Algorithms for deterministic parallel graph exploration

Dominik Pajak

université de BORDEAUX

THÈSE

présentée pour obtenir la grade de

DOCTEUR

L'UNIVERSITÉ BORDEAUX

École Doctorale de Mathémathiques et Informatique de Bordeaux

SPÉCIALITÉ : **INFORMATIQUE**

par

Dominik PAJĄK

Algorithms for Deterministic Parallel Graph Exploration

Soutenue le 13 Juin 2014 au Laboratoire Bordelais de Recherche en Informatique (LaBRI)

Aprés avis de rapporteurs:

Pierre Fraigniaud	Directeur de Recherche CNRS à Université Paris Diderot (France)
Tomasz Radzik	Reader à King's College London (Angleterre)
Peter Widmayer	Professeur à ETH Zürich (Suisse)

Devant la commission d'examen composée de :

Examinateurs	
Philippe Duchon	Professeur à l'Université Bordeaux (France)
Rapporteurs	
Pierre Fraigniaud	Directeur de Recherche CNRS à Université Paris Diderot (France)
Tomasz Radzik	Reader à King's College London (Angleterre)
Peter Widmayer	Professeur à ETH Zürich (Suisse)
Directeurs de thèse	
Ralf Klasing	Directeur de Recherche CNRS à Université Bordeaux (France)
Adrian Kosowski	Chargé de Recherche Inria à Université Paris Diderot (France)

Acknowledgements

First of all, I would like to thank my PhD advisors Ralf Klasing and Adrian Kosowski for their patience, friendship and support.

The work included in this thesis was performed during my 3-year PhD studies at the LaBRI in Bordeaux. Additional results, some of which are not included in the thesis, are the product of research visits to Perugia, Liverpool, Ottawa and Wroclaw. I would like to thank all my collaborators and co-authors. I had a pleasure working with (in alphabetical order): Evangelos Bampas, Jurek Czyzowicz, Yann Disser, Leszek Gąsieniec, Thomas Gorry, Marcin Kardas, Ralf Klasing, Marek Klonowski, Adrian Kosowski, Evangelos Kranakis, Russell Martin, Alfredo Navarra, Eduardo Pacheco, Cristina Pinotti, Thomas Sauerwald, Najmeh Taleb, Przemysław Uznański and Kamil Wolny.

Thanks to my friends, my stay in Bordeaux was very enjoyable. My thanks go to Abbas, Agnieszka, Anna, Andjela, Carlos, Cedric, Charlotte, Clement, Elham, Farhad, Florent, Ikram, Ipsita, Jesus, Juan Angel, Lorenzo, Lorijn, Maxime, Meropi, Petru, Przemysław, Sagnik, Sri and Zuzanna. I would like to especially thank Zuzanna and Agnieszka for preparing the reception after my defence. I am very grateful to Jakub who helped me with translating the abstract of the thesis to French.

I am particularly grateful to the referees Pierre Fraigniaud, Tomasz Radzik and Peter Widmayer for agreeing to take on their commitment and for reading this manuscript. Some of their comments have already been taken into account when preparing the final version of the thesis.

I gratefully acknowledge the financial support received from Inria (through the project teams CEPAGE and RealOpt in Bordeaux), from l'Agence Nationale de la Recherche (projects ALADDIN and DISPLEXITY).

Finally I would like to thank my family: my father Andrzej, my mother Krystyna, my brother Krzystof and my sister Karolina for their support and encouragement.

Résumé

Nous étudions dans cette thèse le problème de l'exploration parallèle d'un graphe à l'aide des multiples, synchronisés et mobiles agents. Chaque agent est une entité individuelle qui peut, indépendamment des autres agents, visitez les sommets du graphe ou parcourir ses arêtes. Le but de ensemble des agents est de visiter tous les sommets de graphe.

Nous étudions d'abord l'exploration du graphe dans un modèle où chaque agent est équipé de mémoire interne, mais les nœuds n'ont pas de mémoire. Dans ce modèle les agents sont autorisés à communiquer entre eux en échangeant des messages. Nous présentons des algorithmes qui s'exécutent dans un minimum de temps possible pour polynomiale nombre d'agents (polynomiale en nombre de sommets du graphe). Nous étudions aussi quelle est l'impacte de différent méthodes des communications. Nous étudions des algorithmes où les agents peuvent se communiquer à distance arbitraire, mais aussi où communication est possible seulement entre les agents situés dans le même sommet. Dans les deux cas nous présentons des algorithmes efficaces. Nous avons aussi obtenu des limites inférieures qui correspondent bien à la performance des algorithmes.

Nous considérons également l'exploration de graphe en supposant que les mouvements des agents sont déterminés par le soi-disant rotor-router mécanisme. Du point de vue d'un sommet fixé, le rotor- router envoie des agents qui visitent les sommet voisins dans un mode round-robin. Nous étudions l'accélération défini comme la proportion entre le pire des cas de l'exploration d'un agent unique et des plusieurs agents. Pour générales graphes, nous montrerons que le gain de vitesse en cas de multi-agent rotor-router est toujours entre fonction logarithmique et linéaire du nombre d'agents. Nous présentons également des résultats optimaux sur l'accélération de multi-agent rotor-router pour cycles, expanseurs, graphes aléatoires, cliques, tores de dimension fixé et une analyse presque optimale pour hypercubes.

Finalement nous considérons l'exploration sans collision, où chaque agent doit explorer le graphe de manière indépendante avec la contrainte supplémentaire que deux agents ne peuvent pas occuper le même sommet. Dans le cas où les agents sont donnés le plan de graphe, on présente un algorithme optimal pour les arbres et un algorithme asymptotiquement optimal pour générales graphes. Nous présentons aussi des algorithmes dans le cas de l'exploration sans collision des arbres et des générales graphes dans la situation où les agents ne connaissent pas le graphe.

Nous fermons la thèse par des observations finales et une discussion de problèmes ouverts liés dans le domaine de l'exploration des graphes.

Mots clés: exploration de graphes, équipe d'agents mobiles, algorithme, marche déterministe, rotor-router modèle, marches aléatoires parallèles, exploration collaborative, algorithme d'apprentissage incrémental

Abstract

In this thesis we study the problem of parallel graph exploration using multiple synchronized mobile agents. Each mobile agent is an entity that can, independently of other agents, visit vertices of the graph and traverse its edges. The goal of the agents is to visit all vertices of the graph.

We first study graph exploration in the model where agents are equipped with internal memory but no memory is available at the nodes. Agents in this model are also allowed to communicate between each other by exchanging messages. We present algorithms working in a minimal possible time for a team of polynomial size (in the number of vertices of the graph). We also study the impact of the available range of communication by analysing algorithms for agents which can communicate at arbitrary distance, or only with other agents located at the same node. We present efficient algorithms and lower bounds that almost match our positive results in both communication models.

We also consider graph exploration when movements of agents are determined according to the so-called *rotor-router* mechanism. From the perspective of a fixed node, the rotor-router sends out agents which visit the node along its outgoing edges, in a round-robin fashion. We study the speedup which is the ratio between the worst-case exploration of a single agent and of multiple agents. We first show that the speedup for general graphs for the multi-agent rotor-router is always between logarithmic and linear in the number of agents. We also present a tight analysis of the speedup for the multi-agent rotor-router for cycles, expanders, random graphs, cliques, constant dimensional tori and an almost-tight analysis for hypercubes.

Finally we consider *collision-free exploration*, where each agent has to explore the graph independently with the additional constraint that no two agents can occupy the same node at the same time. In the case when agents are given the map of the graph, we show an optimal algorithm for trees and an asymptotically optimal algorithm for general graphs. We also present algorithms for collision-free exploration of trees and general graphs in the case when agents have no initial knowledge about the graph.

We close the thesis with concluding remarks and a discussion of related open problems in the area of graph exploration.

Keywords: graph exploration, team of agents, algorithm, deterministic walk, rotor-routor model, parallel random walks, collaborative exploration, online algorithm

---

LaBRI,
Universite Bordeaux,
Unité Mixte de Recherche CNRS (UMR 5800),
351, cours de la Libération
33405 Talence Cedex (FRANCE)

Résumé étendu

Nous étudions dans cette thèse le problème de l'exploration parallèle d'un graphe à l'aide des multiples, synchronisés et mobiles agents. Chaque agent est une entité autonome qui n'est pas contrôlée par aucune autorité centrale. Des agents mobiles dans un graphe sont capables de visiter les sommets du graphe et traverser ces arêtes. Pendant la visite a un nœud, l'agent recueille certaines informations (type de l'information dépend du modèle), effectue un calcul selon l'algorithme exécuté et décide à son prochain mouvement.

L'agent mobile dans un graphe peut servir comme une représentation théorétique de certains importantes concepts pratiques. Ainsi, les programmes informatiques qui peuvent traverser des liens dans un réseau social sont utilisés pour classer les nœuds ou d'obtenir des échantillons homogènes de certains sous-ensembles. Programmes qui peuvent voyager dans un réseau informatique peuvent effectuer la maintenance du réseau ou détecter les nœuds défectueux.

Le problème de l'exploration d'un graphe est, étant donné un graphe $G = (V, E)$ avec $n = |V|$ nœuds et $m = |E|$ arêtes, de visiter par les agents mobiles tous les n nœuds du graphe. La question de l'exploration est l'un des problèmes les plus fondamentaux de la théorie des agents mobiles. Elle peut aussi servir comme une sous-procédure pour résoudre des tâches plus complexes comme la création du carte de graphe ou patrouille perpétuel.

Nous allons considérer deux variantes de l'exploration: collaborative et exclusive. Dans le variante de l'exploration collaborative, les agents peuvent être considérés comme une équipe, pour compléter l'exploration chaque nœud doit être visité par l'agent quelconque. Dans le second cas de l'exploration exclusive, chaque agent doit visiter tous les sommets du graphe de façon indépendante. Notre objectif est de réduire le temps d'exploration.

Nos algorithmes sont:

Déterministes Nous se contcentrons uniquement sur les solutions déterministes. Cependant parfois nous comparons nos résultats avec des résultats analogues pour les algorithmes randomisés.

Parallèle La majorité des études existantes sur le problème de l'exploration d'un graphe assume un unique agent. Dans cette thèse, nous proposons des solutions efficaces pour les équipes de k agents se déplaçant en parallèle. Dans le cas de l'exploration collaborative nous attendons que plusieurs agents accomplirons l'exploration plus rapidement qu'un seul. L'objectif est de coordonner les mouvements en telle façon, que les agents différents explorent différentes parties du graphe. Nous sommes intéressés spécialement par l'exploration effectuer par nombreux équipes d'agents.

Afin d'explorer un graphe effectivement, les agents ont besoin d'avoir un certain accès à la mémoire. Cette mémoire peut être soit la mémoire interne d'agent ou la mémoire interne des nœuds, modifiable par les agents. Dans cette thèse, nous considérons tous les deux cas. Pour permettre la parallélisation déterministe, les agents doivent interagir

entre eux. Ce interaction peut être directe ou indirecte. Un exemple d'une méthode indirecte de communication est la capacité des agents à interagir avec l'environnement, par exemple en laissant des messages au niveau des nœuds. Dans le cas de communication directe les agents peuvent échanger des messages directement entre eux.

Dans l'introduction de la thèse nous discutons les plus importants modèles des agents mobiles présents dans la littérature. Ensuite, nous définissons notre modèle et présentons les différences et les similitudes avec les modèles existants. Ensuite, nous présentons les sujets de recherche lies à aux problèmes d'exploration des graphes.

Dans la première partie technique de la thèse on analyse l'exploration de graphe dans le modèle où les agents sont équipés de mémoire interne, mais la mémoire n'est pas disponible au niveau des nœuds. Dans ce modèle les agents sont également autorisés à communiquer entre eux en échangeant des messages. De plus on suppose que tous les agents commencent à le même sommet r. Contrairement aux résultat existantes, dans ce modèle nous étudions les algorithmes pour des nombreux équipes d'agents. Nous étudions aussi l'impact de différent variants de communication par l'analyse des algorithmes où les agents peuvent se communiquer à distance arbitraire (communication globale), ou seulement avec d'autres agents situés dans le même nœud (communication locale). Les algorithmes présentés assume polynomiale nombre des agents (compare au nombre de sommets du graphe). Minimal nombre d'agents k requis par notre algorithme est k $\geq D^* n^{1+\epsilon}$, où D^* est la distance de r a le nœud le plus éloigné du graphe, et $\epsilon > 0$ est une constante positive. Première solution présenté, de même que les résultats établi pour un plus petit nombre d'agents, fonctionnent bien pour les arbres. Le premier algorithme utilise la communication globale et travaille en temps $D^* \left(1 + \frac{1}{c} + o(1)\right)$ pour $k \geq D^* n^c$ agents. Nous montrons comment on peut modifier la première solution pour obtenir un algorithme fonctionnant dans le modèle de communication locale. Nous obtenons un algorithme fonctionnant en temps $D^* \left(1 + \frac{2}{c} + o(1)\right)$ pour $k \geq D^* n^c$. Étonnamment notre algorithme s'étend facilement aux graphes généraux. Ceci montre que la difficulté de cette formulation réside dans l'exploration des arbres. Nous confirmons cette intuition en construisant une famille spécifique d'arbres qui montre les limites inférieures dans les deux modèles de communication qui répondent à près à nos résultats positifs. Nous montrons que chaque algorithme utilisant d'agents a besoin de temps au moins $D^* \left(1 + \frac{1}{c-1} - o(1)\right)$ dans modèle de communication globale et de temps $D^* \left(1 + \frac{2}{c-1} - o(1)\right)$ dans le modèle de communication locale. Nos bornes inférieures et supérieures dans les deux modèles de communication présentent une importante différence entre ces deux modèles. Cependant, la différence dans le temps optimale de l'exploration, dans ces deux modèles de communication, n'est pas grande. Ce nous amène à la conclusion que pour les équipes nombreux, la communication à longue distance n'est pas nécessaire pour efficace 'exploration. Plus forts moyens de communication, comme laisser des messages au niveau des nœuds ou la communication à longue distance, sont nécessaire pour efficace solution du problème en cas des petites équipes d'agents. Nous fermons ce chapitre en formulant les problèmes en suspens concernant directement ce modèle.

Nous considérons également le cas des agents qui n'ont pas de mémoire interne, mais la mémoire est seulement disponible au niveau des nœuds. Coordination des mouvements des agents se fait selon la dite rotor-router mécanisme. Du point de vue d'un nœud fixe, le rotor router envoie des agents qui visitent les nœuds voisin, dans un mode round-robin. Cumulativement au cours du temps, chaque nœud envoie approximativement le même nombre d'agents à chaque arête sortant. Un tel mécanisme peut être considéré comme un dé randomisation de la marche aléatoire, où chaque nœud envoie au fil du temps, en moyenne, le même nombre d'agents dans chaque direction. Le temps d'exploration de l'ensemble rotor-router avec un agent unique est bien examiné. Il a été montré dans les travaux existants que pour chaque graphe, le temps de l'exploration dans le pire des cas est $\Theta(mD)$, où m est le nombre d'arêtes et D est le diamètre du graphe. Dans cette thèse, nous nous concentrons sur le temps d'exploration par multiples agents qui interagissent avec le système rotor-router. Nous considérons le temps de l'exploration collaborative du multi-agent rotor-router modèle. Les progrès récents dans l'analyse de la durée de la couverture de plusieurs marches aléatoires indépendantes nous permet de comparer nos résultats à ceux sur les marches aléatoires et observer des similarités intéressantes. Nous comparons le gain de vitesse: la proportion entre le temps de couverture d'un agent unique et de plusieurs agents. Notre analyse du rotor-router modèle contemple généralement le pire des cas d'exécution. Nous comparons la performances du pire cas de rotor-router avec le résultat attendu par les marches aléatoires. Premièrement on développe des techniques qui nous permettent de analyser le multi-agent rotor-router modèle. Une technique particulièrement utile, appelé déploiement tardif consiste à arrêter un certain nombre d'agents pour un nombre de tours. Nous montrons dans le ralentissement lemme que si le nombre de tours pendant lesquelles les agents sont arrêtés n'est pas trop grand, le temps de couverture du déploiement tardif correspond à la durée de la couverture de l'ensemble rotor-router sans retard. En utilisant le ralentissement lemme de nous montrons que le temps de couverture de graphe quelconque par k agent rotor-router est toujours $O\left(\frac{mD}{\log k}\right)$ et $\Omega\left(\frac{mD}{k}\right)$. Cela signifie que l'accélération pour générales graphes dans le multi-agent rotor-router modèle est toujours comprise entre fonction logarithmique et linéaire du nombre d'agents. Le résultat analogue pour les marches aléatoires est une conjecture formulée par Alon *et al.*en 2008. Ensuite, nous montrons que l'accélération logarithmique de pire cas se produit sur le cycle, en montrant un estimation précise $\Theta\left(\frac{n^2}{\log k}\right)$ sur la durée du pire cas de la couverture d'un cycle. Nous avons constaté que les agents du rotor-router partage le cycle dans une collection de domaines. Pendant ce processus, chaque agent patrouille son domaine et, éventuellement, l'entende par la conquête de nouveaux nœuds. Une analyse scrupule de l'évolution des tailles de domaines nous a permis d'obtenir des limites étroites sur le temps de couverture du multi-agent rotor-router sur le ring pour différents schèmes d'initialisation d'agents (le pire des cas et le meilleur des cas). Nous calculons également le temps entre deux visites d'un nœud donné après suffisamment nombreux mouvements (temps de retour). Nous comparons ces trois résultats (temps de couverture dans le cas de meilleur initialisation, temps

de couverture dans le cas de pire initialisation et le temps de retour) pour multi-agent rotor-router et plusieurs marches aléatoires, et observons des similarités frappantes. Il s'avère que le temps de couverture dans le cas de meilleur initialisation et le temps de retour sont les mêmes, et le temps de couverture dans le cas de pire initialisation diffère par un polylogarithmique (en nombre d'agents) facteur. Nous continuons l'étude du multi-agent rotor-router modèle en montrant des limites étroites sur le temps de couverture pour expanseurs, graphes aléatoires, cliques, tores de dimension fixé et une analyse presque optimale pour hypercubes. Tous obtenir ce résultats nous avons élaborer une autre technique. Nous relions la différence entre le nombre total de visites à un nœud donné dans le multi-agent rotor-router modèle et l'attendu nombre des visites par plusieurs marches aléatoires. Notre analyse de la multi-agent rotor-router montre un lien fondamentale au processus de plusieurs indépendantes marches aléatoires. Toutefois, pour le rotor-router, nous avons prouvé des résultats qui reste des conjectures pour les marches aléatoires. Le plus étonnant est le fait que pour le rotor-router, l'accélération est au moins logarithmique pour chaque graphe pendant que pour les marches aléatoires même une accélération superconstante est encore une question ouverte.

Enfin, nous examinons de nouveau le modèle où les agents sont équipés de mémoire interne, mais ne sont pas autorisés à interagir avec l'environnement. Dans ce modèle, nous considérons l'exploration sans collision, où chaque agent doit explorer le graphe d'une façon indépendante avec une contrainte supplémentaire que deux agents ne peuvent pas occuper le même nœud en même temps. De plus, dans ce scenario les agents doivent retourner à leurs positions de départ après avoir terminé l'exploration. D'abord nous considérons l'exploration d'un arbre, dans le cas où les agents ont la carte de l'arbre. Dans ce cas, nous montrons un algorithme de complexité $n\Delta$, où Δ est le degré maximal. Ensuite nous prouvons que $n\Delta$ étapes sont nécessaires pour n'importe quel algorithme sur graphe quelconque. Ensuite, nous étendons ce résultat pour les arbres et obtenons un asymptotiquement optimal algorithme. Plus intéressant, nous avons obtenu que cette complexité est asymptotiquement optimal, en montrant un $\Omega(n\Delta^*)$ borne inférieure. Dans le cas des agents qui n'ont aucune a priori connaissance de graphe nous avons établi un algorithme fonctionnant en temps $O(n^2)$ pour les arbres et $O(n^5 \log n)$ pour graphes quelconques. Une conclusion intéressante de ce chapitre est le fait que la question est résoluble pour tout les graphes même si au départ chaque nœud contient un agent.

Nous fermons la thèse par des observations finales et une discussion de problèmes ouverts liés à au domaine de l'exploration des graphes. Nous présentons un plan de future recherche dans un modèle hybride dans la mémoire est disponible à la fois au niveau des nœuds et des agents.

Extended abstract

In this thesis we study the problem of parallel graph exploration using multiple synchronized mobile agents. Each mobile agent is an autonomous entity that is not controlled by any central authority. Mobile agents in a graph are capable of visiting nodes of the graph and traversing its edges. While being located at a node, the agent gathers some information (type of the information depends on the model), performs a computation according to the executed algorithm and decides about its next move.

The mobile agent in a graph can serve as a theoretical representation of some important practical concepts. For example computer programs that can traverse links in a social network are used to rank nodes or to get uniform samples of nodes from some subset. Programs that can travel in a computer network can perform network maintenance or detect faulty nodes.

In the graph exploration problem, the goal of the agents is, given a graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, to visit all n nodes of the graph. The problem of exploration is one of the most fundamental problems in the mobile agent theory, which can also serve as a subprocedure for solving more complex tasks like map drawing or perpetual patrolling.

We will consider two variants of exploration: collaborative and exclusive. In the collaborative exploration problem, the agents can be seen as a cooperating team, and in order to complete the exploration, each node needs to be visited by some agent. In the latter case of exclusive exploration, each agent has to visit all vertices of the graph independently. In this thesis our objective is to minimize the time of exploration.

Our algorithms are:

Deterministic We focus solely on deterministic solutions. However sometimes we compare our results with analogous results for randomized algorithms.

Parallel Most of the existing studies on the graph exploration problem focus on single-agent exploration. In this thesis we propose efficient solutions for teams of k agents moving in parallel. In the case of collaborative exploration we want to benefit from having multiple agents by completing the exploration more quickly than in the case of a single agent. The goal will be to coordinate the movements of the agents so that different agents explore different parts of the graph in parallel. We are particularilly interested in exploration with large teams of agents.

In order to explore a graph efficiently, the agents need to have access to some memory. The memory can either be the internal memory of an agent that remains intact when the agent traverses an edge or it can be an internal memory at each node that is modifiable by the agents. In this thesis we consider both cases. To achieve parallelisation deterministically, the agents also need to interact with each other directly or indirectly. An example of an indirect method of communication is ability of the agents to interact with the environment, for example by leaving messages at the nodes. In the direct communication agents can exchange messages between each other.

