}{2}$ et il l'ajoute à $C(v, n-1)$, formant ainsi le code $C(v, n)$.

Considérons maintenant l'ensemble suivant de quadruples. Un quadruple (c_1, c_2, s_1, s_2) appartient à cet ensemble si c_1 et c_2 sont des séquences binaires différentes de longueur n et s_1 et s_2 sont des séquences de nombres $1, 2, \ldots, n$ de longueur au plus n. La liste de tous ces quadruples ordonnés lexicographiquement est notée par P_n. Soit $\lambda_{n,i}$ le i-ième quadruple dans la liste.

Après avoir obtenu le code $C = C(v, n)$, le reste de l'époque n est consacré au traitement des quadruples de la liste P_n dans l'ordre, et en procédant par étape. L'étape i est consacrée au traitement du quadruple $\lambda_{n,i} = (c_1, c_2, s_1, s_2)$. Au début et à la fin de chaque étape, l'agent est sur le noeud v. Si C est différent de c_1 et de c_2, rien ne se passe à l'étape i. Si $C = c_1$ ou $C = c_2$, l'agent traite le quadruple $\lambda_{n,i}$ de la même manière que cela a été fait pour le quadruple $\psi_{n,i}$ dans l'algorithme déterministe-RV. Notez que dans notre algorithme actuel (par opposition à l'algorithme déterministe-RV) tous les quadruples traités satisfont $c_1 \neq c_2$.

Après avoir terminé toutes les étapes de l'époque n, l'agent commence l'époque $n+1$, jusqu'à ce qu'il y ait rendez-vous ou indéfiniment.

Théorème 5.5.1 *L'algorithme aléatoire-RV garantit le rendez-vous asynchrone avec une probabilité de 1, pour des positions initiales arbitraires des agents dans un graphe arbitraire connexe.*

Preuve: Considérons les agents situés dans les noeuds v et w. Observons d'abord que la probabilité de l'événement que pour un certain entier m les codes $C(v,m)$ et $C(w,m)$ soient différents, est 1. Par conséquent, il suffit de prouver que, à un certain moment de l'exécution de l'algorithme, les routes des agents forment un tunnel, sous l'hypothèse que $C(v,m) \neq C(w,m)$ pour un certain m.

Soit p le plus petit entier pour lequel $C(v,p) \neq C(w,p)$. Soit $n \geq p$ le plus petit entier pour lequel il existe un chemin q de longueur au plus n qui joint v et w, tel que tous les ports sur le chemin q ont des nombres au plus de grandeur n. Notons $c_1 = C(v,n)$ et $c_2 = C(w,n)$. Soit s_1 la séquence des numéros de ports induisant q et s_2 la séquence des numéros de ports induisant $\overline{q}$. Ainsi, le quadruple (c_1, c_2, s_1, s_2) apparaît dans la liste P_n. Supposons que $\lambda_{n,i} = (c_1, c_2, s_1, s_2)$. Le même argument que dans la preuve du lemme 5.4.1 montre qu'à la fin de l'étape i de l'époque n par les deux agents, un tunnel est formé. □

Nous terminons cette section en présentant une application de l'algorithme ci-dessus au problème de rendez-vous asynchrone approximatif d'agents anonymes dans le plan. Deux agents (modélisés comme des points mobiles) équipés de boussoles, débutent dans des positions initiales arbitraires dans le plan. Pour une constante fixe $\epsilon > 0$ connue des agents, le ϵ-rendez-vous consiste à amener les agents à distance au plus ϵ.

Ce problème peut être résolu avec une probabilité 1, en appliquant l'algorithme aléatoire-RV présenté ci-dessus comme suit. Chaque agent exécute cet algorithme sur une ϵ-grille rectangulaire dont l'un des noeuds est la position initiale de l'agent. Plus précisément, les voisins d'un noeud donné dans chaque grille sont les quatre points Nord, Est, Sud et Ouest du point donné, à une distance ϵ. Les numéros de ports correspondant à ces arêtes à chaque noeud de la grille sont étiquetés $1, 2, 3, 4$. Étant donné que chaque agent a une boussole, il peut déterminer le numéro de ports à chaque noeud visité de la grille.

Soit v et w les points de départ des agents et G_v et G_w les grilles respectives des agents. Désignons par w' le noeud de la grille G_v le plus proche de w (prendre n'importe quel noeud avec cette propriété, s'il y en a plus qu'un seul). La distance entre w et w' est inférieure à ϵ. Simuler les mouvements

de l'agent à partir de w en faisant un déplacement parallèle par le vecteur $[w, w']$. Les mouvements simulés sont sur la grille G_v débutant sur w'. D'après le théorème 5.5.1, l'agent débutant sur v et l'agent (virtuel) débutant sur w' se rencontreront avec une probabilité de 1 sur la grille G_v. Au moment de leur rencontre, l'agent (réel) débutant sur w sera à une distance inférieure à ϵ de l'agent débutant sur v. Ainsi le ϵ-rendez-vous des agents à partir de v et w s'accomplira (avec une probabilité de 1).