In the introduction to the thesis we present all the most important models of mobile agents present in the literature. Then we define our setting and highlight the differences and similarities with the existing models. Then we present a survey of the related work in the graph exploration problems and some related problems.

In the first technical part of the thesis we study the graph exploration in the model where agents are equipped with internal memory but no memory is available at the nodes. Agents in this model are also allowed to communicate between each other by exchanging messages. Additionally we assume that all agents start at the same position r. Compared to the existing solutions, in this model we study algorithms for larger teams of agents. We also study the impact of the available range of communication by analysing algorithms for agents which can communicate at arbitrary distance (*global communication*), or only with other agents located at the same node (*local communication*). We present algorithms working for a team of polynomial size (in the number of vertices of the graph). The minimal team size k required by our algorithm is $k \geq D^* n^{1+\epsilon}$, where D^* denotes distance from r to the most remote node of the graph and $\epsilon > 0$ is any constant. Our first solution, similarly as the existing work for smaller number of agents, works for trees. The first algorithm uses the global communication and works in time $D^* \left(1 + \frac{1}{c} + o(1)\right)$ for $k \geq D^* n^c$ agents. We show how to modify the first solution to obtain an algorithm working in the local communication model. We obtain an algorithm working in time $D^* \left(1 + \frac{2}{c} + o(1)\right)$ for $k \geq D^* n^c$ in the local communication. Surprisingly our algorithm easily, and without any overhead, extends to general graphs which shows that most of the difficulty in this formulation of the exploration problem lies in the tree exploration. We confirm this intuition by constructing a specific family of trees that shows lower bounds in both communication models that almost match our positive results. We show that any algorithm using $D^* n^c$ agents needs time at least $D^* \left(1 + \frac{1}{c-1} - o(1)\right)$ in the global communication and $D^* \left(1 + \frac{2}{c-1} - o(1)\right)$ in the local communication model. Our lower and upper bounds in both communication models show a separation between these two models. However, the difference in the optimal exploration time in these two communication models is not big which leads to the conclusion that for such large teams of agents, the long-distance communication is not necessary for efficient graph exploration. Stronger means of communication like leaving messages at nodes or the long-distance communication might be necessary for solving the problem efficiently by smaller teams of agents. We close the chapter with open problems related directly to exploration in this model.

We also consider agents which have no internal memory but the memory is available at the nodes. Coordination of movements of the agents is made according to the so-called *rotor-router* mechanism. From the perspective of a fixed node, the rotor-router sends out agents which visit the node along its outgoing edges, in a round-robin fashion. Cumulatively over time, each node is sending approximately the same number of agents to every outgoing edge. Such a mechanism can be seen as a derandomization of the random walk where each node is sending over time, on average, the same number of agents in

every direction. The exploration time of the rotor-router with a single agent is well understood. It has been shown in the existing works that for any graph, the worst-case exploration time is $\Theta(mD)$, where m denotes number of edges and D, the diameter of the graph. In this thesis we focus on the exploration time of multiple agents interacting with the same rotor-router system. We consider the collaborative exploration time (or the cover time) of the multi-agent rotor-router. Recent progress in the analysis of the cover time of multiple independent random walks allows us to compare our results to the results about random walks and observe some interesting similarities. We compare values of speedup which is the ratio between the cover time of a single agent and of multiple agents. Our analysis of the rotor-router usually assumes the worst-case initialization of the system. We compare the worst-case performance of the rotor-router to the expected case in the random walks. First we develop techniques that allow us to analyse the multi-agent rotor-router. A particularly useful technique, called *delayed deployment* consists in stopping some number of agents for some number of rounds. We show in the *slow-down lemma* that if the number of rounds in which agents are stopped is not too big, then the cover time of the delayed deployment corresponds to the cover time of the undelayed rotor-router. Using the slow-down lemma we show that the worst-case cover time for any graph of k agent rotor-router is always $O\left(\frac{mD}{\log k}\right)$ and $\Omega\left(\frac{mD}{k}\right)$. This means that the speedup for general graphs for the multi-agent rotor-router is always between logarithmic and linear in the number of agents. An analogous claim for random walks is a conjecture made by Alon *et al.* in 2008. Then we show that the logarithmic speedup occurs for the worst-case initialization on the cycle by showing a tight bound $\Theta\left(\frac{n^2}{\log k}\right)$ on the worst-case cover time on a cycle. We observed that agents following the rotor-router on the cycle partition the cycle between themselves into a collection of domains. During the process, each agent is patrolling its domain and possibly extending its size by capturing new nodes. Careful analysis of the evolution of sizes of domains allows us to obtain tight bounds on the cover time of the multi-agent rotor-router on the ring for different initialization of agents (worst-case and best-case). We also show what is the time between two visits at a given node after sufficiently many steps (*return time*). We compare all these three values (cover time in the best case initialization, cover time in the worst case initialization and the return time) for multi-agent rotor-router and multiple random walks and observe striking similarities. It turns out that the cover time in the best case and the return time are the same and the best-case cover time differs by a polylogarithmic (in the number of agents) factor. We continue the study on the multi-agent rotor-router by showing tight bounds on the cover time of the multi-agent rotor-router for expanders, random graphs, cliques, constant dimensional tori and an almost-tight bounds for hypercubes. To show these results we develop another useful technique. We bound the discrepancy between the total number of visits at a given node in the multi-agent rotor-router and the expected total number of visits by multiple random walks. Our analysis of the multi-agent rotor-router shows how closely this process is linked to the process of multiple independent random walks. However for the

rotor-router we proved some results that remain a conjecture for the random walks. Most surprisingly we show that a for the rotor-router, the speedup is at least logarithmic for any graph and for random walks even a superconstant speedup for any graph is still an open question.

Finally we consider again the model in which agents are equipped with memory but are not allowed to interact with the environment. In this model we consider *collision-free exploration*, where each agent has to explore the graph independently with the additional constraint that no two agents can occupy the same node at the same time. Moreover, in this problem agents are required to return to their starting positions after finishing the exploration. We first consider tree exploration in the case when agents are given the map of the tree. In such a case we shown an algorithm with time complexity $n\Delta$, where Δ denotes the maximum degree. Then we show that $n\Delta$ steps are needed for any algorithm on any graph. Then we extend the result for trees and obtain an asymptotically optimal for general graphs working in time $O(n\Delta^*)$, where Δ^* denotes the maximum degree of the minimum degree spanning tree of the considered graph. Interestingly we show that this complexity is asymptotically optimal by showing a $\Omega(n\Delta^*)$ lower bound on any graph. For the case in which initially agents have no knowledge about the graph we show an algorithm working in time $O(n^2)$ for trees and $O(n^5 \log n)$ for general graphs. An interesting conclusion from this chapter is the fact that the problem is solvable for any graph even if initially each node contains an agent.

We close the thesis with concluding remarks and a discussion of related open problems in the area of graph exploration. Presented directions of future work involve graph exploration in a hybrid model in which memory is available both at the nodes and at the agents.

Contents

Chapter 1

Introduction

Autonomous, unmanned, mobile robots are capable of performing many operations that were previously done by humans. Progress in electronics and robotics has allowed mobile robots to become sufficiently inexpensive to appear on the mass market. Domestic robots nowadays perform certain household tasks like vacuuming or gardening. In hazardous environments, such as minefields [17] or damaged buildings [125], deploying mobile robots reduces risk of human casualties. A team of robots can also perform a surveillance and reconnaissance mission in a dynamic urban environment [101].

Study on mobile robots is not restricted to physical robots. There is a broad spectrum of applications of multiple mobile software agents that can perform tasks in computer networks. Such agents can collect information, perform network maintenance, or rank nodes in a web of information. Mobile agents have the ability to react dynamically, which makes them suitable for heterogeneous, dynamically changing, or faulty environments [119]. In a communication network, instead of transmitting large amounts of data between hosts, it is often more efficient to transfer a mobile agent that can perform an operation in the target node. This can lead to reduced communication load and latency in the network. A similar application are web crawlers that traverse links in the Internet in order to create an index of web pages (e.g. Googlebot). In social networks, where nodes represent users and connections represent friendship relations, crawler agents can also be used to obtain uniform samples of users [96].

The main principle in designing both physical robots and software agents is simplicity, due to limited available resources or price of the device. Entities are subject to a number of constraints such as limited memory, speed and range of communication. Nevertheless, a large team of even very weak devices executing simple algorithms can often perform tasks that are very hard for centralized systems.

The problem on which we will focus the most in this thesis is *exploration*. In this task the goal of an agent or a group of agents is to visit the whole of the accessible environment. Exploration of an unknown terrain is a fundamental problem in robotics as it is one of the most natural applications of multiple mobile entities. It is also a basic subtask in many more complex problems like patrolling or map drawing.

1.1 Modelling a mobile agent

A mobile agent is an entity equipped with some operational memory and computational power. The agent can operate in a discrete environment (modelled as a graph) or in a continuous environment (modelled for example as a subset of the plane). The agent operating in a graph is capable of traversing edges and visiting nodes of the graph. Upon traversing edges, the memory of the agents remains intact. When visiting a node, the agent can gather new information about the graph (e.g., the degree of its host node), perform computations, and decide about its next move. An agent is allowed to perform computations and possibly modify its memory only when located at a node. The agent deployed in the continuous environment can move along a continuous curve within available subset of the plane. In the following we give a brief overview of different features of the models of the mobile agents considered in the literature.

1.1.1 Properties of mobile agents

Memory. When considering the amount of operational memory available to agents, three cases are considered in literature. Results for agents with unbounded memory usually focus on minimizing the time of completion of the task. The second case is that of agents with bounded memory. Here, the studied aspect is the feasibility of performing the task subject to the given memory constraint or the tradeoff between time and memory. Finally, for the case of agents with no memory (*oblivious agents*), most of the research effort focuses on the feasibility of performing the given task.

Vision. In order to perceive the environment, the agents are endowed with visibility sensors. As the power of the sensors may vary, we distinguish between different models of view ranges of the agents. With *global vision*, agents can see the whole environment. If agents are deployed in a graph, then in this model, each agent receives information about the topology of the graph, its own position in the graph, and sometimes the positions of other agents. In one special case, well-studied in the literature under the name of *multiplicity detection*, an agent receives information if there is one or more agents located at a given node of the graph. With *local vision*, each agent receives only local information. For example if agents are operating in a graph, they receive the degree of currently occupied node, they may receive information about other agents occupying the same node or about identifiers of the neighboring nodes.

Knowledge of global parameters. The knowledge of certain global parameters of the graph can sometimes be essential when deciding the solvability and efficiency of certain tasks. Thus, often agents operating in the visibility model are given the values of the size of the graph, the diameter of the graph, or the whole topology. Revealing this information can significantly reduce the complexity of the problem.

Interactions. Other properties that may be considered in mobile agent computing include the capability of agents to interact with each other and with the environment. Interactions between agents include exchange of information by agents located at the same node (*local communication*) or at arbitrary locations (*global communication*).

Another indirect method of communication between agents is through their interaction with the environment. A large number of different variants of such interaction have been studied in literature. Agents may write arbitrary information on nodes (*whiteboards*), or may leave movable or immovable tokens. In some models, agents may also interact with the environment and each other by influencing values of counter, pointers and other variables stored on nodes and edges of the system. In one significant part of this thesis we will study a method of interaction with the environment known as the *rotor-router*. In the rotor-router, each node maintains a cyclic ordering of its outgoing arcs, and during successive visits of the agents, propagates it along arcs chosen according to this ordering in round-robin fashion.

Symmetry breaking. When multiple agents are deployed in the same graph in order to perform some task, we assume that they operate in a distributed setting, i.e. are not coordinated by any centralized authority and have to make decisions autonomously. It is also assumed that all agents are executing the same algorithm. Breaking the symmetry between the agents is frequently an essential part of solving the agents' task, and can be achieved in a number of ways. Agents may have distinct identifiers which are a part of the input for the algorithm executed by the agents (*labeled agents*). By contrast, agents without identifiers are referred to as *anonymous agents*. Symmetry for anonymous agents can be broken by placing agents in different starting positions. Finally, we may also consider agents which have access to some source of randomness and can make probabilistic choices, and e.g. use random coin tosses to break symmetries.

Synchronization. Regardless of the specific setting, an algorithm executed by the mobile agent can be seen as a sequence of Look-Compute-Move cycles. In one cycle, an agent takes a snapshot of the current configuration (Look), makes a decision (Compute) to stay idle or to move. A potential move is made in the third phase of the cycle (Move). Three main models of synchronization of these cycles are considered in literature. In the *synchronous* model, agents have access to a global clock and in every round all agents execute each phase of each cycle simultaneously. Another model of synchronization is the semi-synchronous ATOM model [142], in which each agent executes the Look-Compute-Move cycle independently at unpredictable time instants. In the ATOM model, the whole Look-Compute-Move cycle is atomic, i.e. is performed instantaneously. In the last model, the asynchronous Look-Compute-Move model, the delays between each phase may be arbitrarily long and hence, agents may move based on significantly outdated perceptions. Sometimes a model in which not only delays between phases are arbitrary long but also each of three phases Look-Compute-Move can last for any number of time steps are considered [31]. Most results for asynchronous models assume global visibility.

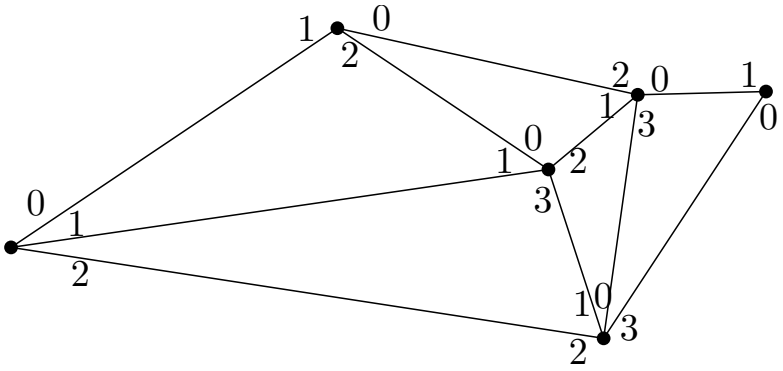

FIGURE 1.1: An example of a port labelled graph.

Faults. Since mobile agent models assume simple, cheap and relatively weak devices, one cannot assume fail-proof software or hardware, in particular when such agent systems are deployed in hazardous environments. Thus, an important line of study concerning the theory of algorithms for mobile agents deals with aspects of fault tolerance. Two types of faults are prevalent in the literature: *crashes* and *Byzantine faults*. When an agent crashes, it simply stops executing the algorithm, does not take any further action, and remains stationary indefinitely. By contrast, a Byzantine fault can be seen as an adversary taking control over an agent. Such an agent can behave in an arbitrary way, at times pretending to be operating correctly, at others possibly disrupting the work of other agents. In this thesis, we will assume the fault-free models and leave the problems of faulty agents as possible directions of further study in the topic of the thesis.

1.1.2 Model definition and notation

In this thesis we focus on exploration of discrete environments. The environment in which the agents are deployed is defined simply as an undirected[1] graph $G = (V, E)$. In order to allow for the agents to distinguish among incident edges, each node of the graph admits a local labelling (port labelling). For every node $v \in V$ this local edge labelling is defined as a function $\lambda_v : \{(v, w) \in E\} \to \{0, 1, \ldots, \deg(v) - 1\}$, where $\deg(v)$ is the degree of v (see Figure 1.1 for an example). We will use the following notation. We will denote the number of nodes of graph G by $n = |V|$, the number of its edges by $m = |E|$. By D we will denote the diameter of G. The set of neighbors of a vertex $v \in V$ is denoted by $\Gamma(v)$. When each node of the graph has a unique label $id : V \to \{1, 2, \ldots, n'\}$, for some $n' \geq n$, we will say that the graph is *labelled*. In the literature usually the case of bijective labelling is considered, i.e., the case of $n' = n$. If a labelling is not available we will say that the graph is *anonymous*.

Behaviour of an agent. A mobile agent is defined as an automaton which is capable of both performing computations and traversing edges of the graph G. The agent has some number of states. The number of states of an agent may depend on the size of

[1]This thesis studies only undirected graphs but some aspects of exploration of directed graphs are also considered.

the graph and thus is potentially infinite. We will say that an agent equipped with b bits of local memory has 2^b states. We will assume that an agent while being located at a node can use additional memory for local computation but when traversing an edge it can carry only b bits. The agent starts at some node of G in some state s. During successive time steps the agent can change its position by traversing edges and moving to a neighboring node and also can change the state of its local memory. Time is divided into steps and in each step t every agent performs the following:

1. The agent gathers available information about the graph or other agents, depending on the model. It learns the degree of the current node v, and depending on the model, it may communicate with other agents or interact with the environment.

2. Based on the gathered information, its local memory state, and according to the executed algorithm, the agent performs its computation.

3. As the result of its computation, the agent can modify its local memory.

4. The output of the computation is either a label of an edge incident to v leading to one of the neighbours of v or an idle move.

5. If the agent chooses a label, we say that it *traverses* the corresponding edge. At the end of time step t, it appears at the neighbouring node. Otherwise, if the agent chooses an idle move, it remains in v.

1.1.3 Our focus

Different assumptions in mobile agent computing theory give rise to a large number of models. Here, we define properties of the model which will be common to all results presented in this thesis.

***Discrete* vs. continuous setting.** The first important assumption we make is about the environment in which the agents are deployed. In general, the environment can either be continuous (modelled, for example, as a subset of the plane) or discrete (modelled as a graph). In this thesis, we focus only on discrete environments, modelled as an undirected graph.

***Deterministic* vs. randomized algorithms.** The other assumption concerns the way agents can move. Their choices can either be randomized or deterministic. In the randomized model it is assumed that each agent has access to an independent source of randomness. In this thesis, we will assume that such sources are not available and we will consider only deterministic strategies.

***Synchronized* vs. asynchronous agents.** A different aspect of the model is synchronization. We will assume that agents move in synchronous rounds as if they had access to a global clock. In one round every agent can traverse one edge. We will also assume that all agents start executing the algorithm at the same moment. This

can be contrasted with the asynchronous setting, where delays between operations made by each agent can be arbitrary as if they were decided by an adversary.

***Time* vs. space efficiency.** Finally, we can consider different objectives. We may want to minimize time of exploration, memory of each agent or the tradeoff between time and memory. In this thesis, we will focus only on minimizing the time. We will always assume that the time of internal computations made by the agents is negligible and thus by the *runtime* of the algorithm we will understand the number of rounds of the global clock at the time moment when the agents have achieved their objective.

As we are working in the synchronous model, all steps are done by agents in parallel. In particular, agents are making their observations and choices at the same time at the beginning of each round. During each round, all agents that chose to move are traversing edges in parallel.

Local model with memory on agents. In Chapters 2 and 4, we are using the following model. We will assume that nodes of the graph are uniquely labelled, but no memory at nodes is available for the agents. Thus agents cannot use whiteboards and cannot mark nodes in any way. Agents have local visibility and each agent can perceive the identifier of the currently occupied node v and the identifiers of all the neighbors of v. Moreover, each agent is equipped with internal memory that remains intact while the agent traverses an edge. We do not make any assumptions about the size of the internal memory of each agent.

Local model with memory on nodes. By contrast, in Chapter 3, the agents have no internal memory and the whole mechanism determining the movements of the agents is provided by the environment. In such a model the nodes have states, but not the agents. Each node in this model has some state and the next move of an agent located at node v depends on the state of v. Each visit of an agent at a node modifies the state of the node. We assume that agents are labelled and upon visiting a node, each agent receives identifiers of other agents located at the same node. With this assumption it is possible for agents located at the same node to move to different neighbors.

1.2 Studied aspects of the graph exploration problem

In the graph exploration problem, an agent or group of agents is placed on a node of a graph and moves between adjacent nodes, with the goal of visiting all the nodes of the graph. An *exploration strategy* for G is a sequence of moves performed independently by the agents.

Collaborative exploration time. We will say that an agent *visits* a node v in step t if it is located at v at the beginning of step t. A node v is *explored* in step t if an agent

is located in v in the end of step t and no agent was located in v before t. Graph G is said to be *explored* by step t if all nodes of G are explored in or before step t. The *exploration time* (or *cover time*) of an exploration strategy for G (or an exploration algorithm) controlling the mobility of the agents is the smallest time t such that G is explored by step t.

Collision-free exploration time. An agent is said to *explore* graph G if it visits all nodes of the graph. The collision-free exploration time of an algorithm for a team of agents is the smallest time t such that until the end of time step t each agent, independently of other agents, explores the graph G with the additional constraint that no two agent can occupy the same node at the same time.

Exploration with stop. The problem of exploration with stop is considered to be completed by time t, if t is the smallest time such that the considered exploration is completed by time t, all agents are aware of this fact and do not make any more moves for any time step bigger than t.

Exploration with return. The time of exploration with return is the smallest time t such that the considered exploration task is completed by time t and all agents are once again located at their starting positions at time moment t.

Formulations of our problems. In Chapters 2 and 3 we will consider the collaborative exploration problem. We will assume that agents do not need to stop after completion of the task and are not even required to be aware that the exploration has been finished.

By contrast, in Chapter 4 we will study the collision-free exploration problem with return. In this problem each agent has to explore the graph without meeting other agents and moreover after completing the task each agent has to return to its starting location.

1.3 Overview of the thesis and results

In this thesis, we will study the exploration problem in three different settings.

In Chapter 2, we will consider the model where agents are allowed to interact between each other. We will study exploration in two different communication models: *global communication*, where agents can exchange information at any time, and *local communication*, where agents can interact only while being simultaneously present at the same node. We will study exploration using a large team of agents (i.e., of size polynomial in the size of the graph) as opposed to [67, 85] where the authors considered smaller number of agents. In both communication models, we will propose efficient algorithms for exploration for a large team of agents as well as almost matching lower bounds.

In Chapter 3, we will consider exploration by agents guided by a specific mechanism provided by the environment called the rotor-router model. In this model, each node

of the graph maintains a pointer to one of its neighbors and a cyclic sequence of its neighbors. Each agent during each time step, follows the pointer and traverses the corresponding edge. The pointer is then advanced to the next neighbor in the sequence. We will study the cover time (i.e. time until each nodes has been visited by at least one agent) of the system of multiple agents propagated by the rotor-router mechanism. We will study the exploration time (= the cover time) of multiple agents interacting with the same rotor-router system. We will study the speedup, i.e., the ratio between the cover time for single agent and for multiple agents. We will completely characterize the possible range of speedups for general graphs. Surprisingly, the range of speedups for multi-agent rotor-router turns out to be exactly the same as the conjectured range of speedups for multiple random walks [7]. We will also derive the precise values of cover time for multi-agent rotor-router for many graph classes like for example cycles, multidimensional tori and random graphs. Our results can be compared to those obtained by Elsässer and Sauerwald [71] in an analogous study of the cover time of k independent parallel random walks in a graph; for the rotor-router, we obtain tight bounds in a slightly broader spectrum of cases.

In Chapter 4, we study graph exploration in the model where agents cannot communicate between each other nor interact with the environment. Agents in this model are not aware of the number nor the positions of the other agents. We consider the problem of collision-free graph exploration, i.e., a task in which each agent has to visit every node of the graph and in no round may two agents occupy the same node. We consider two scenarios: one in which each mobile agent knows the map of the graph, as well as its own initial position and the second in which the graph is unknown in advance. In the first scenario we provide tight (up to a constant factor) lower and upper bounds on the collision-free exploration time in general graphs, and the exact value of this parameter for trees. For our second scenario, we also propose collision-free exploration strategies for tree networks and for general graphs. The collision-free exploration problem turns out to be solvable for any graph even if each node is initially occupied by an agent.

In Chapter 5 we conclude the thesis, presenting perspectives for further research in mobile agent graph exploration.

1.4 State-of-the-art on the exploration problem

The study of graph exploration is closely linked to central problems of theoretical computer science, such as the question of deciding if two nodes of the graph belong to the same connected component (*st-connectivity*).

Recent progress in the graph exploration problem has resulted in a large number of results, which due to a multitude of models and assumptions, are sometimes hard to compare. Here we will try to organize results presented in the literature and highlight the main features of the models for which they have been established.

Whereas our focus will be on the case in which the graph is not known in advance to the exploring agents (*the online scenario*), we start our discussion with a broader and to some extent historical overview of related work, which also includes the offline scenarios with complete knowledge.

1.4.1 State-of-the-art on deterministic exploration

Single-agent exploration of labelled graphs. For a given known graph, the problem of deciding whether there exists an exploration strategy using a given number of edge traversals is NP-hard. Indeed, a n-node graph admits exploration in $n-1$ steps if and only if it contains a Hamiltonian path starting at the initial location of the agent. There do exist however simple and efficient exploration strategies.

One of the simplest deterministic exploration algorithms is Depth-First Search (DFS), which was first investigated in the context of escaping from a maze by Charles Pierre Trémaux in the 19th-century [73]. In DFS, the agent performs exploration of a spanning tree of the graph G rooted at the starting position of the agent. An agent located at node v moves to an arbitrary unexplored neighbor of v, if such a neighbor exists. If all neighbors of v have been explored, then the agent backtracks, i.e., moves to a parent of v in the spanning tree. To perform such an algorithm, the agent needs to be able to distinguish between visited and unvisited nodes, thus some form of node labels is necessary. It also needs to remember at least the path in the spanning tree between its current location and the starting position. Thus, exploration with the simple DFS algorithm requires time at most $2m$ and space $O(n)$. An improvement in the time complexity in this scenario was made by Panaite and Pelc in [131] where an algorithm requiring time $m+3n$ is shown, which improves the time of DFS provided that $m > 3n$.

A different approach is to study exploration using agents whose moves are restricted in some way. Such a related setting is studied by Duncan *et al.* [64], where an agent has to explore a graph while being attached to the starting point by a rope of restricted length or has to return regularly to the starting point, for example for refuelling. In this model, the authors show an asymptotically optimal algorithm with $\Theta(m)$ edge traversals.

In the case when the graph is not known in advance (also known as the *online case*), one of the measures of efficiency of an algorithm is its *competitive ratio*, understood as the worst-case ratio between the time of the online algorithm and time of the optimal offline algorithm, i.e., the optimal algorithm for an agent working with full knowledge of the graph. For unknown undirected graphs, DFS can be considered as an online strategy, which achieves a competitive ratio of at most 2, because it traverses every edge at most twice. Dessmark and Pelc [51] studied the competitive ratio of algorithms assuming different initial knowledge about the graph. If a map of the graph with marked starting position of the agent is given, then the optimal competitive ratio is $7/5$ for lines and $3/2$ for trees. For a map without a starting position it is shown that the optimal competitive ratio for lines is $\sqrt{3}$ and that it is less than 2 for trees (thus DFS is not optimal).

The problem becomes more difficult for directed graphs, as an agent may not be able to backtrack its moves. Deng and Papadimitriou [49] proposed an algorithm with competitive ratio $\delta^{O(\delta)}$, where δ is the minimum number of edges that must be added to make the graph Eulerian (the *deficiency* of the graph). They also conjectured that there exists a $poly(\delta)$-competitive algorithm. The conjecture was proven by Fleischer and Trippen [78] where a $O(\delta^8)$-competitive algorithm was presented.

Memory-efficient graph exploration. When considering the problem of minimizing the required space needed by an agent to explore any graph, a first natural question would be if an agent with a constant number of states (equivalent to a finite automaton) can explore any graph. The question was answered negatively by Rollik [140], who showed that even a team of finite cooperating automata cannot explore graphs of arbitrary size. A sequence $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \ldots$, where $\mathcal{G}_i$ is the set of all regular graphs that can be explored by a finite automaton with i states was studied by Fraigniaud *et al.* [87]. They showed that there exists a polynomial $f : \mathbb{N} \to \mathbb{N}$ such that $\mathcal{G}_i$ is strictly included in $\mathcal{G}_{f(i)}$. The question whether for all $i > 0$, $\mathcal{G}_i$ is strictly included in $\mathcal{G}_{i+1}$ is a conjecture that has been proven for $i = 1, 2$ [87].

The problem of exploring the graph is related to the problem of deciding whether a given pair of nodes s, t belong to the same connected component of some undirected graph (*st-connectivity, USTCON*). This problem is of central importance in complexity theory due to the fact that $USTCON$ is complete in the class of problems solvable on a log-space Symmetric Nondetermistic Turing Machine (SL). If we model vertices of the network as states of a given Turing Machine and directed edges as possible transitions between the states, then *st*-connectivity on such a graph is equivalent to the question whether the considered Turing Machine starting from state s can finish computation in an accepting state t. $USTCON$ can be solved by a log-space Nondeterministic Turing Machine and thus by Savitch's theorem [141] it can be solved deterministically in space $O(\log^2 n)$. Progress towards reducing this space complexity took almost three decades. Nisan *et al.* [127] made the first improvement over Savitch's algorithm in terms of space showing that USTCON can be solved in $O(\log^{3/2} n)$ space. Later this complexity was reduced to $O(\log^{4/3} n)$ by Armoni *et al.* [13]. Finally Reingold [139] showed that $USTCON$ solvable in logarithmic space, which implies that $SL = L$ (where L is the class of problems solvable on a log-space Turing Machine and SL – on a log-space Symmetric Turing Machine). A (possibly) more difficult, directed version of *st*-connectivity ($STCON$) can be shown to be a complete problem in the class of all problems solvable on a log-space Nondeterministic Turing Machine (class NL). It can also be solved deterministically in space $O(\log^2 n)$ [126]. Until now it is the most space-efficient solution to this problem.

In the context of graph exploration with mobile agent the universal traversal/exploration sequences use port-labels but do not need node identifiers. The existence of polynomial-size universal sequences shows that an agent with $O(\log n)$ bits of memory

and knowledge of n, can deterministically explore any port-labelled anonymous graph of size n. As we do not consider time or memory complexity of local computations, the agent can construct the lexicographically smallest sequence in each step. The $O(\log n)$ bits are needed for the counter that keeps track of the position in the sequence. Fraigniaud *et al.* [86] showed that $\Omega(\log n)$ bits of memory are sometimes required. When expressing the number of bits as a function of diameter D and maximum degree Δ, it is possible to show that $\Theta(D \log \Delta)$ bits are sufficient, and necessary for graphs with arbitrarily large values of D and Δ [86]. For the specific case of trees, $O(\log \Delta)$ bits are sufficient for perpetual exploration and $\Omega(\log \log \log n)$ bits are required for exploration with stop [54]. For exploration of trees with return $\Theta(\log n)$ bits are necessary [54] and sufficient [11], regardless of whether or not the agent knows n. A lower bound on memory size for a team of k agents executing a possibly different algorithm was presented in [88]. If agents are not allowed to communicate then to explore any graph of size n deterministically, each agent needs $\Omega(\log(n/k))$ bits of memory [88].

The results for *st*-connectivity can also be studied in a weaker computational model called Jumping Automaton for Graphs (JAG) proposed by Cook and Rackoff [32]. Such an automaton is equipped with P pebbles and Q memory states thus its memory usage is $P \log n + \log Q$ (when considered as a Turing machine). A JAG controls pebbles and can either move a pebble along an edge to an adjacent vertex or jump it to another vertex containing a pebble. The authors in [32] show that there is a JAG which can solve STCON in storage $O(\log^2 n)$ and any JAG needs space $\Omega(\log^2 n / \log \log n)$. This result is strong indication that STCON is probably not in L.

For more detailed surveys of anonymous graph exploration, we refer the interested reader to [95, 107].

Exploration assisted by the environment. An important line of research is devoted to strategies in which the agent has no operational memory and the whole process of propagation is performed by the environment. We will discuss two such mechanisms, the *rotor-router* and the *basic walk*.

In the rotor-router model, introduced by Priezzhev *et al.* [136], the behaviour of the agent is fully controlled by the undirected graph in which it operates. The edges outgoing from each node v are arranged in a fixed cyclic order known as a *port ordering*, which does not change during the exploration. Each node v maintains a pointer which indicates the edge to be traversed by the agent during its next visit to v. The initial position of the pointer can be chosen at random [63] or can be controlled by an adversary [15]. The next time when an agent enters node v, it is directed along the edge indicated by the pointer, which is then advanced to the next edge in the cyclic order of the edges adjacent to v (see Figure 1.2). Each agent propagated by the rotor-router is a memoryless entity and the system requires no special initialization as its state at any moment of time is a valid starting state for the process. Studies of the rotor-router started with works of Wagner *et al.* [143] who showed that in this model, starting from an arbitrary

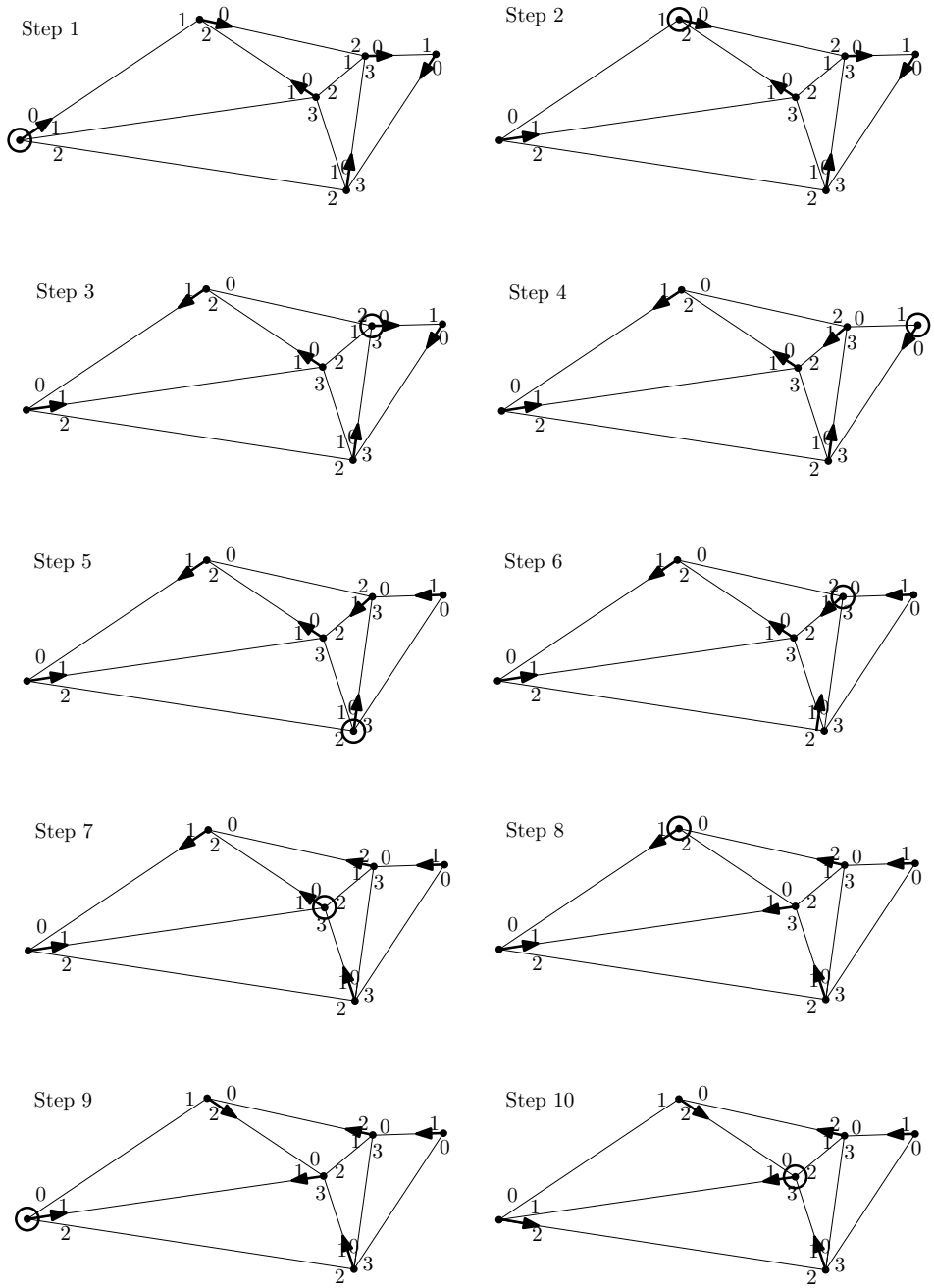

FIGURE 1.2: An example of graph exploration with the single-agent rotor-router (the agent is marked with the circle), assuming that the cyclic ordering of the edges corresponds to the port labeling.

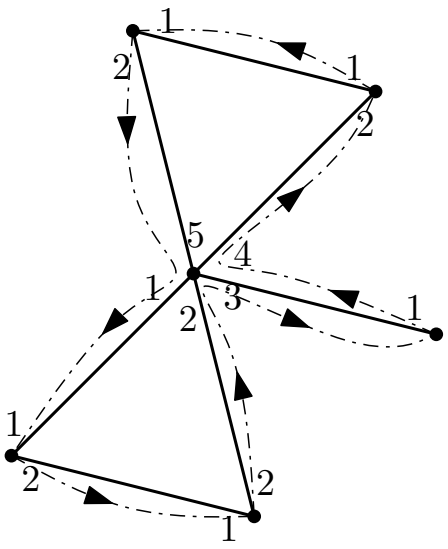

FIGURE 1.3: An example of graph exploration with the basic walk.

configuration (arbitrary cyclic orders of edges, arbitrary initial values of the port pointers and an arbitrary starting node) the agent covers all edges of the graph within $O(nm)$ steps. Bhatt *et al.* [19] showed later that within $O(nm)$ steps the agent not only covers all edges but *enters (establishes) an Eulerian cycle*. More precisely, after the initial *stabilisation period* of $O(nm)$ steps, the agent keeps repeating the same Eulerian cycle of the directed symmetric version $\vec{G}$ of graph G (see the model description for a definition). Subsequently, Yanovski *et al.* [144] and Bampas *et al.* [15] showed that the Eulerian cycle is in the worst case entered within $\Theta(Dm)$ steps in a graph of diameter D. Considerations of specific graph classes were performed in [91]. Robustness properties of the rotor-router were further studied in [16], who considered the time required for the rotor-router to stabilize to a (new) Eulerian cycle after an edge is added or removed from the graph. Regarding the terminology, we note that the rotor-router model has also been referred to in the literature as the *Propp machine* [15] or the *Edge Ant Walk algorithm* [143, 144], and has also been described in [19] in terms of traversing a maze and marking edges with pebbles.

The rotor-router can be also seen as a strategy in which an agent chooses a neighboring edge for which the most time has elapsed since its last traversal in the directed symmetric version $\vec{G}$ of graph G. Such a strategy is sometimes referred to as the Oldest-First strategy. on $\vec{G}$ [34]. By contrast it turns out that in undirected graph, such an Oldest-First strategy can lead to exponential cover time [34]. On the other hand a strategy in which an agent chooses a neighboring edge with the smallest number of traversals (Least-Used-First) in undirected graphs always achieves a cover time of $O(mn)$ [34].

Another simple strategy for movement in a graph is the basic walk, sometimes referred to as a walk following the right-hand rule (see Figure 1.3 for an example). In the basic walk, an agent who entered a node via port p is propagated in the next step via port $p+1$ (modulo the degree of the node). The basic walk can also be seen as a traversal using a Universal Exploration Sequence with all values in the sequence equal to 1. Contrary

to the rotor-router which is a valid exploration strategy regardless of initialization, the basic walk is not guaranteed in general to explore every graph, and may be e.g. easily stuck in a short cycle. Consequently, the question studied for the basic walk consists in finding a specific port labelling which leads to a cycle that visits all nodes of the graph. Kosowski and Navarra [112] proved that there always exists a port labelling resulting in a tour of length $4n-2$, while Czyzowicz *et al.* [40] showed that for some graphs a tour of length $2.8n$ is necessary. A modification of a basic walk introduced by Ilcinkas [103] is to equip the agent with a constant number of bits of memory. In such setting Ilcinkas [103] showed an algorithm for port setting and an algorithm for the agent resulting in a tour of length $4n-2$. This length was later improved by Gąsieniec *et al.* [94] to $3.75n-2$ and finally to $3.5n-2$ by Czyzowicz *et al.* [40].

Exploration of continuous environment. Exploration of a geometric polygon, in a continuous environment, is considered in [43, 99]. The authors study two scenarios: the *unlimited* vision, in which the agent situated at a point p of the terrain explores (sees) all points q of the polygon for which the segment pq belongs to the polygon, and the *limited* vision, when we require additionally that the distance between p and q be at most 1. The task of the agent is to see all points. In [99] the authors propose algorithm for the case with unlimited vision with a competitive ratio of 1.219 for general simple polygons and of 1.167 for rectilinear simple polygons. For unlimited vision, in [43] the authors show a $O(P+DH)$ algorithm, where P is the total perimeter of the terrain, D is the diameter of the convex hull, and H is the number of holes. For limited vision they propose an algorithm with running time $O(P+C+\sqrt{CH})$ where C is the area of the terrain (excluding obstacles) and show a family of polygons for which $\Omega(P+C+\sqrt{CH})$ time is required for every algorithm with limited vision.

Multiple agents. In the problem of collaborative graph exploration with multiple agents, the exploration is completed when every vertex has been visited by at least one agent. As for the case of a single agent, we can consider the offline and online scenarios. The problem of optimal offline collective exploration is NP-hard even if we restrict it only to the case of trees [85]. It is, however, easy to show that an asymptotically optimal algorithm in the offline case takes $\Theta(D+n/k)$ steps (see Proposition 2.1).

The *competitive ratio* for any online exploration algorithm is defined just as for a single agent, as the ratio between the cover time of the algorithm and the cover time of the optimal offline algorithm. Collaborative online graph exploration has been intensively studied for the special case of trees. In [85], a strategy was given which completes collaborative exploration with return of any tree with a team of k agents in $O(D+n/\log k)$ time steps, using a communication model with whiteboards at each vertex that can be used to exchange information. This corresponds to a competitive ratio of $O(k/\log k)$ with respect to the optimum exploration time of $\Theta(D+n/k)$ in the offline scenario. In [100] the authors show that the competitive ratio of the strategy presented in [85] is precisely $k/\log k$ and can be modified to work in local communication model.

Another DFS-based algorithm, given in [22], has an exploration time of $O(n/k + D^{k-1})$ time steps, which provides an improvement for graphs of small diameter and small teams of agents, $k = O(\log_D n)$. For a special subclass of trees called sparse trees, [66] introduces online strategies with a competitive ratio of $O(D^{1-1/p})$, where p is the density of the tree as defined in that work. The best currently known lower bound on competitive ratio is much lower: in [67], it is shown that any deterministic exploration strategy with $k < \sqrt{n}$ has a competitive ratio of $\Omega(\log k / \log\log k)$, even in the global communication model. A stronger lower bound of $\Omega(k/\log k)$ holds for so-called greedy algorithms [100]. In [122] a lower bound of $\Omega(D^{1/(2c+1)})$ on the competitive ratio is shown to hold for a team of $k = n^c$ agents, which holds only for the class of so-called rebalancing algorithms i.e. algorithms which keep all agents at the same height in the tree throughout the exploration process.

The same model for online exploration is studied in [130], in the context of geometric graphs, inspired by real-world tasks known as milling and lawn-mowing problems. A strategy is proposed for exploring graphs which can be represented as a $D \times D$ grid with a certain number of disjoint rectangular holes. The authors show that such graphs can be explored with a team of k agents in time $O(D \log^2 D + n \log D/k)$, i.e., with a competitive ratio of $O(\log^2 D)$, since the optimal offline strategy performs in $\Theta(D + n/k)$ steps. By adapting the approach for trees from [67], they also show lower bounds on the competitive ratio in this class of graphs: a lower bound of $\Omega(\log k/\log\log k)$ for deterministic strategies, and a lower bound of $\Omega(\sqrt{\log k}/\log\log k)$ for randomized strategies. These lower bounds also hold in the model with global communication of agents.

Collaborative exploration has also been studied not only in the context of cover time, but also with different optimization objectives. An exploration strategy for trees with global communication is given in [67], achieving a competitive ratio of $(4 - 2/k)$ for the objective of minimizing the maximum number of edges traversed by an agent. The limited range of an agent or robot can be motivated by energy constraints, resulting from limited battery capacity. In [65] a corresponding lower bound of 3/2 is provided.