Il est intéressant de comparer cette application de l'algorithme aléatoire-RV à ce qui peut être fait en utilisant d'autres méthodes. Pour les agents synchrones (par exemple, les agents se déplaçant dans le plan à vitesse constante identique), le ϵ-rendez-vous dans le plan avec une probabilité de 1 découle du fait qu'une marche aléatoire sur une grille infinie à deux dimensions rencontre un noeud donné de la grille avec une probabilité 1 [29]. Notre algorithme permet d'accomplir le ϵ-rendez-vous dans le plan avec une probabilité 1 dans le cadre asynchrone beaucoup plus difficile.

Plus important encore, notre algorithme fonctionne également dans des dimensions plus élevées (par exemple, dans l'espace à 3 dimensions pour les agents ayant une "Boussole tridimensionnelle" qui peut établir des orientations Nord, Est, Sud, Ouest, Haut et Bas), tandis que la marche aléatoire dans une grille infinie de dimension > 2 ne peut pas être utilisée (il est bien connu [14] que d'atteindre un noeud donné d'une telle grille par une marche aléatoire se produit avec une probabilité strictement inférieure à 1).

Chapitre 6

Conclusion

Nous avons étudié le problème de rendez-vous pour plusieurs modèles d'agents mobiles asynchrones dans les réseaux. Dans les chapitres 3 et 4, il s'agissait des agents sans mémoire qui opèrent en cycles asynchrones Observation - Calcul - Déplacement. Ce modèle est dérivé du modèle CORDA. Notre modification de ce modèle consiste en restriction des champs de vision des agents. Dans le chapitre 3, nous avons imposé la restriction la plus sévère : le rayon de vision des agents était égal à 1. Dans ce modèle très faible nous avons prouvé qu'il y a très peu de configurations regroupables et nous avons proposé un algorithme universel de rendez-vous. Le résultat est valide pour les graphes bipartis réguliers et il reste un problème ouvert s'il y a un algorithme universel de rendez-vous pour toute la classe des graphes réguliers.

Étant donné le petit nombre de configurations regroupables pour le rayon de vision $d = 1$, dans le chapitre 4 nous avons renforcé le modèle considérant le rayon de vision fini mais arbitraire. Dans ce cas, la classe des configurations regroupables est beaucoup plus importante même pour la ligne infinie, mais nous avons prouvé qu'il n'y a pas d'algorithme universel pour cette ligne pour aucun rayon de vision $d > 1$. Il reste ouvert de généraliser ce résultat pour d'autre classes de graphes et de voir s'il existe des algorithmes universels pour certaines classes naturelles de configurations initiales, même dans le cas de la ligne infinie. L'exemple d'une telle classe de configurations serait la classe des segments connexes.

Un autre problème ouvert concernant le modèle dérivé du modèle CORDA consisterait à considérer les changements provoqués par l'ajout d'une petite quantité de mémoire aux agents. Une question intéressante est comment la classe des configurations regroupables changerait si le comportement des agents dépendait non seulement de la capture durant la dernière phase d'observation mais de plusieurs phases d'observations précédentes.

Finalement, dans le chapitre 5, nous avons étudié le rendez-vous pour les agents qui n'ont aucun champ de vision, mais qui disposent d'une mémoire illimitée. Dans ce cas l'asynchronie est modélisée par un adversaire qui contrôle la vitesse des agents. Pour des graphes arbitraires connexes, nous avons caractérisé les positions initiales des agents pour lesquelles le rendez-vous déterministe est possible et nous avons présenté un algorithme de rendez-vous pour ces positions. Il reste ouvert de généraliser le problème du rendez-vous dans ce modèle pour un nombre plus grand des agents. Il serait aussi intéressant de voir si les techniques de [26] permettant un rendez-vous asynchrone en coût polynomial pour les agents étiquetés peuvent être appliquées pour le rendez-vous des agents anonymes.

Bibliographie

[1] Abbas, S., Mosbah, M., and Zemmari, A. Merging time of random mobile agents. *Proc. 1st International Conference on Dynamics in Logistics (LDIC)* (2007), 179–189.

[2] Alpern, S. Rendezvous search : A personal perspective. *Operations Research, vol. 50* (2002), 772 – 795.