Multiple oblivious agents. Most of the aforementioned results considered the case of agents equipped with some internal memory. However, sometimes the assumption that agents can carry some memory along the edges is unrealistic. It is thus desirable to study exploration also in the case of oblivious agents. In such models usually the case of the weakest possible agents is considered. Agents are not only oblivious but also asynchronous and they cannot communicate between each other nor interact with the environment. The only capability of the agents is vision i.e., they can perceive the graph and the positions of the other agents. In contrast to the local models which we predominantly deal with, in the context of oblivious robots, the assumed capability of vision is usually global. With such weak robots, most of the results focus on feasibility of solving the exploration task. Exploration of a ring in the asynchronous Look-Compute-Move model

was studied by Flocchini *et al.* [80] who showed that if k (the number of agents) divides n (the size of the ring), then exploration is impossible for some initial placements of agents. They also presented an algorithm working for any initial configuration if k and n are co-prime and $k \geq 17$. Characterization of pairs (n, k) for which exploration is possible was further studied by Lamani *et al.* [118], who showed that exploration is feasible for all co-prime pairs (n, k) if $k \geq 5$ and is infeasible for $k < 5$ if the size of the ring is even. It was also shown in [80] that $O(\log n)$ agents are always sufficient to explore an n-node ring and $\Omega(\log n)$ agents are sometimes required. In the more powerful relaxed semi-synchronous ATOM model with randomized agents, four agents are sufficient and necessary to explore a ring [52].

Tree exploration in the asynchronous model was studied by Flocchini *et al.* [79], where the authors show that for trees with maximum degree 3 a team of size $O(\log n / \log \log n)$ can always explore the tree and $\Omega(\log n / \log \log n)$ agents are sometimes required. For completely anonymous and unlabelled trees with maximum degree 4 surprisingly a team of $\Omega(n)$ agents is necessary [79]. The problem becomes slightly easier when the graph has locally distinct edge labels (port labelling). In such setting Chalopin *et al.* [29] showed that for $k = 4$ and for any odd $k \geq 5$ exploration is feasible for any asymmetric initial configuration.

1.4.2 State-of-the art on randomized exploration

Deterministic strategies, like DFS, work only for agents with big number of bits of local memory and sometimes require node identifiers. Exploration of anonymous graphs with very little memory ($o(\log n)$ bits) can be achieved using randomized strategies.

Randomized exploration with a single agent. When considering randomized strategies with a single walker, the first idea might be to deploy a random walk. In such a strategy, in every step, an agent located at vertex $u \in V$ chooses a neighbor of u uniformly at random and moves to that vertex. In other words, the transition probability p_{uv} from u to any of its neighbours v is $p_{uv} = \frac{1}{\deg(u)}$, where $\deg(u)$ denotes the degree of u. The transition matrix $\mathbf{M}$ of the random walk is a $|V| \times |V|$ matrix having in the u-th row and v-th column the value p_{uv}. As the number of steps of the random walk tends to infinity, in any non-bipartite graph, the probability of the agent to be located at a specific node u, at any given moment of time converges to the stationary distribution. The stationary distribution of the random walk π is a vector such that $\pi_u = \frac{\deg(u)}{2m}$. This means that the random walk visits the nodes with high degree more often. For bipartite graphs, a stationary distribution can be achieved by adding self-loops, i.e. at any step the agent can stay at a node with some probability.

The cover time $C_{rw}(G, v)$ of the random walk on graph G starting from node v is the expected number of steps until an agent starting from v visits all nodes of G. The cover time $C_{rw}(G)$ of graph G is the maximum of $C_{rw}(G, v)$ taken over all $v \in V$. It is easy to determine the cover time for some graph classes. For example, if G is a clique, then

Graph class	*Result*	*Reference*
General graph	$C_{rw}(G) \geq (1-o(1))n\ln n$ $C_{rw}(G) \leq (1+o(1))\frac{4}{27}n^3$ $C_{rw}(G) \leq 2m(n-1)$	[74] [75] [6]
Regular graph	$C_{rw}(G) \leq 2n^2$	[76]
Random d-regular graphs ($d \geq 3$)	$C_{rw}(G) \leq (1+o(1))\frac{d-1}{d-2}n\ln n$ (w.h.p.)	[33]
Two dimensional grid d-dimensional grid ($d \geq 3$)	$C_{rw}(G) = \Theta(n\log^2 n)$ $C_{rw}(G) = \Theta(n\log n)$	[30, 146] [4]

TABLE 1.1: Known bounds on the cover time of the random walk.

the cover time of G is $C_{rw}(G) = n(\ln n + O(1))$ since the random walk on the clique is closely linked to the coupon collector problem [123]. Known bounds of the cover time for different graph classes are presented in Table 1.1.

The main drawback of the random walk as a strategy for graph exploration is the worst-case cover time of $\Theta(n^3)$, achieved for example for lollipop graphs [123], where a lollipop is defined as a path of length roughly $n/3$ connected to a clique on $2n/3$ nodes. An approach to reduce the $\Theta(n^3)$ maximum cover time for a randomized exploration strategy consists in using biased random walks. In such walks, the transition matrix $\mathbf{M}$ is modified with respect to the simple (unbiased) random walk. In general, the reduction of the cover time is achieved by increasing the probability of transition to a node of lower degree. Transition probabilities that yield a cover time $O(n^2\log n)$ for any graph are shown by Ikeda *et al.* [102]. However, implementing such a strategy requires access at every node to topological information about the degrees of the neighbors. They also showed that for any transition matrix the cover time on the path is $\Omega(n^2)$. A refined strategy [128] based on the so-called Metropolis walks, also achieves $O(n^2\log n)$ cover time, without requiring such extra topological information. Kosowski [111] showed how to implement this idea to obtain an algorithm for mobile agent with $O(\log\log n)$ bits of memory exploring any graph in expected time $O(n^2\log n)$.

Multiple agents. Deploying multiple random walks is another idea to decrease the cover time. When deploying k independent random walks, the cover time is the expected number of steps until each node has been visited by at least one random walk (assuming all agents move simultaneously in synchronised steps). Broder *et al.* [23] considered k random walks where the initial position of each walk was chosen according to the stationary distribution of the random walk. The cover time in such a setting is $O(m^2\log^3 n/k^2)$.

More recently, multiple walks have been studied in a worst-case starting scenario i.e., for initial positions of agents that yield the biggest cover time. Alon *et al.* [7], Efremenko and Reingold [70], and Elsässer and Sauerwald [71] have studied the notion of the *speedup*

$\ln n$ denotes the natural logarithm and $\log n$ the logarithm at base 2.

of the random walk for an undirected graph G, defined as the ratio between the cover time of a single-agent walk in G starting from a worst-case initial position and that of k walks in G for worst-case initial positions of agents, as a function of k. A characterization of the speedup has been achieved for many graph classes, for random graphs, and graphs with special properties, such as small mixing time compared to cover time. However, a central question posed in [7] still remains open: what are the minimum and maximum values of speedup of the random walk in arbitrary graphs? The smallest known value of speedup is $\Theta(\log k)$, attained e.g. for the cycle, while the largest known value is $\Theta(k)$, attained for many graph classes, such as expanders, cliques, and stars.

An interesting problem of searching for a treasure hidden in a grid at distance D from the starting position of the team of k probabilistic agents is studied in [72, 77]. When agents cannot communicate but know the value of k (or a constant approximation), the optimal expected time necessary and sufficient to solve the problem is $\Theta(D + D^2/k)$ [77]. Authors in [77] also show an algorithm with the optimal competitive ratio in the case when agents do not know k. If agents are able to exchange constant-size messages while being at the same location then the problem can be solved in time $\Theta(D + D^2/k)$ w.h.p even if the agents have a constant-size memory and have no knowledge of k [72].

1.5 Related problems for mobile agents

In mobile agent computing, the exploration problem is sometimes a subtask for more complex problems. In other problems only part of the available environment needs to be explored. In this section we want to briefly describe some problems, different than exploration, for multiple mobile agents and recall recent results on these problems.

1.5.1 Rendezvous and Gathering

Another widely studied problem in the mobile agent domain is the *gathering* problem, also known as *rendezvous* (typically for the case of 2 agents). In this problem, two or more agents have to meet, for example to exchange data which they have collected, or to coordinate further actions. The vast literature on this problem includes three surveys [9, 116, 134] and two books [10, 115], and contains numerous results in various models of gathering. Herein, we provide a brief overview of some results in the area.

The gathering in port-labelled networks in the synchronized, deterministic model for agents with distinct identifiers was studied by Dessmark *et al.* [50] and Kowalski and Malinowski in [114]. If we assume that agents present at the same location at the same time are aware of the fact that they have met then the problem of gathering for an arbitrary number of agents is asymptotically no harder than the special case of gathering two agents [114]. For general graphs there exists a gathering algorithm with time complexity polynomial in n, and $\log l$, where l is the value of the smaller identifier [114]. This algorithm works also if the start of agents is not simultaneous, in which case the time of the algorithm is measured between the start of the last agent and

completion of the gathering. The authors of [50, 114] also give results for trees and cycles. When agents have distinct identifiers, gathering is even possible in the asynchronous model [46].

For anonymous agents, the problem of minimizing space instead of time was considered in [44, 89]. Czyzowicz *et al.* [44] showed that it is possible to perform gathering of two agents in any graph if the agents are equipped with memory $O(\log n)$, even with arbitrary delays between the start of the agents. They also showed that $\Omega(\log n)$ memory is needed even if start is simultaneous. The lower bound is already achieved for rings and thus it shows a large gap between the space necessary to complete gathering and exploration (for example for exploration of a port-labeled ring, constant space is sufficient). For trees Fraigniaud and Pelc [89] proved that $\Theta(\log n)$ memory is needed for arbitrary delays and for simultaneous start it is $\Theta(\log L + \log\log n)$, where L is the number of leafs in the tree showing that there is a significant difference between simultaneous and delayed start. Results from [89] work under the assumption that the starting positions of the agents are not perfectly symmetrizable i.e., there does not exist a port-labelling μ of the tree and an automorphism of the tree preserving μ that carries the starting position of one agent on the starting position of the other.

Chalopin *et al.* [25] showed an algorithm for map construction of anonymous undirected, port-labelled graphs, which can be used by two anonymous agents to meet if they are allowed to mark their starting positions. Gathering using the algorithm for map construction is also possible with arbitrary delays and in the asynchronous model.

Gathering has also been considered for directed anonymous port-labelled graphs [28]. The authors show an algorithm for gathering in the model with whiteboards, working for all feasible initial configurations.

In geometric scenarios, gathering of k agents on the plane has received a lot of attention due to its applications in robotics. Ando *et al.* [12] considered this problem in the synchronous model in which each agent observes the other robots and moves simultaneously and instantaneously in discrete time slots. They proposed an algorithm for oblivious agents with limited visibility. This algorithm was shown to work in $O(k^2)$ steps by Degener *et al.* [48], who also showed the corresponding lower bound $\Omega(k^2)$.

In the asynchronous Look-Compute-Move model for oblivious agents with unlimited visibility and with multiplicity detection, the gathering problem on the plane can be solved using an algorithm proposed by Cieliebak *et al.* [31]. If agents share a common sense of direction i.e., the agents share the same coordinate system, the problem can be solved even with limited visibility [82]. A negative result by Prencipe [135] shows that without multiplicity detection, gathering agents that do not have common sense of direction is impossible. In discrete environment, on the ring, characterisation of the initial configurations which admit a gathering for agents with unlimited visibility and multiplicity detection is given in [108, 109].

Gathering on the plane with faulty agents was studied by Agmon and Peleg [1]. They showed that in the semi-synchrounous model it is possible to gather $k \geq 3$ agents in the

presence of one crash fault. In the presence of f Byzantine faults an algorithm in the fully-synchronous model is presented (assuming that $k \geq 3f + 1$).

1.5.2 Black hole search

The exploration problem with mobile agents is often studied from the perspective of applications in robotics. In practical scenarios, agents are often required to explore dangerous zones. Thus, a variant of graph exploration that has gained a lot of attention from theoreticians in recent years is the exploration of a graph with *black holes*, i.e., nodes that destroy any entering agent. The objective in this problem is usually the minimization of the number of agents necessary to complete the task. The only way to locate the black hole is to enter the node containing it. The agent which enters the black hole disappears without leaving any trace. Since the other agents must locate the hole, the model must provide some means of communication or some form of visibility, so that the disappearing agent can somehow notify other agents before entering the black hole. The problem of locating a single black hole is considered in different communication models.

One model of communication is the whiteboard model, in which agents can leave messages at nodes. For undirected networks and asynchronous agents communicating by means of whiteboards, Dobrev *et al.* [60] showed that $\Delta + 1$ agents are necessary and sufficient and number of time steps to complete the task is $\Theta(n^2)$, where Δ is the total number of edges leading into the black hole in the graph. If the agents are given the map of the graph, and the goal is to identify which of the nodes on the map is the black hole, then the problem can be solved for any graph using only two agents in time $O(n \log n)$ [60]. This complexity cannot be improved for the case of the ring as Dobrev *et al.* [61] showed a $\Omega(n \log n)$ lower bound. However, the problem can be solved in time $\Theta(n)$ for many graph classes like hypercubes, cube-connected cycles, star graphs, wrapped butterflies, chordal rings, multidimensional meshes and tori of restricted diameter [59].

A weaker model of communication allows agents to leave tokens at nodes instead of writing to whiteboards. However, if the topology of the graph is known, then weakening the communication model does not make the problem harder as in such a scenario $\Theta(n \log n)$ moves are sufficient even if each agent has only one pebble [81]. A more difficult scenario, where agents are initially located at different positions and are equipped with a constant number of bits of memory, was studied in rings [27] and tori [26].

The black hole search task is also feasible if agents cannot communicate but operate in the synchronous model and are aware if there is any other agent at the same node at the same step. The problem of searching for one black hole with two agents in the synchronous model assuming that the map of the graph is given to the agents in advance is considered in [36, 45, 110]. The goal in such a setting is a time-efficient algorithm for locating the black hole. For synchronous agents in trees, Czyzowicz *et al.* [45] proved that two

agents suffice to identify the location of the black hole by showing a 5/3-approximation algorithm. For general graphs the problem was shown to be APX-hard [110] and a $3\frac{3}{8}$-approximation was presented in [36]. For a more general problem, where agents are also given a subset of nodes of the network initially known to be safe, Klasing *et al.* [110] showed that for general undirected graphs, the problem is not approximable within any constant factor less than $\frac{389}{388}$ and showed a 6-approximation algorithm.

Locating b holes with k agents which can communicate when being located at the same node of an undirected graph is considered in [35, 37]. In [35] authors show that time $O((n/k)\log n/\log\log n + bD_b)$, where D_b is the maximum diameter of the graph after removing at most b nodes, is always sufficient, and that $\Omega(n/k + D_b)$ is sometimes necessary to locate all black holes. Cooper *et al.* [37] considered the model with an additional feature that an agent entering the black hole disappears but also repairs the black hole. When the graph contains at most $k/2$ black holes, the authors showed that $O(n/k + D\log f/\log\log f)$ steps, where $f = \min\{n/k, n/D\}$, are always sufficient and $\Omega(n/k + D\log f/\log\log f)$ steps are sometimes necessary to repair all black holes.

To locate a single black hole in the synchronous model, there is an exponential gap in the necessary number of agents between the case of directed and undirected graphs. For undirected graphs $\Delta + 1$ agents are always sufficient, where Δ is the degree of the black hole [60]. For directed graphs considerably more agents are needed as Czyzowicz *et al.* [41] showed that 2^Δ agents are sometimes required, whereas Kosowski *et al.* [113] showed that $\Delta 2^\Delta$ agents always suffice.

1.5.3 Graph searching

A very widely studied problem in the context of mobile entities in a graph is the graph searching problem. This problem is a game between a team of searchers and a fugitive. The goal of searchers is to find the fugitive who is trying to escape.

The first formal formulation of this problem was made by Parson [133]. In this formulation, the fugitive is assumed to be infinitely fast thus the problem can be seen as graph cleaning. Initially the fugitive can be at any place and thus each edge is considered as contaminated. Searchers are allowed to *clean* edges by traversing them. However, if at any time there exists an unguarded path (i.e., without a searcher located at some node of the path) between some contaminated edge and a clean edge, then the clean edge immediately becomes contaminated again. The *edge search number* of a graph G is the smallest number of searchers that guarantee to clear all edges. It was shown in [120] that for any graph there exists a strategy using the minimum number of searchers in which recontamination does not occur. Finding the edge search number for a given graph is shown to be NP-complete [124], however it can be computed efficiently for trees [124].

One of the variants of graph searching is called the *cops and robbers* problem. This variant is a two-player game with complete information. At the beginning of the game, players freely choose their starting vertices, with the cops choosing first and the robber second. During the game, players make moves alternatively. In the cops' move, each

cop can remain idle or move to an adjacent vertex, likewise in the robber's move, the robber can either stay or move to adjacent vertex. The goal of the cops is to capture the robber by moving at least one cop to a vertex occupied by the robber. The goal of the robber is to remain uncaptured forever. The problem of deciding which player has the winning strategy was recently shown to be EXPTIME-complete by Kinnersley [106]. For a given graph G, the minimum number of cops which guarantees the cops' winning strategy, is called the cop number of G. All the graphs with cop number equal to 1 were characterized independently by Nowakowski and Winkler [129] and Quilliot [137]. However, a characterization of graphs with cop number k for $k > 1$ remains open. The cop number was shown to be at most 3 for all planar graphs [2]. A vast bibliography on this topic has been organized by Fomin and Thilikos [84].

1.5.4 Pattern formation

A highly relevant problem for mobile agents on the plane is that of pattern formation. In this problem, a team of agents has to organize themselves, usually without synchronization, to form a specific pattern. This problem can be seen as a generalization of the gathering problem, because gathering is simply forming a single point by all agents. Usually the agents considered in this context are oblivious and unable to communicate directly.

The pattern formation problem was studied in the case when agents share the knowledge of the chirality of the system (i.e., clockwise/counterclockwise orientation) by Suzuki and Yamashita [142] for agents with unbounded memory. They showed an algorithm for forming a circle and gave a characterization of patterns that can be formed. Défago and Souissi [47] in the same model showed an algorithm for circle formation for agents without memory. Dieudonné and Petit [53] showed the same result for fully asynchronous agents without memory.

Flocchini *et al.* [83] studied the problem of formation of an arbitrary pattern in the asynchronous model and with oblivious agents. They showed that when agents share the knowledge of the direction and orientation of both axes (North-South and East-West), the problem can be solved for any pattern. In contrast, if no common direction is known to the agents, the general problem cannot be solved in that model. They also studied the case when robots know the direction and orientation of one single axis. Then pattern formation can be accomplished whenever the number of agents is odd.

1.5.5 Polygon mapping

In the problem of polygon mapping, the task of the mobile agent is to construct the visibility graph of the polygon in which the agent is deployed. Nodes of the visibility graph correspond to nodes of the polygon and edges of the visibility graph connect pairs of nodes which can be connected by a line segment without crossing any side of the polygon. The feasibility of solving the problem depends on capabilities of sensors with which the agent is equipped. The agent is endowed with a visibility sensor and can see

all visible points of the polygon (a point p is visible if a segment connecting p and the current position of the agent does not cross the border of the polygon).

When the agent can also measure angles between visible vertices of the polygon, the problem is solvable even if the agent can walk only on the border of the polygon [57, 58]. When the agent can walk inside the polygon, the ability of the agent to tell if the visible angle is convex or reflex is sufficient to solve the problem if the agent is given an approximation on the number of vertices of the polygon [24]. Also, Disser *et al.* [56] showed that such an approximation allows the agent equipped with a compass to solve the problem. However, the question whether an agent which can measure distances to all visible vertices of the polygon can reconstruct the visibility graph, remains open. More results on polygon mapping in various models can be found in [55].

1.5.6 Patrolling

Patrolling of a continuous environment is a natural task for multiple autonomous mobile agents. Patrolling problems can be seen as a task of perpetual exploration of a terrain, with the goal of detecting potential intruders. A typical objective is to minimize the so-called *idle time* of a patrolling strategy, i.e., the maximum time between two consecutive visits to any point of the environment. In this context, the problem of patrolling an interval was considered in [42, 104]. *The proportional strategy* proposed in [42] involves partitioning the interval into disjoint parts monitored by individual agents. Such a strategy is optimal for agents with equal speeds and for at most 3 agents with different speeds [104]. Surprisingly, the proportional strategy turns out not to be optimal for at least 6 agents with different speeds [104]. For a more general overview of exploration-type problems in robotics applications we refer the reader to [68].

Chapter 2

Exploration with communicating agents

In this chapter, we will consider collaborative exploration using a team of agents that are allowed to interact between each other in order to coordinate their decisions. Agents have unique identifiers, which allow agents located at the same node and having the same exploration history to differentiate their actions. Agents are initially placed at the same node r. The considered model is the same as in [67, 85], but a smaller number of agents was assumed there. We consider two communication models: one in which all agents have global knowledge of the state of the exploration (*global communication*), and one in which agents may only exchange information when simultaneously located at the same vertex (*local communication*). As the main result of this chapter, we provide a strategy which performs exploration of a graph with n vertices and diameter D, in time $O(D)$, using a team of agents of polynomial size $k = Dn^{1+\epsilon} < n^{2+\epsilon}$, for any $\epsilon > 0$. The strategy works in the local communication model, without knowledge of global parameters such as n or D. We also obtain almost-tight bounds on the asymptotic relation between exploration time and team size, for large k. Let D^* be the distance from the starting vertex r to the most distant vertex of the graph. For any constant $c > 1$, we show that in the global communication model, a team of $k = D^*n^c$ agents can always complete exploration in $D^*(1 + \frac{1}{c-1} + o(1))$ time steps, whereas at least $D^*(1 + \frac{1}{c} - o(1))$ steps are sometimes required. In the local communication model, $D^*(1 + \frac{2}{c-1} + o(1))$ steps always suffice to complete exploration, and at least $D^*(1 + \frac{2}{c} - o(1))$ steps are sometimes required. This shows a clear separation between the global and local communication models. Interestingly, the presented algorithms achieve a constant competitive ratio and therefore show that the lower bound $\Omega(\log k / \log\log k)$ on the competitive ratio given in [67] does not hold for large number of agents.

The results presented in this chapter were published in [T2].

2.1 The team exploration model

Exploring an undirected graph-like environment is relatively straightforward for a single agent. Assuming the agent is able to distinguish which neighboring vertices it has previously visited, there is no better (in terms of the exploration time) systematic traversal strategy than a simple depth-first search of the graph, which takes $2(n-1)$ moves in total for a graph with n vertices. The situation becomes more interesting if multiple agents want to collectively explore the graph starting from a common location. If arbitrarily many agents may be used, then we can generously send n^{D^*} agents through the graph.

While the cases with one agent and arbitrarily many agents are both easy to understand, it is much harder to analyze the spectrum in between these two extremes. Of course, we would like to explore graphs in as few steps as possible (i.e., close to D^*), while using a team of as few agents as possible. In this chapter we study this trade-off between exploration time and team size. A trivial lower bound on the number of steps required for exploration with k agents is $\Omega(D + n/k)$ (see Proposition 2.1). We look at the case of larger groups of agents, for which D is the dominant factor in this lower bound. This complements previous research on the topic for trees [66, 85] and grids [130], which usually focused on the case of small groups of agents (when n/k is dominant).

Another important question when considering collaborating agents concerns the model that is assumed for the communication between agents. We need to allow communication to a certain degree, as otherwise there is no benefit from using multiple agents for exploration [85]. We may, for example, allow agents to freely communicate with each other, independent of their whereabouts, or we may restrict the exchange of information to agents located at the same location, or we may make a compromise between the two. In this chapter, we also study this tradeoff between global and local communication.

2.1.1 The collaborative online graph exploration problem

We are given a graph $G = (V, E)$ rooted at some vertex r. By D we denote the diameter of the graph and by D^* we denote the distance from r to the most distant vertex in the graph. Note that $D^* \leq D \leq 2D^*$, thus when using asymptotic notation O or Ω we do not need to distinguish between D and D^*. But when bounding the precise number of steps in the algorithm, we need to make that distinction. The number of vertices of the graph is bounded by n. Initially, a set $\mathcal{A}$ of k agents is located at r. We recall that a strategy collaboratively explores the graph G in t time steps if for all $v \in V$ there exists a time step $s \leq t$ and an agent $g \in \mathcal{A}$, such that g is located at v in step s. Our goal is to find an exploration strategy which minimizes the time it takes to collaboratively explore a graph in the worst case, with respect to the parameter D^*. We assume that vertices have unique identifiers that admit a total ordering. In each step, an agent visiting vertex v receives a complete list of the identifiers of the nodes in $\Gamma(v)$, where $\Gamma(v)$ is the neighborhood of v. Time is discretized into steps, and in each step, an agent can either

stay at its current vertex or slide along an edge to a neighboring vertex. Agents have unique identifiers, which allow agents located at the same node and having the same exploration history to differentiate their actions. We do not explicitly bound the memory resources of agents, enabling them in particular to construct a map of the previously visited subgraph, and to remember this information between time steps.

We distinguish between the following communication models.

Global communication In this model, we assume that, at the end of each step s, all agents have complete knowledge of the explored subgraph i.e., the subgraph which has been visited by agents up to step s inclusive. In particular, in step s all agents know the number of edges incident to each vertex of the explored subgraph which lead to unexplored vertices, but they have no information on any subgraph consisting of unexplored vertices.

Whiteboards In this scenario, robots can communicate by writing previously acquired information at the currently visited node and reading information available at this node.

Local communication In this model, two agents can exchange information only if they occupy the same vertex. The information that is exchanged includes the subgraph that each of these agents has explored itself and the information it has received from other agents prior to the current meeting. Thus, each agent g has its own view on the vertices that were explored so far, based only on the knowledge that originates from the agent's own observations and from other agents that it has met so far.

No communication In this model, agents are not allowed to communicate or to interact with the environment in any way.

All these communication models are present in the literature [67, 85]. In this chapter we will consider only global and local communication. Observe that global communication is the strongest model because, given global communication, it is possible to simulate all other models. In the model with whiteboards it is possible to simulate local communication, thus whiteboards are the second strongest model. Clearly the model without communication is the weakest model but Fraigniaud *et al.* [85] showed that in this model there is no parallelization for some graph classes. Thus, local communication is the weakest known communication model in which collaborative exploration can be performed faster than single-agent exploration.

The main contribution of this chapter is an exploration strategy for a team of polynomial size to explore graphs in an asymptotically optimal number of steps. More precisely, for any $\epsilon > 0$, the strategy can operate with $D^* n^{1+\epsilon} < n^{2+\epsilon}$ agents and takes time $O(D)$. It works even under the local communication model and without prior knowledge of n or D.

Communication Model	*Upper bound*	*Lower bound*
Global communication:	$D^* \cdot (1 + \frac{1}{c-1} + o(1))$ Theorem 2.9	$D^* \cdot (1 + \frac{1}{c} - o(1))$ Theorem 2.11
Local communication :	$D^* \cdot (1 + \frac{2}{c-1} + o(1))$ Theorem 2.9	$D^* \cdot (1 + \frac{2}{c} - o(1))$ Theorem 2.11

TABLE 2.1: Our bounds on the time required to explore general graphs using $D^* n^c$ agents. The same upper and lower bounds hold for trees. The lower bounds use graphs with $D^* = n^{o(1)}$.

We first restrict ourselves to the exploration of trees (Section 2.2). We show that with global communication, trees can be explored in time $D^* \cdot (1 + 1/(c-1) + o(1))$, for any $c > 1$, using a team of $D^* n^c$ agents. Our approach can be adapted to show that with local communication trees can be explored in time $D^* \cdot (1 + 2/(c-1) + o(1))$, for any $c > 1$, using the same number of agents. We then carry the results for trees over to the exploration of general graphs (Section 2.3). For graph exploration with $D^* n^c$ agents we obtain precisely the same asymptotic time bounds as for the case of trees, under both communication models. The limit of our approach in terms of the smallest allowed team of agents is a team of $k = (2 + \epsilon) n D^*$ agents exploring graphs in time $\Theta(D \log n)$, with local communication, for any constant $\epsilon > 0$.

Finally, we provide lower bounds for collaborative graph exploration that almost match our positive results (Section 2.4). More precisely, we show that, in the worst case and for any $c > 1$, exploring a graph with $D^* n^c$ agents takes at least $D^* \cdot (1 + 1/c - o(1))$ time steps in the global communication model, and at least $D^* \cdot (1 + 2/c - o(1))$ time steps in the local communication model. Table 2.1 summarizes our upper bounds and corresponding lower bounds.

We remark that in the offline case, the optimal collaborative exploration algorithm works in time $\Theta(D + n/k)$, hence the smallest team which explores a graph in $\Theta(D^*)$ steps in the offline model has $\Theta(n/D)$ agents. We include the proof of this fact for completeness.

Proposition 2.1. *An optimal offline collaborative exploration of a graph G and diameter D, using k agents located initially at the same node r requires time $\Theta(D + n/k)$.*

Proof. To prove the lower bound $\Omega(D + n/k)$ observe that there exists a node at distance at least D^* from r, thus time $\Omega(D)$ is necessary. Since we have k agents, at most k new nodes can be explored in any time step. Thus, time $\Omega(n/k)$ is also necessary.

To prove the upper bound we construct the following algorithm. First, find the BFS (Breadth First Search) tree T of graph G, rooted at r. Next, construct a cycle of edges traversing every edge of tree T twice (i.e. Eulerian traversal of T with all edges doubled). Since tree T has $n - 1$ edges, such cycle will have $2(n - 1)$ edges. Each agent selects a destination vertex on the cycle such that the destination vertices are

equidistant on the cycle. By equidistant we mean that the distance on the cycle between two consecutive destination vertices is $\lceil 2(n-1)/k \rceil$ or $\lfloor 2(n-1)/k \rfloor$. Note that since some nodes of the graph appear multiple times in the cycle, some agents may have the same destination. Next, each agent moves to its destination. Since each vertex is at distance at most D^* from r, this can be achieved in time at most D^*. Next, each agent follows the cycle for $\lceil 2(n-1)/k \rceil$ steps in the same direction. Since destination vertices are at most $\lceil 2(n-1)/k \rceil$ edges apart on the cycle, agents will traverse all edges of the cycle. Since the cycle contains each node of the graph at least once, such an algorithm will explore every node of the graph. The total number of required time steps is bounded by $D^* + \lceil 2(n-1)/k \rceil$ □

2.2 Tree exploration

We start our considerations by designing online exploration strategies for the special case when the explored graph is a tree T rooted at a vertex r.

For any exploration strategy, the set of all encountered vertices (i.e., all visited vertices and their neighbors) at the beginning of step $s = 1, 2, 3, \ldots$ forms a connected subtree of T, rooted at r and denoted by $T^{(s)}$. In particular, $T^{(1)}$ is the vertex r together with its children, which have not yet been visited. For $v \in V(T)$ we write $T^{(s)}(v)$ to denote the subtree of $T^{(s)}$ rooted at v. We denote by $L(T^{(s)}, v)$ the number of leaves of the tree $T^{(s)}(v)$. Note that $L(T^{(s)}, v) \leq L(T^{(s+1)}, v)$ because each leaf in $T^{(s)}(v)$ is either a leaf of the tree $T^{(s+1)}$ or the root of a subtree containing at least one vertex. If v is an unencountered vertex at the beginning of step s, i.e., its parent was not yet visited, we define $L(T^{(s)}, v) = 1$.

2.2.1 Tree exploration with global communication

We are ready to give the procedure `TEG` (*Tree Exploration with Global Communication*). The pseudocode uses the command "move$^{(s)}$", describing the move to be performed by each agent, specifying the destination at which the agent appears at the start of time step $s+1$. Since the agents can communicate globally, the procedure can centrally coordinate the movements of each agent. For simplicity we assume that x agents spawn in r in each time step, for some given value of x. Then, the total number of agents used after l steps is simply lx.

Procedure TEG (tree T with root r, integer x) **at time step** s:
Place x new agents at r.
for each $v \in V(T^{(s)})$ which is not a leaf **do**: { determine moves of the agents located at v }
Let $\mathcal{A}_v^{(s)}$ be the set of agents currently located at v.
Denote by $v_1, v_2, \ldots, v_d$ the set of children of v.
Let $i^* := \arg\max_i \{L(T^{(s)}, v_i)\}$. { v_{i^*} is the child of v with the largest value of L }
Partition $\mathcal{A}_v^{(s)}$ into disjoint sets $\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, \ldots, \mathcal{A}_{v_d}$, such that:
(*i*) $|\mathcal{A}_{v_i}| = \left\lfloor \frac{|\mathcal{A}_v^{(s)}| \cdot L(T^{(s)}, v_i)}{L(T^{(s)}, v)} \right\rfloor$, for $i \in \{1, 2, \ldots, d\} \setminus \{i^*\}$,
(*ii*) $|\mathcal{A}_{v_{i^*}}| = |\mathcal{A}_v^{(s)}| - \sum_{i \in \{1,2,\ldots,d\} \setminus \{i^*\}} |\mathcal{A}_{v_i}|$.
for each $i \in \{1, 2, \ldots, d\}$ **do for each** agent $g \in \mathcal{A}_{v_i}$ **do move**$^{(s)}$ g to vertex v_i.
end for
end procedure TEG.

The following lemma provides a characterization of the tradeoff between exploration time and the number of agents x released at every round in procedure TEG. In the following, all logarithms are in base 2 unless a different base is explicitly given.

Lemma 2.2. *In the global communication model, procedure* TEG *with parameter x explores any rooted tree T in at most $D^* \cdot (1 + \frac{\log n}{\log(x/n)})$ time steps, for $x > n$.*

Proof. Fix any leaf f of the tree T. We want to prove that procedure TEG visits the leaf f after at most $D^* \cdot (1 + \frac{\log n}{\log(x/n)})$ time steps. Take the path $\mathcal{F} = (f_0, f_1, f_2, \ldots, f_{D_f})$ from r to f in T, where $r = f_0$,$f = f_{D_f}$, and $D_f \leq D^*$. We define the *wave* of agents w_s starting from r at time s and traversing the path $\mathcal{F}$ as the maximum sequence of the non-empty sets of agents which leave the root in step s and traverse edges of $\mathcal{F}$ in successive time steps, i.e., $w_s = (\mathcal{A}_{f_0}^{(s)}, \mathcal{A}_{f_1}^{(s+1)}, \ldots)$, where we use the notation from procedure TEG. The size of wave w_s in step $s+t$ is defined to be $|\mathcal{A}_{f_t}^{(s+t)}|$, i.e., the number of exploring agents located at vertex f_t at the beginning of time step $s+t$; initially, every wave has size $|\mathcal{A}_{f_0}^{(s)}| = x$. Note that each agent in $\mathcal{A}_{f_i}^{(s+i)}$, $0 \leq i < D_f$, is located at r at the start of time step s. We denote the number of leaves in the subtree of $T^{(i)}$ rooted at f_j by $\lambda_j^{(i)} = L(T^{(i)}, f_j)$. Recall that if f_j is not yet discovered in step i, by definition of the function L, we have $\lambda_j^{(i)} = 1$. In general, $1 \leq \lambda_j^{(i)} \leq n$. We define

$$\alpha_i = x \frac{\lambda_1^{(i)}}{\lambda_0^{(i)}} \frac{\lambda_2^{(i+1)}}{\lambda_1^{(i+1)}} \cdots \frac{\lambda_{D_f}^{(i+D_f-1)}}{\lambda_{D_f-1}^{(i+D_f-1)}}.$$

We define the value α_i^* as the number of agents of the i-th wave that reach the leaf f, i.e., the size of the i-th wave in step $i + D_f$. If $\alpha_1^* = \alpha_2^* = \cdots = \alpha_{i-1}^* = 0$ and $\alpha_i^* \geq 1$ for some time step i, then we say that leaf f is explored by the i-th wave. Before we proceed with the analysis, we show the following auxiliary claim.

Claim (*). For every i, if $\alpha_i \geq 1$ then $\alpha_i^* > \alpha_i - 1$, and thus $\alpha_i - 1$ is a lower bound on the number of agents reaching f in step $i + D_f - 1$.

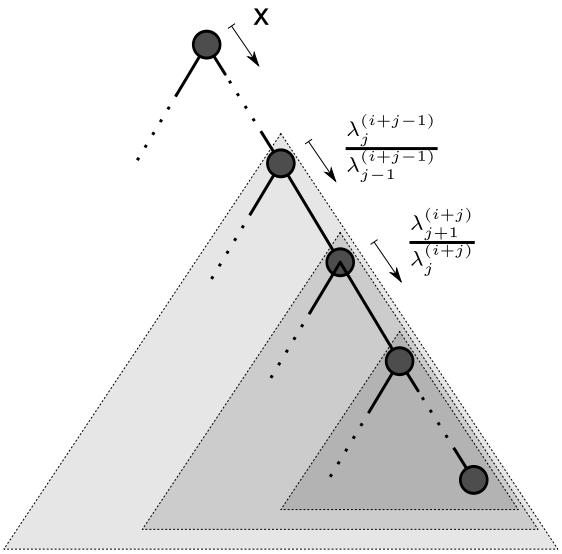

FIGURE 2.1: Illustration of proof of Lemma 2.2: computation of the value of α_i for a wave of agents descending in tree T

Proof (of the claim). We define $c_j = \lambda_{j+1}^{(i+j)}/\lambda_j^{(i+j)}$ for $j = 0, \ldots, D_f - 1$. For $i \geq 1$ we have

$$\alpha_i = x \prod_{j=0}^{D_f-1} c_j.$$

Since $c_j \leq 1$ for all j and since $\alpha_i \geq 1$, there exist at most $\log x$ different j such that $c_j \leq 1/2$. Denote the set of all such j by $\mathcal{J}$, with $|\mathcal{J}| \leq \log x$. Also, denote the size of wave w_i in step $i + s$ by a_s (for $s = 0, 1, 2, \ldots$), in particular $a_0 = x$.

Consider some index s for which $c_s > 1/2$. We have $\lambda_{s+1}^{(i+s)}/\lambda_s^{(i+s)} > 1/2$, thus more than half of all leaves of the tree $T^{(i+s)}(f_s)$ also belong to the tree $T^{(i+s)}(f_{s+1})$. But then, in time step $i + s + 1$, agents are sent from f_s to f_{s+1} according to the definition in expression (ii) in procedure `TEG`. Thus, we can lower-bound the size of wave w_i in step $i + s + 1$ by $a_{s+1} \geq a_s c_s$. Otherwise, if $c_s \leq 1/2$ (i.e., if $s \in \mathcal{J}$), then agents are sent according the definition in expression (i) in procedure `TEG`, and hence $a_{s+1} \geq \lfloor a_s c_s \rfloor$. Note that these bounds also hold if there are no agents left in the wave, i.e., $a_s = a_{s+1} = 0$. Thus, we have:

$$a_{s+1} \geq a_s c_s - \delta_s, \quad \text{where } \delta_s = \begin{cases} 1, & \text{if } s \in \mathcal{J}, \\ 0, & \text{otherwise.} \end{cases}$$

Denote consecutive elements of $\mathcal{J}$, $s_1 < s_2 < s_3 < \cdots < s_{|\mathcal{J}|}$. In this way we expand the expression for $\alpha_i^* = a_{D_f}$:

$$\alpha_i^* = a_{D_f} \geq a_{D_f-1} c_{D_f-1} - \delta_{D_f-1} \geq \ldots \geq (\ldots((a_0 c_0 - \delta_0)c_1 - \delta_1)c_2 - \ldots)c_{D_f-1} - \delta_{D_f-1} =$$

$$= x \prod_{j=0}^{D_f-1} c_j - \sum_{j=0}^{D_f-1} \left(\delta_j \prod_{p=j+1}^{D_f-1} c_j \right) \geq \alpha_i - \sum_{i=1}^{|\mathcal{J}|} \prod_{p=s_i+1}^{D_f-1} c_p \geq \alpha_i - \sum_{i=1}^{|\mathcal{J}|} 2^{-|\mathcal{J}|+i-1} > \alpha_i - 1,$$

where in the last transformations we have taken into account that in the product $\prod_{p=s_i+1}^{D_f-1} c_p$ there are $|\mathcal{J}| - (i+1)$ elements c_p belonging to the set of indices $\mathcal{J}$, which are less or equal than $1/2$. We have obtained $\alpha_i^* > \alpha_i - 1$, which completes the proof of the claim.

We now show that if the number of waves a in the execution of the procedure is sufficiently large, then there exists an index $i \leq a$, such that $\alpha_i \geq 1$. Thus, taking into account Claim (*), leaf f is explored at the latest by the a-th wave.

Take a waves and consider the product $\prod_{i=1}^{a} \alpha_i$. Note that $\lambda_{D_f}^{(s)} = 1$ for every s. Thus, simplifying the product of all α_i by shortening repeating terms in numerators and denominators, and using $1 \leq \lambda_j^{(i)} \leq n$, we get

$$\prod_{i=1}^{a} \alpha_i = x^a \prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \frac{\lambda_{j+1}^{(i+j)}}{\lambda_j^{(i+j)}} = x^a \frac{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_{j+1}^{(i+j)}}{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_j^{(i+j)}} = x^a \frac{\prod_{i'=0}^{a-1} \prod_{j'=1}^{D_f} \lambda_{j'}^{(i'+j')}}{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_j^{(i+j)}} =$$

$$= x^a \frac{\left(\prod_{j'=1}^{D_f} \lambda_{j'}^{(j')}\right) \left(\prod_{i'=1}^{a-1} \prod_{j'=1}^{D_f-1} \lambda_{j'}^{(i'+j')}\right) \left(\prod_{i'=1}^{a-1} \lambda_{D_f}^{(i'+D_f)}\right)}{\left(\prod_{i=1}^{a} \lambda_0^{(i)}\right) \left(\prod_{i=1}^{a-1} \prod_{j=1}^{D_f-1} \lambda_j^{(i+j)}\right) \left(\prod_{j=1}^{D_f-1} \lambda_j^{(a+j)}\right)} \geq \frac{x^a}{n^a n^{D_f-1}} \geq \frac{x^a}{n^{a+D^*}}. \tag{2.1}$$

We want to find a, such that

$$\prod_{i=1}^{a} \alpha_i \geq 1.$$

Taking into account (2.1), it is sufficient to find a satisfying

$$\frac{x^a}{n^{a+D^*}} \geq 1,$$

which for $x > n$ can be equivalently transformed by taking logarithms and elementary arithmetic to the form

$$a \geq \frac{D^* \log n}{\log(x/n)}.$$

Hence, for $a = \lceil \frac{D^* \log n}{\log(x/n)} \rceil$, we have that there exists some i such that $\alpha_i \geq 1$. For the same i we have $\alpha_i^* > \alpha_i - 1 \geq 0$, by Claim (*), which implies that $\alpha_i^* \geq 1$, since α_i^* is an integer. Thus, a waves are sufficient to explore the path $\mathcal{F}$. This analysis can be done for any leaf f, thus it is enough to send a waves in order to explore the graph G. Considering that a wave w_i is completed by the end of step $D^* + i - 1$, the exploration takes at most $D^* + a - 1$ time steps in total. Thus, the exploration takes at most $D^* \cdot (1 + \frac{\log n}{\log(x/n)})$ time steps. □

We remark that in the above lemma, the total number of agents used throughout all steps of procedure `TEG` is $x \cdot D^* \cdot (1 + \frac{\log n}{\log(x/n)})$. For any $c > 1$, by appropriately setting $x = \Theta(n^c)$, we obtain the following theorem.

Theorem 2.3. *For any fixed $c > 1$ and known n, the online tree exploration problem with global communication can be solved in at most $D^* \cdot \left(1 + \frac{1}{c-1} + o(1)\right)$ time steps using a team of $k \geq D^* n^c$ agents.*

In the following we propose a strategy for tree exploration under the local communication model. In the implementation of the algorithm we assume that whenever two agents meet, they exchange all information they possess about the tree. Thus, after the meeting, the knowledge about the explored vertices and their neighborhoods, is a union of the knowledge of the two agents before the meeting. Since agents exchange information only if they occupy the same vertex, at any time s, the explored tree $T^{(s)}$ may only partially be known to each agent, with different agents possibly knowing different subtrees of $T^{(s)}$.

In order to obtain a procedure for the local communication model, we modify procedure `TEG` from the previous section. Observe that in procedure `TEG`, agents never move towards the root of the tree, hence, in the local communication model, agents cannot exchange information with other agents located closer to the root. The new strategy is given by the procedure `TEL` (*Tree Exploration with Local Communication*).