[3] Alpern, S., and Gal, S. The theory of search games and rendezvous. *International Series in Operations Research and Management Science* (2003).

[4] Anderegg, L., Cieliebak, M., and Prencipe, G. A linear time algorithm to identify equiangular and biangular point configurations. *Technical Report TR-03-01. Universita di Pisa* (2003).

[5] Awerbuch, B., Betke, M., Rivest, R. L., and Singh, M. Piecemeal graph exploration by a mobile robot. *Inf. Comput. vol. 152* (1999), 155–172.

[6] Awerbuch, B., and Kobourov, S. Polylogarithmic-overhead piecemeal graph exploration. *Proc. 11th Annual Conference on Computational Learning Theory (COLT)* (1998), 280–286.

[7] Baba, D., Izumi, F., Ooshita, F., Kakugawa, H., and Masuzawa, T. Linear time and space gathering of anonymous mobile agents in asynchronous trees. *Theor. Comput. Sci. vol. 478* (2013), 118–126.

[8] Bajaj, C. The algebraic degree of geometric optimization problems. *Discrete and Computational Geometry vol. 3* (1988), 177–191.

[9] Bampas, E., Czyzowicz, J., Gasieniec, L., Ilcinkas, D., and Labourel, A. Almost optimal asynchronous rendezvous in infinite multidimensional grids. *Proc. 24th International Symposium on Distributed Computing (DISC)* (2010), 297–311.

[10] Bender, M. A., Slonim D. The power of team exploration : Two robots can learn unlabeled directed graphs. *Proc. 35th Annual Symposium on Foundations of Computer Sciences (FOCS)* (1994), 75–85.

[11] Berman, P., Blum, A., Fiat, A., Karloff, H., Rosen, A., and Saks, M. Randomized robot navigation algorithm. *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (1996), 75–84.

[12] Cieliebak, M. Gathering non-oblivious mobile robots. *Proc. 6th Latin American Symposium on Theoretical Informatics (LATIN)* (2004), 577–588.

[13] Cieliebak, M., Flocchini, P., Prencipe, G., and Santoro, N. Distributed Computing by Mobile Robots : Gathering. *SIAM J. Comput. vol. 41* (2012), 829–879.

[14] Cohen, R., and Peleg, D. Robot convergence via center-of-gravity algorithms. *Proc. 11th International Colloquium on Structural Information and Communication Complexity (SIROCCO)* (2004), 79–88.

[15] Collins, A., Czyzowicz, J., Gasieniec, L., and Labourel, A. Tell me where I am so I can meet you sooner. *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP)* (2010), 502–514.

[16] Coppersmith, D., Tetali, P., and Winkler, P. Collisions among random walks on a graph. *SIAM J. Discrete Math. vol. 6* (1993), 363–374.

[17] Czyzowicz, J., Dobrev, S., Kranakis, E., and Krizanc, D. The power of tokens : Rendezvous and symmetry detection for two mobile agents in a ring. *Proc. 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)* (2008), 234–246.

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

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

[20] Czyzowicz, J., Labourel, A., and Pelc, A. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms vol. 8* (2012), 37.

[21] D'ANGELO, G., DI STEFANO, G., AND NAVARRA, A. Gathering of six robots on anonymous symmetric rings. *Proc. 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO)* (2011), 174 – 185.

[22] D'ANGELO, G., DI STEFANO, G., AND NAVARRA, A. How to gather asynchronous oblivious robots on anonymous rings. *Proc. 26th International Symposium on Distributed Computing (DISC)* (2012), 330–344.

[23] DATTA, A. K., LAMANI, A., LAMORE, L. L., AND PETIT, F. Ring exploration with oblivious myopic robots. *Proc. 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO)* (2011), 174 – 185.

[24] DE MARCO, G., GARGANO, L., KRANAKIS, E., KRIZANC, D., PELC, A., AND VACCARO, U. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci. vol. 355* (2006), 315–326.

[25] DESSMARK, A., FRAIGNIAUD, P., KOWALSKI, D., AND PELC, A. Deterministic rendezvous in graphs. *Algorithmica vol. 46* (2006), 69–96.

[26] DIEUDONNÉ, Y., PELC, A., AND VILLAIN, V. How to meet asynchronously at polynomial cost. *Proc. 32th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)* (2013), 92–99.

[27] DOBREV, S., FLOCCHINI, P., PRENCIPE, N., AND SANTORO, N. Searching for a black hole in arbitrary networks : optimal mobile agents protocols *Distributed Computing vol.19* (2006), 1–18.

[28] DUNCAN, C. A., KOBOUROV, S. G., AND ANIL KUMAR, V. S. Optimal constrained graph exploration. *ACM Transactions on Algorithms vol. 2* (2006), 380–402.