Procedure `TEL` (tree T with root r, integer x) **at time step** s:
  Place x new agents at r in state "exploring".
  for each $v \in V(T^{(s)})$ which is not a leaf **do**: { determine moves of the agents at v }
    if $v \neq r$ **then for each** agent g at v in state "notifying" **do move**$^{(s)}$ g to the parent of v.
    if v contains at least two agents in state "exploring" **and** agents at v do not have information of any agent which visited v before step s **then**:
      { send two new notifying agents back to the root from newly explored vertex v }
      Select two agents g^*, g^{**} at v in state "exploring".
      Change state to "notifying" for agents g^* and g^{**}.
      move$^{(s)}$ g^* to the parent of v. { g^{**} will move to the parent one step later }
    end if
    Let $\mathcal{A}_v^{(s)}$ be the set of all remaining agents in state "exploring" located at v.
    Denote by $v_1, v_2, \ldots, v_d$ all children of v, and by δ the distance from r to v.
    $s' := \lfloor \frac{\delta+s}{2} \rfloor$. { s' is a time in the past such that $T^{(s')}(v)$ is known to the agents at v }
    Let $i^* := \arg\max_i \{L(T^{(s')}, v_i)\}$. { v_{i^*} is the child of v with the largest value of L }
    Partition $\mathcal{A}_v^{(s)}$ into disjoint sets $\mathcal{A}_{v_1}, \mathcal{A}_{v_2}, \ldots, \mathcal{A}_{v_d}$, such that:
    (i) $|\mathcal{A}_{v_i}| = \left\lfloor \frac{|\mathcal{A}_v^{(s)}| \cdot L(T^{(s')}, v_i)}{L(T^{(s')}, v)} \right\rfloor$, for $i \in \{1, 2, \ldots, d\} \setminus \{i^*\}$,
    (ii) $|\mathcal{A}_{v_{i^*}}| = |\mathcal{A}_v^{(s)}| - \sum_{i \in \{1,2,\ldots,d\} \setminus \{i^*\}} |\mathcal{A}_{v_i}|$.
    for each $i \in \{1, 2, \ldots, d\}$ **do if** $|\mathcal{A}_{v_i}| \geq 2$ **then for each** agent $g \in \mathcal{A}_{v_i}$ **do move**$^{(s)}$ g to v_i.
    for each $i \in \{1, 2, \ldots, d\}$ **do if** $|\mathcal{A}_{v_i}| = 1$ **then** change state to "discarded" for the agent in $\mathcal{A}_{v_i}$.
  end for
  for each $v \in V(T^{(s)})$ which is a leaf **do move**$^{(s)}$ all agents located at v to the parent of v.
end procedure `TEL`.

In procedure `TEL`, all agents are associated with a state flag which may be set either to the value "exploring", "notifying" or "discarded". Agents in the "exploring" state

act similarly as in global exploration, with the requirement that they always move to a vertex in groups of 2 or more agents. Every time a group of "exploring" agents visits a new vertex, it detaches two of its agents, changes their state to "notifying", and sends them back along the path leading back to the root. These agents notify every agent they encounter on their way about the discovery of the new vertices. Although information about the discovery may be delayed, in every step s, all agents at vertex v know the entire subtree $T^{(s')}(v)$ which was explored until some previous time step $s' \leq s$. The state flag also has a third state, "discarded", which is assigned to agents no longer used in the exploration process.

The formulation of procedure TEL is not given from the perspective of individual agents, however, based on its description, the decision on what move to make in the current step can be made by each individual agent. The correctness of the definition of the procedure relies on the following lemma, which guarantees that for a certain value s' the tree $T^{(s')}(v)$ is known to all agents at v.

Lemma 2.4. *Let T be a tree rooted at some vertex r and let v be a vertex with distance δ to r. After running procedure* TEL *until time step s, all agents which are located at vertex v at the start of time step s know the tree $T^{(s')}(v)$, for $s' = \left\lfloor \frac{\delta+s}{2} \right\rfloor$.*

Proof. Suppose the claim of the lemma holds until time step $s-1$, i.e., procedure TEL is well defined until time step $s-1$.

Assume that agents following procedure TEL discover vertex v^* in the subtree of v at distance δ^* from v at the beginning of time step $s^* \leq s$. This means that the parent of v^* is visited at the beginning of s^* and notifying agents sent from the parent of v^* carry knowledge about v^* towards the root. We need to prove that that if $s^* \leq s'$ (i.e., if $v^* \in V(T^{(s')})$), then agents located at v at time s know of v^*. It suffices to show that, by the start of time step s, these agents have met a notifying agent (as defined in procedure TEL) coming from the parent of v^*.

Since the distance from the root to the parent of v^* is $\delta+\delta^*-1$, we have $s^* \geq \delta+\delta^*-1$. Thus:

$$\frac{\delta+s}{2} \geq s' \geq s^* \implies s \geq 2s^* - \delta \geq s^* + \delta^* - 1.$$

Since $s \geq s^* + \delta^* - 1$, the first of the notifying agents for v^* (agent g^* sent out from parent of v^* at time s^*) reached vertex v on the path to the root by the start of time step s, and then continued its walk on the path to the root. The second of the corresponding notifying agents, g^{**}, is exactly one step further from the root. Suppose that $g \in \mathcal{A}_v^{(s)} \neq \emptyset$. By the construction of procedure TEL, agent g has been descending along a path from root r to vertex v in consecutive time steps, reaching v at the start of time step s. It follows that g has encountered at some vertex on the path from r to v exactly one of the notifying agents g^*, g^{**} (passing the other on an edge), and so the claim holds. □

Lemma 2.5. *In the local communication model, procedure* TEL *with parameter x explores any rooted tree T in at most $D^* \cdot (1 + \frac{2\log n + \log x/\kappa(x)}{\log(x/(2n\kappa(x)))})$ time steps, for $x > 2n\kappa(x)$ for any positive nondecreasing integer function $\kappa(x)$.*

Proof. As in the proof of Lemma 2.2, we consider any leaf f and the path $\mathcal{F} = (f_0, f_1, \ldots, f_{D_f})$ from r to f. As before, we denote the number of leaves in the subtree of $T^{(i)}$ rooted at f_j by $\lambda_j^{(i)} = L(T^{(i)}, f_j)$. Recall that if f_j is not yet discovered in step i, we have $L(T^{(i)}, f_j) = 1$. We adopt the definition of a wave from Lemma 2.2. We define the values α_i differently, however, to take into account the fact that the procedure relies on a delayed exploration tree, and that some waves lose agents as a result of deploying notifying agents:

$$\alpha_i = x \frac{\lambda_1^{(\lfloor \frac{i}{2} \rfloor)}}{\lambda_0^{(\lfloor \frac{i}{2} \rfloor)}} \frac{\lambda_2^{(\lfloor \frac{i}{2} \rfloor + 1)}}{\lambda_1^{(\lfloor \frac{i}{2} \rfloor + 1)}} \cdots \frac{\lambda_{D_f}^{(\lfloor \frac{i}{2} \rfloor + D_f - 1)}}{\lambda_{D_f-1}^{(\lfloor \frac{i}{2} \rfloor + D_f - 1)}}.$$

We call a wave that discovered at least $\kappa(x)$ new nodes (or equivalently, a wave whose agents were the first to visit at least $\kappa(x)$ nodes of the tree) a *discovery wave*. Thus, there are at most $\lfloor \frac{D_f}{\kappa(x)} \rfloor$ discovery waves, each of which explores at least one node of the considered path. Observe that if a wave is not a discovery wave, then the number of notifying agents it sends out is at most $2\kappa(x) - 2$.

We define by α_i^* the number of agents of the i-th wave that reach leaf f. We now prove that the following analogue of Claim (*) from the proof of Lemma 2.2 holds for non-discovery waves.

Claim (**). Let i be a time step for which $\alpha_i \geq 1$ and w_i is not a discovery wave then, $\alpha_i^* > \alpha_i - 2\kappa(x)$, and thus $\alpha_i - 2\kappa(x)$ is a lower bound on the number of agents reaching f in step $i + D_f - 1$.

Proof (of claim). We define $c_j = \lambda_{j+1}^{(\lfloor \frac{i}{2} \rfloor + j)} / \lambda_j^{(\lfloor \frac{i}{2} \rfloor + j)}$ for $j = 0, \ldots, D_f - 1$. Then

$$\alpha_i = x \prod_{j=0}^{D_f - 1} c_j.$$

Since $c_j \leq 1$ for all j and since $\alpha_i \geq 1$, there exist at most $\log x$ different j such that $c_j \leq 1/2$. Denote the set of all such j by $\mathcal{J}$, with $|\mathcal{J}| \leq \log x$. Denote by $\mathcal{Q}$ the set of all such indices s that wave w_i sends two notifying agents from vertex f_s. By the assumption of the claim, we have that w_i is not a discovery wave thus $|\mathcal{Q}| \leq \kappa(x) - 1$. Also, denote the size (number of agents) of wave w_i in step $i + s$ by a_s $(s = 0, 1, 2, \ldots)$, where $a_0 = x$. Finally, let $\mathcal{R}$ be the set of indices s such that $a_s \geq 2$ and $a_{s+1} = 0$; note that $\mathcal{R}$ has at most one element.

Consider an index $s \notin \mathcal{R}$ for which $c_s > 1/2$ and assume that wave w_i does not send notifying agents from vertex f_s (i.e. $s \notin \mathcal{Q}$). We have $\lambda_{s+1}^{(i+s)} / \lambda_s^{(i+s)} > 1/2$, thus more than half of all leafs of the tree $T^{(i+s)}(f_s)$ also belong to the tree $T^{(i+s)}(f_{s+1})$. But then, in time step $i + s + 1$, agents are sent from f_s to f_{s+1} according to the definition in expression (ii) in the pseudocode of procedure `TEL`. Thus, we can lower-bound the size of wave w_i in step $i + s + 1$ as: $a_{s+1} \geq a_s c_s$. Otherwise, if $s \notin \mathcal{R} \cup \mathcal{Q}$ and $c_s \leq 1/2$ (i.e., if $s \in \mathcal{J}$), then agents are sent according the definition in expression (i) in the pseudocode, and then $a_{s+1} \geq \lfloor a_s c_s \rfloor$. Finally, if $s \in \mathcal{Q}$ then in vertex f_s wave w_i reduces

by 2 notifying agents, while if $s \in \mathcal{R}$ then the wave may be reduced by one more agent ($a_{s+1} = 0$ instead of $a_{s+1} = 1$, since agents are always deployed in groups of two or more), and after that we can perform a similar analysis. Eventually, depending on which of the sets $\mathcal{J}, \mathcal{Q}, \mathcal{R}$ node s belongs to, we obtain:

$$a_{s+1} \geq a_s c_s - \delta_s, \quad \text{where } \delta_s = \delta_s^{(j)} + \delta_s^{(q)} + \delta_s^{(r)},$$

and

$$\delta_s^{(j)} = \begin{cases} 0, & \text{if } s \notin \mathcal{J} \\ 1, & \text{if } s \in \mathcal{J} \end{cases}, \quad \delta_s^{(q)} = \begin{cases} 0, & \text{if } s \notin \mathcal{Q} \\ 2, & \text{if } s \in \mathcal{Q} \end{cases}, \quad \delta_s^{(r)} = \begin{cases} 0, & \text{if } s \notin \mathcal{R} \\ 1, & \text{if } s \in \mathcal{R}. \end{cases}$$

Denote consecutive elements of $\mathcal{J}$, $s_1 < s_2 < s_3 < \cdots < s_{|\mathcal{J}|}$. In this way we expand the expression for $\alpha_i^* = a_{D_f}$:

$$\alpha_i^* = a_{D_f} \geq a_{D_f-1} c_{D_f-1} - \delta_{D_f-1} \geq \ldots \geq (\ldots((a_0 c_0 - \delta_0)c_1 - \delta_1)c_2 - \ldots)c_{D_f-1} - \delta_{D_f-1} =$$

$$= x \prod_{j=0}^{D_f-1} c_j - \sum_{j=0}^{D_f-1} \left(\delta_j \prod_{p=j+1}^{D_f-1} c_j \right) \geq \alpha_i - \sum_{j=0}^{D_f-1} \left((\delta_j^{(j)} + \delta_j^{(q)} + \delta_j^{(r)}) \prod_{p=j+1}^{D_f-1} c_j \right)$$

$$\geq \alpha_i - \sum_{j=0}^{D_f-1} \left(\delta_j^{(j)} \prod_{p=j+1}^{D_f-1} c_j \right) - \sum_{j=0}^{D_f-1} (\delta_j^{(q)} + \delta_j^{(r)}) \geq \alpha_i - \sum_{i=1}^{|\mathcal{J}|} \prod_{p=s_i+1}^{D_f-1} c_p - 2|\mathcal{Q}| - |\mathcal{R}|$$

$$\geq \alpha_i - \sum_{i=1}^{|\mathcal{J}|} 2^{-|\mathcal{J}|+i-1} - 2|\mathcal{Q}| - |\mathcal{R}| > \alpha_i - 1 - 2(\kappa(x) - 1) - 1 = \alpha_i - 2\kappa(x).$$

We have $\alpha_i^* > \alpha_i - 2\kappa(x)$, which completes the proof of the claim.

It is left to prove that if the number of waves a in the execution of the procedure is sufficiently large, there exists an index $i \leq a$, such that wave w_i is not a discovery wave and $\alpha_i \geq 2\kappa(x)$. Note that we can set $\kappa(x)$ to an arbitrary nondecreasing function (even a constant). We again consider the product

$$\prod_{i=1}^{a} \alpha_i = x^a \prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \frac{\lambda_{j+1}^{(\lfloor \frac{i}{2} \rfloor + j)}}{\lambda_j^{(\lfloor \frac{i}{2} \rfloor + j)}} =$$

$$= x^a \frac{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_{j+1}^{(\lfloor \frac{i}{2} \rfloor + j)}}{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_j^{(\lfloor \frac{i}{2} \rfloor + j)}} = x^a \frac{\prod_{i'=-1}^{a-2} \prod_{j'=1}^{D_f} \lambda_{j'}^{(\lfloor \frac{i'}{2} \rfloor + j')}}{\prod_{i=1}^{a} \prod_{j=0}^{D_f-1} \lambda_j^{(\lfloor \frac{i}{2} \rfloor + j)}} =$$

$$= x^a \frac{\left(\prod_{j'=1}^{D_f} \lambda_{j'}^{(j'-1)}\right) \left(\prod_{j'=1}^{D_f} \lambda_{j'}^{(j')}\right) \left(\prod_{i'=1}^{a-2} \prod_{j'=1}^{D_f-1} \lambda_{j'}^{(i'+j')}\right) \left(\prod_{i'=1}^{a-2} \lambda_{D_f}^{(\lfloor \frac{i'}{2} \rfloor + D_f)}\right)}{\left(\prod_{i=1}^{a} \lambda_0^{(\lfloor \frac{i'}{2} \rfloor)}\right) \left(\prod_{i=1}^{a-2} \prod_{j=1}^{D_f-1} \lambda_j^{(i+j)}\right) \left(\prod_{j=1}^{D_f-1} \lambda_j^{(\lfloor \frac{a-1}{2} \rfloor + j)}\right) \left(\prod_{j=1}^{D_f-1} \lambda_j^{(\lfloor \frac{a}{2} \rfloor + j)}\right)} \geq$$

$$\geq x^a \frac{1^{(D_f-1)\cdot(a-1)}}{n^{a+2D_f-2}} \geq \frac{x^a}{n^{a+2D^*}}. \tag{2.2}$$

We now choose a so as to guarantee that there exists at least one non-discovery wave $\alpha_i \geq 2\kappa(x)$. Since there are at most $\lfloor \frac{D^*}{\kappa(x)} \rfloor$ discovery waves, we require that the $\left(\lfloor \frac{D^*}{\kappa(x)} \rfloor + 1\right)$-st biggest value α_i is at least $2\kappa(x)$. Observe that since we have $\alpha_i \leq x$, it suffices to choose a so that:

$$\prod_{i=1}^{a} \alpha_i \geq x^{\lfloor \frac{D^*}{\kappa(x)} \rfloor} (2\kappa(x))^a.$$

Taking into account (2.2), it is sufficient to find a satisfying

$$\frac{x^a}{n^{a+2D}} \geq x^{\frac{D^*}{\kappa(x)}} (2\kappa(x))^a,$$

which holds for sufficiently large x (we assume that $x > 2n\kappa(x)$) for $a = \lceil \frac{2D^* \log n + D^* \log x/\kappa(x)}{\log(x/(2n\kappa(x))} \rceil$. Now, we have that there exists some index $i \leq a$ such that $\alpha_i \geq 2\kappa(x)$ and wave w_i is not a discovery wave. For the same i we have $\alpha_i^* > \alpha_i - 2\kappa(x) \geq 0$, by Claim (**), which implies that $\alpha_i^* \geq 1$, since α_i^* is an integer. Thus, a waves are sufficient to explore the path $\mathcal{F}$. This analysis can be done for any leaf f, thus it is enough to send a waves in order to explore the graph G. We obtain that exploration takes at most $D^* + a - 1 \leq D^* \cdot (1 + \frac{2\log n + \log x/\kappa(x)}{\log(x/(2n\kappa(x))})$ time steps. □

By setting $x = n^c$ and $\kappa(x) = \lceil \log x \rceil$ in Lemma 2.5 we obtain a strategy for online exploration of trees in the model with local communication.

Theorem 2.6. *For any fixed $c > 1$, the online tree exploration problem can be solved in the model with local communication and knowledge of n using a team of $k \geq D^* n^c$ agents in at most $D^* \left(1 + \frac{2}{c-1} + o(1)\right)$ time steps.*

2.3 General graph exploration

In this section we develop strategies for exploration of general graphs, both with global communication and with local communication. These algorithms are obtained by modifying the tree-exploration procedures given in the previous section.

Given a graph $G = (V, E)$ with root vertex r, we call $P = (v_0, v_1, v_2, \ldots, v_m)$ with $r = v_0$, $v_i \in V$, and $\{v_i, v_{i+1}\} \in E$ a *walk* of length $\ell(P) = m$. Note that a walk may contain a vertex more than once. We introduce the notation $P[j]$ to denote v_j, i.e., the j-th vertex of P after the root, and $P[0, j]$ to denote the walk $(v_0, v_1, \ldots, v_j)$, for $j \leq m$. The last vertex of path P is denoted by $end(P) = P[\ell(P)]$. The concatenation of a vertex u to path P, where $u \in \Gamma(end(P))$ is defined as the path $P' \equiv P + u$ of length $\ell(P) + 1$ with $P'[0, \ell(P)] = P$ and $end(P') = u$.

Let $\mathcal{P}$ be the set of walks P in G having length $0 \leq \ell(P) < n$. We introduce a linear order on walks in $\mathcal{P}$ such that for two walks P_1 and P_2, we say that $P_1 < P_2$ if $\ell(P_1) < \ell(P_2)$, or $\ell(P_1) = \ell(P_2)$ and there exists an index $j < \ell(P_1)$ such that

$P_1([0,j]) = P_2([0,j])$ and $P_1([j+1]) < P_2([j+1])$. The comparison of vertices from V is understood as comparison of their identifiers in G.

We now define the tree T with vertex set $\mathcal{P}$, root $(r) \in \mathcal{P}$, such that vertex P' is a child of vertex P if and only if $P' = P + u$, for some $u \in \Gamma(end(P))$. We first show that agents can simulate the exploration of T while in fact moving around graph G. Intuitively, while an agent is following a path from the root to the leaves of T, its location in T corresponds to the walk taken by this agent in G.

Lemma 2.7. *A team of agents can simulate the virtual exploration of tree T starting from root (r), while physically moving around graph G starting from vertex r. The simulation satisfies the following conditions:*

(1) *An agent virtually occupying a vertex P of T is physically located at a vertex $end(P)$ in G.*

(2) *Upon entering a vertex P of T in the virtual exploration, the agent obtains the identifiers of all children of P in T.*

(3) *A virtual move along an edge of T can be performed in a single time step, by moving the agent to an adjacent location in G.*

(4) *Agents occupying the same virtual location P in T can communicate locally, i.e., they are physically located at the same vertex of G.*

Proof. We define the simulation so that claims (1-4) hold for all time steps. Initially, claim (1) is trivially true since $end((r)) = r$. Suppose that at the start of some step s, an agent occupies some virtual location P in T, and its corresponding physical location is $end(P)$. Claim (2) holds for this step, since the set of children of P in T is given as $\{P + u \in \mathcal{P} : u \in \Gamma(end(P))\}$, P is stored in the agents memory (as the identifier of its location in T), and the neighborhood of $end(P)$ in G is accessible to the agent by definition. When required to move to a virtual location P' adjacent to P in T, the agent performs a move to vertex $end(P') \in V$. Note that if P' is the child of P in T, then $end(P') \in \Gamma(end(P))$ by definition of T, whereas if P' is the parent of P in T, then $end(P') = P[\ell(P) - 1] \in \Gamma(end(P))$ from the definition of walk P. After such a move, claim (1) is immediately satisfied, and claims (2-3) follow by induction on time. Claim (4) is a trivial consequence of claim (1). □

We remark that the number of vertices of tree T is exponential in n. Hence, our goal is to perform the simulation with only a subset of the vertices of T. For a vertex $v \in V$, let $P_{\min}(v) \in \mathcal{P}$ be the minimum (with respect to the linear order on $\mathcal{P}$) walk ending at v. We observe that, by property (1) in Lemma 2.7, if, for all $v \in V$, the vertex $P_{\min}(v)$ of T has been visited by at least one agent in the virtual exploration of T, the physical exploration of G is completed. We define $\mathcal{P}_{\min} = \{P_{\min}(v) : v \in V\}$, and show that all vertices of $\mathcal{P}_{\min}$ are visited relatively quickly if we employ the procedure `TEG` (or `TEL`) for T, subject to a simple modification. In the original algorithm, we divided the

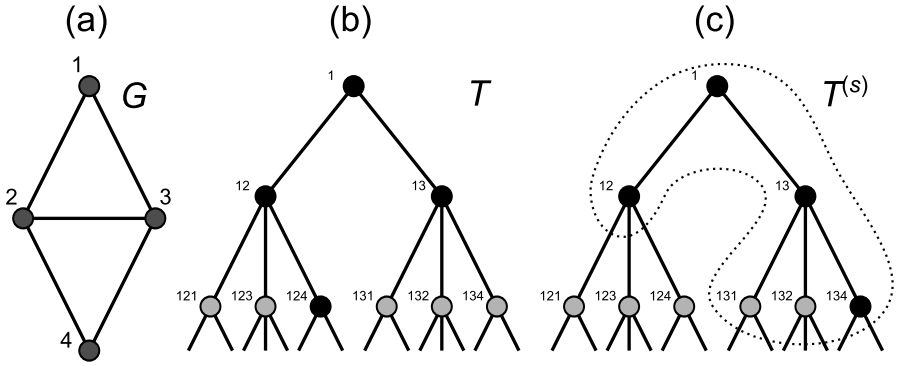

FIGURE 2.2: Illustration of exploration of general graphs: (a) The explored graph G, (b) The virtually explored tree of walks T, with highlighted nodes belonging to $\mathcal{P}_{\min}$, (c) An example of a subtree $T^{(s)} \subseteq T$ with highlighted nodes which are counted when computing function L (this tree $T^{(s)}$ does not correspond to a real execution of procedure TEL on T).

agents descending to the children of the vertex according to the number of leaves of the discovered subtrees. We introduce an alternate definition of the function $L(T^{(s)}, v)$, so as to take into account only the number of vertices in $T^{(s)}$ corresponding to walks which are smallest among all walks in $T^{(s)}$ sharing the same end-vertex.