[29] FELLER, W. An introduction to probability theory and its applications. *Wiley* (1968).

[30] FLOCCHINI, P., ILCINKAS, D., PELC, A., AND SANTORO, N. Computing without communicating : Ring exploration by asynchronous oblivious robots. *Algorithmica vol. 65* (2013), 562–583.

[31] FLOCCHINI, P., ILCINKAS, D., PELC, A., AND SANTORO, N. Remembering without memory : Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci. vol. 411* (2007), 1583–1598.

[32] Flocchini, P., Kranakis, D., Krizanc, N., Santoro, N., and Sawchuk, C. Multiple mobile agent rendezvous in a ring. *Proc. 6th Latin American Symposium on Theoretical Informatics (LATIN)* (2004), 599–608.

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

[34] Fraigniaud, P., and Pelc, A. Delays induce an exponential memory gap for rendezvous in trees. *ACM Transactions on Algorithms vol. 9* (2013), 17

[35] Guilbault, S., and Pelc, A. Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. *Theor. Comput. Sci. vol. 509* (2013), 86–96.

[36] Guilbault, S., and Pelc, A. Asynchronous rendezvous of anonymous agents in arbitrary graphs. *Proc. 15th International Conference on Principles of Distributed Systems (OPODIS)* (2011), 421–434.

[37] Guilbault, S., and Pelc, A. Gathering asynchronous oblivious agents with restricted vision in an infinite line. *Proc. 15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)* (2013), to appear.

[38] Izumi, T., Izumi, T., Kamei, S., and Ooshita, F. Time-optimal gathering algorithm of mobile robots with local weak multiplicity detection in rings. *IEICE Transactions vol. 96* (2013), 1072–1080.

[39] Klasing, R., Kosowski, A., and Navarra, A. Taking advantage of symmetries : Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci. vol. 411* (2010), 3235–3246.

[40] Klasing, R., Markou, E., and Pelc, A. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci. vol. 390* (2008), 27–39.

[41] Koucky, M. Universal traversal sequences with backtracking. *J. Comput. Syst. Sci. vol. 65* (2002), 717–726.

[42] Kowalski, D., and Malinowski, A. How to meet in anonymous network. *Theor. Comput. Sci. vol. 399* (2008), 141–156.

[43] Kranakis, E., Krizanc, D., and Markou, E. Mobile agent rendezvous in a synchronous torus. *Proc. 7th Latin American Symposium on Theoretical Informatics (LATIN)* (2006), 141–156.

[44] KRANAKIS, E., KRIZANC, D., AND MARKOU, E. The mobile agent rendezvous problem in the ring. *Morgan and Claypool Publishers* (2010).

[45] KRANAKIS, E., KRIZANC, D., AND MORIN, P. Randomized rendezvous with limited memory. *ACM Transactions on Algorithms vol. 7* (2011), 34.

[46] KRANAKIS, E., KRIZANC, D., AND RAJSBAUM, S. Mobile agent rendezvous : A survey. In *SIROCCO* (2006), P. Flocchini and L. Gasieniec, Eds., vol. 4056 of *Lecture Notes in Computer Science*, Springer, 1–9.

[47] KRANAKIS, E., KRIZANC, D., SANTORO, N., AND SAWCHUK, C. Mobile agent rendezvous in a ring. *Proc. 23rd International Conference on Distributed Computing Systems (ICDCS)* (2003), 592–599.

[48] KUPITZ, Y., AND MARTINI, H. Geometric aspects of the generalized fermat-torricelli problem. *Intuitive Geometry vol.6* (1997), 55–127.

[49] MARKOU, E., AND PELC, A. Efficient exploration of faulty trees. *Theory Comput. Systems vol. 40* (2007), 225–247.

[50] PRENCIPE, G. Corda : Distributed coordination of a set of autonomous mobile robots. *Proc. 21th European Research Seminar on Advances in Distributed Systems (ERSADS)* (2001), 185–190.

[51] PRENCIPE, G. Impossibility of gathering by a set of autonomous mobile robots. *Theor. Comput. Sci. vol. 384* (2007), 222–231.

[52] SCHELLING, T. The strategy of conflict. *Oxford University Press, Oxford* (1960).

[53] SUZUKI, I., AND YAMASHITA, M. Distributed anonymous mobile robots : Formation of geometric patterns. *SIAM J. Comput. vol.28* (1999), 1347–1363.

[54] TA-SHMA, A., AND ZWICK, U. Deterministic rendezvous, treasure hunts and strongly universal exploration sequences. *Proc. 18st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2007), 599–608.

[55] WEISZFELD, E. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical vol.43* (1936), 355–386.