Lemma 2.8. *Let $T^{(s)} \subseteq T$ be a subtree of T rooted at (r). For $P \in V(T^{(s)})$, let $L(T^{(s)}, P)$ be the number of vertices v of G, for which the subtree of $T^{(s)}$ rooted at P contains a vertex representing the smallest among all walks contained in $T^{(s)}$ which end at v:*

$$L(T^{(s)}, P) = \Big| V(T^{(s)}(P)) \cap \bigcup_{v \in V} \Big\{ \min\{P' \in V(T^{(s)}) : end(P') = v\} \Big\} \Big|,$$

and for $P \in \mathcal{P} \setminus V(T^{(s)})$, let $L(T^{(s)}, P) = 1$. Subject to this definition of L, procedure TEG *with parameter $x > n$ (procedure* TEL *with parameter $x > 2n\kappa(x)$ for any nondecreasing function $\kappa(x)$) applied to tree T starting from root (r) visits all vertices from $\mathcal{P}_{\min}$ within $D^* \cdot (1 + \frac{\log n}{\log(x/n)})$ (respectively, $D^* \cdot (1 + \frac{2\log n + \log x/\kappa(x)}{\log(x/(2n\kappa(x))})$) time steps.*

Proof. The set $\mathcal{P}_{\min}$ spans a subtree $T_{\min} = T[\mathcal{P}_{\min}]$ in T, rooted at (r). We can perform an analysis analogous to that used in the Proofs of Lemmas 2.2 and 2.5, evaluating sizes of waves of agents along paths in the subtree $T_{\min}$. We observe that for any $P \in \mathcal{P}_{\min}$ which is not a leaf in $T_{\min}$, we always have $L(T^{(s)}, P) \geq 1$. Moreover, we have $L(T^{(s)}, P) \leq |V(T^{(s)}(P))|$, and so $L(T^{(s)}, P) \leq n$. Since these two bounds were the only required properties of the functions L in the Proofs of Lemmas 2.2 and 2.5, the analysis from these proofs applies within the tree $T_{\min}$ without any changes. It follows that each vertex of $\mathcal{P}_{\min}$ is reached by the exploration algorithm within $D^* \cdot (1 + \frac{\log n}{\log(x/n)})$ time steps in case of global communication, and within $D^* \cdot (1 + \frac{2\log n + \log x/\kappa(x)}{\log(x/(2n\kappa(x))})$ time steps in case of local communication. □

We recall that by Lemma 2.7, one step of exploration of tree T can be simulated by a single step of an agent running on graph G. Thus, appropriately choosing $x = \Theta(n^c)$ and $\kappa(x) = \lceil \log x \rceil$ in Lemma 2.8, we obtain our main theorem for general graphs.

Theorem 2.9. *For any $c > 1$, the online graph exploration problem with knowledge of n can be solved using a team of $k \geq D^* n^c$ agents:*

- *in at most $D^* \cdot \left(1 + \frac{1}{c-1} + o(1)\right)$ time steps in the global communication model.*
- *in at most $D^* \cdot \left(1 + \frac{2}{c-1} + o(1)\right)$ time steps in the local communication model.*

For the case when we do not assume knowledge of (an upper bound on) n, we provide a variant of the above theorem which also completes exploration in $O(D)$ steps, with a slightly larger multiplicative constant.

Theorem 2.10. *For any $c > 1$, there exists an algorithm for the local communication model, which explores a rooted graph of unknown order n and unknown diameter D using a team of k agents, such that its exploration time is $O(D)$ if $k \geq Dn^c$.*

Proof. Let $c' = \frac{c+1}{2}$, $1 < c' < c$. For a graph G, the algorithm proceeds by assuming geometrically increasing upper bounds $\bar{D} = 1, 2, 4, \ldots$, on the value of D^*. For a fixed value of $\bar{D}$, we set $\bar{n} = \lfloor (k/\bar{D})^{1/c'} \rfloor$, and perform exploration of the graph using the algorithm from Theorem 2.9 with parameter c', assuming that the explored graph has at most $\bar{n}$ vertices, and using $\bar{D}\bar{n}^{c'} \leq k$ agents. After at most $\bar{D} \cdot \left(1 + \frac{2}{c'-1} + o(1)\right)$ time steps (where the asymptotic $o(1)$ value follows from Theorem 2.9) exploration is interrupted, and all agents return to the root vertex in at most $O(\bar{D})$ steps. If exploration of G has been completed, then the algorithm stops. This can be detected since the agents are aware which vertices still have unexplored neighbors. If the exploration has not been completed, we continue for a doubled value of $\bar{D}$, until the bound $\bar{n} = 0$ is reached. Finally, if exploration has been unsuccessful so far, we perform an arbitrary valid exploration algorithm, e.g. Depth First Search (DFS) with a single agent.

The algorithm always completes exploration successfully in finite time. Observe that if in the stage with $\bar{D} = 2^{\lceil \log_2 D^* \rceil}$ and $\bar{n} = \lfloor (k/\bar{D})^{1/c'} \rfloor$ we have $\bar{n} \geq n$, then exploration is completed successfully in this stage, and the total time of all exploration stages is $O(D^*)$. Observe that we have $\bar{D} < 2D^* \leq 2D$ and $k \geq Dn^c$, and so it suffices that $\lfloor (n^c/2)^{1/c'} \rfloor \geq n$. This holds for sufficiently large n. If the condition $k \geq Dn^c$ does not hold or n is too small, then the algorithm reaches the final phase in which DFS is executed, resulting in a correct exploration of the graph in finite time. □

We remark that by choosing $\kappa(x) = 1$ and $x = (2 + \epsilon)n$ for any constant $\epsilon > 0$ in Lemma 2.4, we can also explore a graph using $k = (2 + \epsilon)nD^*$ agents in time $\Theta(D \log n)$, with local communication. This bound is the limit of our approach in terms of the smallest allowed team of agents.

2.4 Lower bounds

In this section, we show lower bounds for exploration with D^*n^c agents, complementary to the positive results given by Theorem 2.9. The graphs that produce the lower bound are a special class of trees. The same class of trees appeared in the lower bound from [85] for the competitive ratio of tree exploration algorithms with small teams of agents. In our scenario, we obtain different lower bounds depending on whether communication is local or global.

Theorem 2.11. *For all $n > 1$ and for every increasing function f, such that $\log f(n) = o(\log n)$, and every constant $c > 0$, there exists a family of trees $\mathcal{T}_{n,D^*}$, each with n vertices and height $D^* = \Theta(f(n))$, such that*

(i) for every exploration strategy with global communication that uses D^*n^c agents there exists a tree in $\mathcal{T}_{n,D^*}$ such that the number of time steps required for its exploration is at least $D^*\left(1 + \frac{1}{c} - o(1)\right)$,

(ii) for every exploration strategy with local communication that uses D^*n^c agents there exists a tree in $\mathcal{T}_{n,D^*}$ such that the number of time steps required for exploration is at least $D^*\left(1 + \frac{2}{c} - o(1)\right)$.

Proof. We prove the theorem assuming that the number of agents is n^c, rather than D^*n^c. The asymptotic form of the bounds in claims (i) and (ii) remains unchanged since $D^* = n^{o(1)}$ by assumption, and $D^*n^c = n^{c+o(1)}$. In previous sections we assumed that a certain number of agents spawned in the root r every round. Here we assume that all n^c agents are available in the first round.

(i) First we define the family of trees $\mathcal{T}_{n,D^*}$. It is possible to find $D^* = \Theta(f(n))$ such that for any n there exist integers Δ and κ such that $n = D^*\Delta + \kappa + 1$ and $0 \leq \kappa \leq D^* - 1$. Note that $\Delta = \frac{n-(\kappa+1)}{D^*}$. Given a vector $\boldsymbol{q} = (q_1, \ldots, q_{D^*}) \in \{1, \ldots, \Delta\}^{D^*}$, we define $T(\boldsymbol{q})$ as the tree rooted at r with vertex set

$$V(T(\boldsymbol{q})) = \{r\} \cup \bigcup_{i=1}^{D^*} V_i \cup W,$$

where $V_i = \{v_1^i, \ldots, v_\Delta^i\}$ is the set of nodes at distance i from the root r and $W = \{w_1, w_2, \ldots, w_\kappa\}$ is the set of additional nodes attached to the root. For convenience, we set $v_{q_0}^0 = r$, and we define the edge set by

$$E(T(\boldsymbol{q})) = \bigcup_{i=1}^{D^*} \left\{ \{v_{q_{i-1}}^{i-1}, v_j^i\} \;\middle|\; j = 1, \ldots, \Delta \right\} \cup \left\{ \{r, w_j\} \;\middle|\; j = 1, 2, \ldots \kappa \right\},$$

which means that one specific vertex $v_{q_{i-1}}^{i-1}$ from level $i-1$ is connected to all vertices on level i. We set $\mathcal{T}_{n,D^*} = \{T(\boldsymbol{q}) \mid q \in \{1, \ldots, \Delta\}^{D^*}\}$. Since we are interested in lower bounds we will not consider vertices from W, we assume that exploration is finished when all vertices from sets V_i are explored.

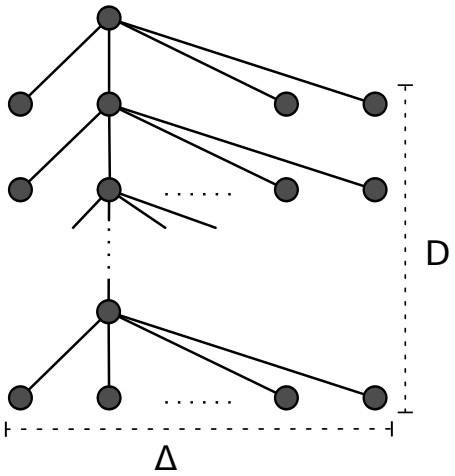

FIGURE 2.3: An example of a tree in $\mathcal{T}_{n,D^*}$.

We prove that each exploration strategy that uses at most n^c agents needs at least $D^*(1 + \frac{1}{c} - o(1))$ steps to explore some tree in $\mathcal{T}_{n,D^*}$.

Let $\mathcal{S}$ be any exploration strategy that uses at most n^c agents. We select a tree from $\mathcal{T}_{n,D^*}$ based on the behavior of $\mathcal{S}$ in the class of trees $\mathcal{T}_{n,D^*}$. More precisely, let $T \in \mathcal{T}_{n,D^*}$ be such that, for each $i = 1, \ldots, D^* - 1$, if s is the first step in which a vertex in V_i is visited, then one of the vertices in V_i holding the minimum number of agents in step s is the one having the vertices in V_{i+1} as children. In the following we bound the number of steps of $\mathcal{S}$ while exploring T. We say that $\mathcal{S}$ *makes progress* in time step s if for some $i \in \{1, \ldots, D^*\}$, some vertex in V_i is not visited in step $s-1$ and all vertices in V_i are visited at the start of step s. If only a strict non-empty subset of vertices of V_i are visited in some step s, then, by the choice of T, the vertex in V_i that has Δ children is among those not visited in step s. We have n^c agents in $v^0_{q_0}$ in step 1. In step 2 at most n^c/Δ agents reach $v^1_{q_1}$. In step 3 at most n^c/Δ^2 agents reach $v^2_{q_2}$, and so on. Thus, $\mathcal{S}$ exploring tree T can make progress in at most $\lfloor \log_\Delta n^c \rfloor$ consecutive time steps. This is due to the choice of T.

Let p be the number of maximal sequences of consecutive time steps in which $\mathcal{S}$ makes progress. Let s_i, $i = 1, \ldots, p$, be the length of the i-th such sequence. By the above, we obtain that $s_i \leq \lfloor \log_\Delta(n^c) \rfloor$ for each $i = 1, \ldots, p$. Since the strategy $\mathcal{S}$ explores the entire tree T, the total number of steps in which $\mathcal{S}$ makes progress equals D^*, the height of the tree T. We obtain $D^* = \sum_{i=1}^{p} |s_i| \leq p \lfloor \log_\Delta n^c \rfloor$. Thus we can lower bound the value p

$$p \geq \frac{D^*}{\lfloor c \log_\Delta n \rfloor} \geq D^* \frac{\log \Delta}{c \log n} = D^* \frac{\log(n - (\kappa + 1)) - \log D^*}{c \log n} = D^* \left(\frac{1}{c} - o(1) \right), \quad (2.3)$$

because $\log D^* = \Theta(\log f) = o(\log n)$ and $\log(n - (\kappa + 1))/\log n = 1 - o(1)$. Each pair of maximal sequences of consecutive time steps in which $\mathcal{S}$ makes progress has to be separated by at least one step in which $\mathcal{S}$ makes no progress in tree T. Thus there are at least $p - 1$ steps without progress and at most D^* steps with progress. Let s' be the first

step in which all vertices are visited when executing $\mathcal{S}$ in T. By (2.3) and by the choice of T,

$$s' \geq p-1+D^* \geq D^*\left(\frac{1}{c}-o(1)\right)-1+D^* = D^*\left(1+\frac{1}{c}-\frac{1}{D^*}-o(1)\right) \geq D^*\left(1+\frac{1}{c}-o(1)\right),$$

where $\frac{1}{D^*} = o(1)$ because f is increasing. This completes the proof of (i).

(ii) We use the same family of trees $\mathcal{T}$ as in (i). Let $\mathcal{S}$ be any exploration strategy with local communication that uses at most n^c agents.

We select a tree from $\mathcal{T}_{n,D^*}$ based on the behavior of $\mathcal{S}$ in the class of trees $\mathcal{T}_{n,D^*}$. If step s is the first step in which a vertex in V_i is visited, then, since communication is local, agents located in vertex $v^{i-1}_{q_{i-1}}$ have no knowledge about the degrees of the vertices in V_i in step s. Since no agent comes back from V_i in step s, agents in $v^{i-1}_{q_{i-1}}$ have the same knowledge in steps $s-1$ and s. We select $T \in \mathcal{T}$ in such a way that a vertex in V_i for which the sum of the number of agents in steps s and $s+1$ is minimized, is the vertex $v^i_{q_i}$. Now, similarly as in (i), we lowerbound the number of steps.

We have n^c agents in $v^0_{q_0}$ in step 1. Together, in steps 2 and 3, in total at most n^c/Δ agents reach $v^1_{q_1}$. In steps 3 and 4 at most n^c/Δ^2 agents reach $v^2_{q_2}$, and so on. Thus in the first $\lfloor \log_\Delta n^c \rfloor + 2$ time steps, there are two steps in which the algorithm does not make progress in terms of levels explored. Similarly as in previous part of the theorem the number of time steps without progress can be lowerbounded by $D^*\left(\frac{2}{c}-o(1)\right)$. Thus the exploration takes at least $D^*\left(1+\frac{2}{c}-o(1)\right)$ steps. □

2.5 Conclusions

In this chapter we studied the collaborative online graph exploration problem by agents endowed with communication capabilities. We showed that even a short-range communication (local communication) allows for a polynomial team of agents to explore any graph in the asymptotically shortest possible time $O(D)$. We also studied the tradeoff between the optimal exploration time and the size of the team of agents. Our analysis showed that for the case of known n, the optimal exploration time converges to D^* and the convergence rate corresponds to $\log_n k$.

When looking at the problem of minimizing the size of the team of agents, our work (Theorem 2.10) shows that it is possible to achieve asymptotically-optimal online exploration time of $O(D)$ using a team of $k \leq Dn^{1+\epsilon}$ agents, for any $\epsilon > 0$. For graphs of small diameter, $D = n^{o(1)}$, we can thus explore the graph in $O(D)$ time steps using $k \leq n^{1+\epsilon}$ agents. This result almost matches the lower bound on team size of $k = \Omega(n^{1-o(1)})$ for the case of graphs of small diameter, which follows from the trivial lower bound $\Omega(D + n/k)$ on exploration time (cf. Proposition 2.1). The question of establishing precisely what team size k^* is necessary and sufficient for performing exploration in $O(D)$ steps in a graph of larger diameter remains open. The lower bound $\Omega(D + n/k)$ shows that $k^* = \Omega(n/D)$ and our algorithm shows that $k^* = O(Dn^{1+\epsilon})$. A

similar open problem is the problem of finding the smallest number of agents k' for which it is possible to design an algorithm with constant competitive ratio. The lower bound from [67] shows that $k' = \Omega(\sqrt{n})$ and our algorithm shows that $k' = O(Dn^{1+\epsilon})$.

Another question would be to generalize the model in some way. If we consider crashes or Byzantine faults in the local communication model will it be still possible to explore graphs (or trees) efficiently? Our analysis of algorithms in this chapter is robust and it should be possible to generalize it to the model with faults. For example if we consider a model where the first agent entering a node disappears then almost the same algorithm works and the proof holds after small modifications. Another possible extension would be to consider dynamic graphs.

We did not consider space complexity of the algorithms presented in this chapter. It would be interesting to study how many bits of internal memory of the agents are necessary to execute the presented algorithms. Decreasing the available memory should result in time-space tradeoffs as agents can keep approximation on the sizes of subtrees (e.g. keep $\lceil \log L \rceil$ instead of L).

Chapter 3

Exploration with the Rotor-Router system

In this chapter, we perform an extensive study of the cover time of collaborative exploration of the multi-agent rotor-router. The rotor-router mechanism was introduced by Priezzhev *et al.* [136] as a deterministic alternative to the random walk in undirected graphs. In this model, a set of k identical agents is deployed in parallel, starting from a chosen subset of nodes, and moving around the graph in synchronous steps. During the process, each node maintains a cyclic ordering of its outgoing arcs, and successively propagates agents which visit it along its outgoing arcs in round-robin fashion, according to the fixed ordering. Thus the rotor-router system can be seen as a model in which agents are interacting with the environment as opposed to Chapter 2 where agents were interacting between each other. We propose new techniques which allow us to perform a theoretical analysis of the multi-agent rotor-router model, and to compare it to the scenario of parallel independent random walks in a graph. Using these techniques we provide tight bounds on the cover time of k agent rotor-router system. We show that this cover time in the worst case initialization is at most $\Theta(mD/\log k)$ and at least $\Theta(mD/k)$ for any graph, which corresponds to a speedup of between $\Theta(\log k)$ and $\Theta(k)$ with respect to the cover time of a single walk. Both of these extremal values of speedup are achieved for some graph classes. We prove that the speedup of $\Theta(\log k)$ is achieved in the worst case initialization of agents for the ring. We show that on the ring, depending on the initial locations of agents admits the cover time of between $\Theta(n^2/k^2)$ in the best case and $\Theta(n^2/\log k)$ in the worst case. The corresponding expected value of cover time for k random walks, depending on the initial placement of the agents, is proven to belong to a similar range, namely between $\Theta(n^2/(k^2/\log^2 k))$ and $\Theta(n^2/\log k)$. Then we develop a relation linking the cover time of the rotor-router to the mixing time of the random walk and the local divergence of a discrete diffusion process on the considered graph. Using this relation we determine the precise asymptotic value of the rotor-router cover time for the worst-case initial placement of agents, for all values of k for degree-restricted expanders, random graphs and constant-dimensional tori. For

hypercubes, we also resolve the question precisely, except for values of k much larger than n. For constant-dimensional tori we observe an interesting phenomenon where linear speedup is achieved for k up to a threshold value k_1. Adding more agents above k_1 gives only logarithmic speedup i.e. for $k > k_1$ the cover time decreases with $\log(k/k_1)$. Finally when k is increased beyond the second threshold, then we observe no speedup and adding more agents no longer decreases the cover time.

This chapter contains results which have appeared in conference publications [T3-T5].

3.1 The rotor-router model

The study of deterministic exploration strategies in agent-based models of computation is largely inspired by considerations of random walk processes. For an undirected graph $G = (V, E)$, exploration with the random walk has many advantageous properties: the expected arrival time of the agent at the last unvisited node of the graph, known as the *cover time* $C(G)$, can in general be bounded as, e.g., $C(G) \in O(D|E| \log |V|)$, where D is the diameter of the graph. The random walk also has the property that in the limit it visits all of the edges of the graph with the same frequency, traversing each edge, on average, once every $|E|$ rounds. In this chapter, we consider a specific deterministic model of walks on graphs, known as the *rotor-router*. The rotor-router model, introduced by Priezzhev *et al.* [136] and further popularised by James Propp, provides a mechanism for the environment to control the movement of the agent deterministically, whilst retaining similar properties of exploration as the random walk.

In the rotor-router model, the agent has no operational memory and the whole routing mechanism is provided within the environment. The edges outgoing from each node v are arranged in a fixed cyclic order known as a *port ordering*, which does not change during the exploration. Each node v maintains a *pointer* which indicates the edge to be traversed by the agent during its next visit to v. If the agent has not visited node v yet, then the pointer points to some initial edge adjacent to v. The next time when the agent enters node v, it is directed along the edge indicated by the pointer, which is then advanced to the next edge in the cyclic order of the edges adjacent to v.

The behavior of the rotor-router for a single agent is well understood. Studies of the rotor-router started with works of Wagner *et al.* [143] who showed that in this model, starting from an arbitrary configuration (arbitrary cyclic orders of edges, arbitrary initial values of the port pointers and an arbitrary starting node) the agent covers all m edges of an n-node graph within $O(nm)$ steps. Bhatt *et al.* [19] showed later that within $O(nm)$ steps the agent not only covers all edges but enters (establishes) an Eulerian cycle. More precisely, after the initial stabilization period of $O(nm)$ steps, the agent keeps repeating the same Eulerian cycle of the directed symmetric version $\vec{G}$ of graph G. Subsequently, Yanovski *et al.* [144] and Bampas *et al.* [15] showed that the Eulerian cycle is in the worst case entered within $\Theta(mD)$ steps in a graph of diameter D. Considerations of specific graph classes were performed in [91]. Robustness properties of the rotor-router

were further studied in [16], who considered the time required for the rotor-router to stabilize to a (new) Eulerian cycle after an edge is added or removed from the graph. Regarding the terminology, we note that the rotor-router model has also been referred to as the *Propp machine* [15] or *Edge Ant Walk algorithm* [143, 144], and has also been described in [19] in terms of traversing a maze and marking edges with pebbles.

Our work deals with the problem of exploring a graph with the *multi-agent rotor-router*, i.e., a rotor-router system in which more than one agent is deployed in the same environment. Due to the interaction of the agents, which move the same set of pointers at nodes, this can be seen as an example of a deterministic interacting particle system. In the first work on the topic, Yanovski *et al.* [144] showed that adding a new agent to a rotor-router system with k agents cannot increase the cover time, and showed experimental evidence suggesting that a speedup does indeed occur. In this chapter we completely resolve the question of the possible range of speedups in the worst-case initial setting of the parallel rotor-router model in a graph, showing that its value is between $\Theta(\log k)$ and $\Theta(k)$, for any graph. Both of these bounds are tight. We also determine the precise asymptotic value of the rotor-router cover time for all values of k for degree-restricted expanders, random graphs, constant-dimensional tori and cycles. For hypercubes, we also resolve the question precisely, except for values of k much larger than n. For cycles we provide the result describing the structure of the process. This allows us to bound the cover time as well as to describe the limit behavior of the system. In this chapter, we also perform a comparative case study of two seemingly different scenarios: deterministic exploration with interacting particles in the rotor-router model vs. randomized exploration with non-interacting particles in the random walk, showing certain similarities between them. We compare our results with the so-called *parallel random walk*, achieved by deploying independent agents performing random walks in a graph independently and without any form of coordination.

Recent work on the area of parallel random was initiated by Alon *et al.* [7] who introduced the notion of the speedup of k independent random walks as the ratio of the cover time of a single walk to the cover time of k random walks. The speedup may sometimes be as low as $\log k$ [7], and sometimes as high as exponential in terms of k [7] depending on graph topology and the initial positions of the agents. Speedup in the worst-case initial placement was shown to be k for many graph classes, such as complete graphs [7], d-dimensional grids [7, 71], hypercubes [7, 71], expanders [7, 71], and different models of random graphs [7, 71]. For the cycle, the worst-case speedup is equal to $\log k$ [7]. For general graphs, an upper bound $\min\{k \log n, k^2\}$ on the worst-case speedup was obtained by Efremenko *et al.* [70]. The $k \log n$ upper bound was shown independently by Elsässer *et al.* [71]. The speedup for parallel random walks for the worst-case initial placement of agents is conjectured to be between $\log k$ and k for any graph [7]. Interestingly, the multi-agent rotor-router, which can be seen as a derandomization of multiple random walks, achieves the range of speedups that corresponds to the conjectured range for multiple random walks.

Another measure studied by Efremenko *et al.* [70] concerns the speedup with respect to a different exploration parameter — the maximizing hitting time, i.e., the maximum over all pairs of nodes of the graph of the expected time required by the walk to move from one node to the other. For this parameter, they show a bound on speedup of $O(k)$, mentioning that it is tight in many graph classes.

In the context of graph exploration, before the work presented in this thesis, the only study of the multi-agent rotor-router was performed by Yanovski *et al.* [144], who showed that adding a new agent to the system cannot slow down exploration, and provided some experimental evidence showing a nearly-linear speedup of cover time with respect to the number of agents in practical scenarios. They also show that the multi-agent rotor-router eventually visits all edges of the graph a similar number of times. Beyond this, a characterization of the behavior of the k-agent rotor-router in general graphs remained an open question.

In Section 3.2 we introduce the techniques used in the analysis of the multi-agent rotor-router system. The basic tool is applicable to general graphs and gives us an algorithmic perspective for analysis of the rotor-router through *delayed deployments*, allowing the occasional stopping of some of the agents without affecting asymptotic cover time.

In Section 3.3, we prove that the cover time of multi-agent rotor router system in any graph is in $O(mD/\log k)$, when $k < 2^{16D}$. We then extend this result to the case of $k \in O(poly(n))$, i.e., $k < n^c$ for some absolute constant c. The main part of our proofs relies on a global analysis of the number of visits to edges in successive time steps, depending on the number of times that these edges have been traversed in the past. We first prove a stronger version of local structural lemmas proposed by Yanovski *et al.* [144], and apply them within a global amortization argument over all time steps and all edges in the graph.

In Section 3.4, we show a complementary lower bound on the cover time of the k-agent rotor-router in worst case initialization, namely, that the cover time is in $\Omega(mD/k)$. As a starting point, the proof uses a decomposition of the edge set of a graph, introduced by Bampas *et al.* [15], into a "heavy part" containing a constant proportion of the edges and a "deep part", having diameter linear in D. The main part of the analysis is to show that an appropriate initialization of k agents in the heavy part takes a long time to reach the most distant nodes of the deep part. The argument also takes advantage of the delayed deployment technique. We close the section by remarking that a cover time of $\Theta(mD/k)$ is, in fact, achieved for some graphs, such as stars.

In Section 3.5, for the specific case of the rotor-router on the ring (cycle), we describe states in the evolution of the system in which particular agents cover nearly disjoint, dynamically changing parts of the graph, known as *agent domains*. We also introduce a *continuous time approximation* of the evolution of the system on the ring, which allows us to postulate an asymptotic description of the behavior of the agents on the ring. Formal proofs of correctness are obtained through an analysis of the motion of agents within

Model	Cover time		Return time
	for worst placement	*for best placement*	
k-agent rotor-router	$\Theta(n^2/\log k)$ Thm 3.15, 3.17	$\Theta(n^2/k^2)$ Thm 3.25, 3.28	$\Theta(n/k)$ Thm 3.33
k random walks (expectations)	$\Theta(n^2/\log k)$ [7]	$\Theta\left(n^2 \Big/ \frac{k^2}{\log^2 k}\right)$ Thm 3.32	$\Theta(n/k)$ e.g. [5]

TABLE 3.1: The cover time of the multi-agent roter-router on the ring compared to multiple random walks, depending on the initial placement of the agents.

their domains in delayed deployments of the rotor-router. We show that for a k-agent rotor-router system in a graph G with n nodes, m edges and diameter D, the cover time is between $\Theta(n^2/k^2)$ and $\Theta(n^2/\log k)$, depending on the initial placement of the agents in the rotor-router. The first bound is achieved, in particular, for agents distributed uniformly on the ring, while the latter for agents initially located on the same node of the ring. The return times for the ring of the k-agent rotor-router is determined in Section 3.6 as $\Theta(n/k)$. Summary of results from Sections 3.5 and 3.6 is organized in Table 3.1. We remark that for a single agent, the rotor-router on the ring deterministically achieves a cover time of $\Theta(n^2)$, which matches that of the random walk. As the number of agents k increases, the speedup of the rotor-router with respect to a single-agent system is seen from our results as between $\Theta(\log k)$ and $\Theta(k^2)$, depending on the initialization. These results are comparable with the corresponding speedup of the random walk, which is between $\Theta(\log k)$ and $\Theta(k^2/\log^2 k)$. The speedup in terms of return time is $\Theta(k)$, in both cases. By showing that the cover time for the ring in the worst case is $\Theta(n^2/\log k)$ we prove that the lower bound $O(\log k)$ on speedup (Section 3.3) cannot be improved.

In Section 3.7 we outline the technique which we subsequently use to bound the cover time in different graph classes. The main theorem of Section 3.7 captures the link between the cover time of the k-rotor-router system, the mixing time $\mathsf{MIX}_{1/4}$ of the random walk process in the graph, and a graph parameter known as its discrepancy Ψ [18, 138], in its simplest form.

We recall that for $k = 1$, the worst-case cover time of the rotor-router is $\Theta(mD)$, and note that for sufficiently large k ($k > n\Delta^D$ for a graph of maximum degree Δ), the cover time of the rotor-router is equal to precisely D, since the graph can be flooded with agents starting from a fixed node initially having Δ^D agents. Above this threshold ($k > n\Delta^D$), adding new agents to the system does not speed up exploration. The results presented in Section 3.8 show that for complete graphs, random graphs, and expanders, a cover time of $\Theta(D)$ is attained already for much smaller teams of agents. These graphs also display dichotomous behaviour: up to a certain threshold value of $k_1 = \Theta(m)$, the cover time decreases linearly with the number of agents, and above this threshold, the cover time remains fixed at $\Theta(D)$.

Graph	k	*Cover time*	*Reference*
General graph	$\leq poly(n)$	$O\left(\frac{mD}{\log k}\right)$	Thm. 3.6, 3.10
		$\Omega\left(\frac{mD}{k}\right)$	Thm. 3.11
Cycle	$< 2^n$	$\Theta\left(\frac{n^2}{\log k}\right)$	Thm. 3.15 3.39
	$\geq 2^n$	$\Theta(n)$	
d-dim. torus	$< n^{1-1/d}$	$\Theta\left(\frac{n^{1+1/d}}{k}\right)$	Thm. 3.41
	$\in [n^{1-1/d}, 2^{n^{1/d}}]$	$\Theta\left(\frac{n^{2/d}}{\log(k/n^{1-1/d})}\right)$	Thm. 3.41
	$> 2^{n^{1/d}}$	$\Theta(n^{1/d})$	
Hypercube	$< n\frac{\log n}{\log\log n}$	$\Theta\left(\frac{n\log^2 n}{k}\right)$	Cor. 3.42
	$\in \left[n\frac{\log n}{\log\log n}, n2^{\log^{1-\varepsilon} n}\right]$ (for any $\varepsilon > 0$)	$\Theta(\log n \log\log n)$	Thm. 3.43
	$> n2^{\log^{1-\varepsilon} n}$	$O(\log n \log\log n)$	Thm. 3.43
	$> (\log_2 n)^{\log_2 n}$	$\Theta(\log n)$	
Complete	$< n^2$	$\Theta\left(\frac{n^2}{k}\right)$	Thm. 3.38
	$\geq n^2$	$\Theta(1)$	
Expander	$< n\log n$	$\Theta\left(\frac{n\log^2 n}{k}\right)$	Thm. 3.38
	$\geq n\log n$	$\Theta(\log n)$	
Random graph	$< n\log n$	$\Theta\left(\frac{n\log^2 n}{k}\right)$	Thm. 3.38
	$\geq n\log n$	$\Theta(\log n)$	

TABLE 3.2: Worst-case cover time of the k-agent rotor-router system for different values of k in a n-node graph with m edges and diameter D. The results for d-dimensional tori are presented for d constant. The result for expanders concerns the case when the ratio of the maximum degree and the minimum degree of the graph is $O(1)$. The result for random graphs holds in the Erdős-Renyi model with edge probability $p > (1+\varepsilon)\frac{\log n}{n}$, $\varepsilon > 0$, a.s.

In Section 3.9 we extend the results for the ring from Section 3.5. We show that the logarithmic speedup is attained for all $k < 2^n$. Thus the cover time of the ring is $\Theta(n^2/\log k)$ for $k < 2^n$, and of $\Theta(n)$ for $k \geq 2^n$.

In Section 3.10, we prove that the d-dimensional torus for constant d (with $D = n^{1/d}$) admits precisely two threshold values of k (cf. Table 3.2). For $k < k_1 = n^{1-1/d}$, the speedup is linear with k; for $k_1 \leq k < k_2 = 2^{n^{1/d}}$, the cover time further decreases with $\log(k/k_1)$, and above k_2, the cover time is asymptotically fixed at $\Theta(n^{1/d})$. We remark that the for parallel random walks, the situation appears to be similar, however the question of obtaining a complete characterization remains open.

Finally in Section 3.11 we also prove threshold behaviour for the speedup of the k-agent rotor-router for the hypercube, showing that there exist at least three threshold values of k (linear speedup for small k, a flat period with no speedup for k slightly larger than n, a further period of slow growth, and finally a flat period for extremely large

k). We also completely characterize the cover time of the hypercube for k up to a point beyond the first threshold.

Table 3.2 contains a summary of all our results on the cover time of the k-agent rotor-router.

3.1.1 Definitions and notation

Let $G = (V, E)$ be an undirected connected graph with n nodes, m edges and diameter D. We denote the neighborhood of a node $v \in V$ by $\Gamma(v)$. The directed graph $\vec{G} = (V, \vec{E})$ is the directed symmetric version of G, where the set of arcs $\vec{E} = \{(v, u), (u, v) : \{v, u\} \in E\}$. We will refer to the undirected links in graph G as *edges* and to the directed links in graph $\vec{G}$ as *arcs*. We will denote arc (v, u) by $v \to u$. We will also keep using an arrow on the top of a symbol, as in $\vec{G}$ and $\vec{E}$, to stress that we refer to directed graphs and arcs. For a node $v \in V$, $d(v)$ denotes the degree of v in G.

We consider the rotor-router model (on graph G) with $k \geq 1$ indistinguishable agents, which run in rounds, synchronized by a global clock. In each round, each agent moves in discrete steps from node to node along the arcs of graph $\vec{G}$. A *configuration* at the current step is defined as a triple $((\rho_v)_{v\in V}, (\pi_v)_{v\in V}, \{r_1, \ldots, r_k\})$, where ρ_v is a cyclic order of the arcs (in graph $\vec{G}$) outgoing from node v, π_v is an arc outgoing from node v, which is referred to as *the (current) port pointer at node* v, and $\{r_1, \ldots, r_k\}$ is the (multi-)set of nodes currently containing an agent. For each node $v \in V$, the cyclic order ρ_v of the arcs outgoing from v is fixed at the beginning of exploration and does not change in any way from step to step (unless an edge is dynamically added or deleted as discussed in the previous section). For an arc (v, u), let $next(v, u)$ denote the arc next after arc (v, u) in the cyclic order ρ_v.

The exploration starts from some initial configuration and then keeps running in all future rounds, without ever terminating. During the current round, first each agent i is moved from node r_i traversing the arc π_{r_i}, and then the port pointer π_{r_i} at node r_i is advanced to the next arc outgoing from r_i (that is, π_{r_i} becomes $next(\pi_{r_i})$). This is performed sequentially for all k agents. Note that the order in which agents are released within the same round is irrelevant from the perspective of the system, since agents are indistinguishable. For example, if a node v contained two agents at the start of a round, then it will send one of the agents along the arc π_v, and the other along the arc $(v, next(\pi_v))$. In some considerations, we will also assign explicit labels $\{0, 1, \ldots, \deg(v) - 1\}$ to the ports adjacent to v, in such a way that initially $\pi_v = 0$, and $next(v, i) = (v, (i + 1) \bmod \deg v)$. Then, at the completion of any round, the total number of traversals of agents along an arc (v, u) is equal to $\left\lceil \frac{e_v - port_v(u)}{\deg(v)} \right\rceil$, where e_v is the total number of times agents exited node v until the completion of the round and $port_u(v)$ denotes the label of the port leading from v to u.

In all our considerations, we will assume that the initialization of ports and pointers in the system is performed by an adversary. In particular, when studying a best-case scenario of initial agent locations, we assume that the ports and pointers have been set

by the adversary so as to maximize the studied parameter (e.g., cover time). For the case of the ring, there exists only one cyclic permutation of the two neighbors of each node, hence only the initial pointer arrangement (and not the configuration of ports) is relevant.

In our work we will consider both the unmodified k-agent rotor-router system $R[k]$ and its *delayed deployments*, in which some agents may be stopped at a node, skipping their move for some number of rounds. A delayed deployment D of k agents is formally defined as a function $D : V \times \mathbb{N} \to \mathbb{N}$, where $D(v,t) \geq 0$ represents the number of agents which are stopped in vertex v in round t of the execution of the system. (The rotor-router system $R[k]$ corresponds to the deployment $R[k](v,t) = 0$, for all v and t). Delayed deployments may be conveniently viewed as algorithmic procedures for delaying agents, and are introduced for purposes of analysis, only.

We will say that a node is *visited* by an agent in round t if the agent is located at this node at the start of round $t+1$. Let $\mathbf{n}_v^D(t)$ denote the total number of visits of agents to node v during the interval of rounds $[1,t]$ for agents following some (possibly delayed) deployment D, and let $C_{rr}^D(G)$ be the cover time of this deployment. The notation $\mathbf{n}_v^D(0)$ refers to the number of agents at a node directly after initialization (at the start of round 1). We will denote by $\mathbf{a}_e^D(t)$ the number of agents traversing directed arc $e \in \vec{E}$ during step $t+1$. Let $\mathbf{m}_e^D(t)$ denote the total number of traversals of arc $e \in \vec{E}$ during the interval of rounds $[1,t]$, $\mathbf{m}_e^D(t) = \sum_{t' \in [0,t)} \mathbf{a}_e^D(t')$. Let $\mathbf{e}_v^D(t)$ denote the total number of traversals of arcs outgoing from v during the interval of rounds $[1,t]$ in deployment D. For a node $v \in V$, let $\mathbf{r}_v^D(t) = \min_{w \in \Gamma(v)}\{\mathbf{m}_{v\to w}^D(t)\}$ be the number of fully completed rotations of the rotor at node v at the end of step t. For undelayed deployment $R[k]$ we will use notation $\mathbf{n}_v(t) = \mathbf{n}_v^{R[k]}(t)$, $\mathbf{m}_{v\to w}(t) = \mathbf{m}_{v\to w}^{R[k]}(t)$, $\mathbf{e}_v(t) = \mathbf{e}_v^{R[k]}(t)$, $\mathbf{r}_v(t) = \mathbf{r}_v^{R[k]}(t)$ and $\mathbf{e}_e(t) = \mathbf{e}_e^{R[k]}(t)$. We note that for any arc $u \to v \in \vec{E}$, $0 \leq \mathbf{n}_{u\to v}(t) - \mathbf{r}_u(t) \leq 1$ [144]. We recall that multiple agents traversing one arc $e \in \vec{E}$ in the same time step t are considered to move simultaneously. We also denote $V(t,i) = \{v \in V : \mathbf{n}_v(t) \leq i\}$ and $E(t,i) = \{e \in \vec{E} : \mathbf{m}_e(t) \leq i\}$. $\mathbb{N}_+$ denotes the set of positive integers, and $\mathbb{N} = \mathbb{N}_+ \cup \{0\}$. We will denote by $\mathcal{A}$ the set of all agents.

Symbol	*Description*
$\mathbf{n}_v^D(t)$	total number of visits to node v up to time t
$\mathbf{m}_e^D(t)$	total number of traversals of arc e up to time t
$\mathbf{e}_v^D(t)$	total number of traversals of arcs outgoing from v up to time t
$\mathbf{a}_e^D(t)$	number of agents traversing arc e during step $t+1$
$\mathbf{r}_v^D(t)$	number of completed rotations of the rotor at v at the end of step t

TABLE 3.3: Explanation of the notation. All symbols are for some (possibly delayed) deployment D.

We also introduce compact notation for discrete intervals of integers: $[a,b] \equiv \{a, a+1, \ldots, b\}$, and $[a,b) \equiv [a,b-1]$, for $a,b \in \mathbb{N}$. Given a graph $G = (V,E)$ and a subset $X \subseteq V$, $G[X]$ denotes the subgraph of G *induced by* X, $G[X] = (X, \{\{u,v\} \in E \mid u,v \in X\})$.

3.2 The delayed deployment technique for the multi-agent rotor-router

We start by showing that by delaying more agents in a deployment, one cannot increase the number of visits to nodes at any time. We assume that all considered deployments start from the same (arbitrarily chosen) initial configuration.

Lemma 3.1. *Let D_1 and D_2 be two delayed deployments of the k-agent rotor-router system, such that for all vertices $v \in V$ and rounds t, $D_1(v,t) \geq D_2(v,t)$. Then, for all vertices $v \in V$ and rounds t, we have $\mathbf{n}_v^{D_1}(t) \leq \mathbf{n}_v^{D_2}(t)$.*

Proof. For $t = 0$, the claim holds, since by definition:

$$\mathbf{n}_v^{D_1}(0) = \mathbf{n}_v^{D_2}(0) = \mathbf{n}_v^{R[k]}(0), \quad \text{for all } v \in V. \tag{3.1}$$

Recall that $\mathbf{e}_v^{D_1}(t)$ and $\mathbf{e}_v^{D_2}(t)$ denotes the total number of traversals of arcs outgoing from v during the interval of rounds $[1,t]$ for executions D_1 and D_2, respectively. For an arbitrary agent, the difference between the number of times the agent leaves v in rounds $[1,t+1]$ and the number of times it enters node v in rounds $[0,t]$ is equal to either -1 or 0, depending on whether the agent is delayed at v in round $t+1$ or not. Summing over all agents, we obtain:

$$\mathbf{e}_v^{D_i}(t+1) = \mathbf{n}_v^{D_i}(t) - D_i(v,t+1), \quad i \in \{1,2\} \tag{3.2}$$

The rest of the proof proceeds by induction on time t. Suppose that for some $t > 1$, $\mathbf{n}_v^{D_1}(t-1) \leq \mathbf{n}_v^{D_2}(t-1)$ holds for all $v \in V$. Then, we have from (3.2):

$$\mathbf{e}_v^{D_1}(t) + D_1(v,t) \leq \mathbf{e}_v^{D_2}(t) + D_2(v,t)$$

and since $D_1(v,t) \geq D_2(v,t)$:

$$\mathbf{e}_v^{D_1}(t) \leq \mathbf{e}_v^{D_2}(t), \quad \text{for all } v \in V. \tag{3.3}$$

Now, fix an arbitrary node u and observe that the number of visits to node u within the interval $[1,t+1]$ is equal to the sum of the number of agents placed at u in round 1, and the number of times an agent exited one of its neighbors $v \in \Gamma(u)$ along an arc (v,u) in rounds $[1,t]$:

$$\mathbf{n}_u^{D_i}(t+1) = \mathbf{n}_u^{D_i}(0) + \sum_{v \in \Gamma(u)} \left\lceil \frac{\mathbf{e}_v^{D_i}(t) - port_v(u)}{\deg(v)} \right\rceil, \tag{3.4}$$

where we took into account that agents leaving a node v exit along the ports adjacent to v in round-robin fashion. Combining expressions (3.1), (3.2), and (3.4) we obtain $\mathbf{n}_u^{D_1}(t+1) \leq \mathbf{n}_u^{D_2}(t+1)$. Since $u \in V$ was arbitrarily chosen, the inductive claim follows. □

We remark that the above lemma immediately implies that $\mathbf{n}_v^{R[k-1]}(t) \leq \mathbf{n}_v^{R[k]}(t)$, since the $(k-1)$-agent rotor-router $R[k-1]$ is equivalent to a deployment of the k-agent rotor-router with one agent permanently stopped. (This observation is due to [144].)

Lemma 3.2. *Let D be a delayed deployment of the k-agent rotor-router system. Let T be any fixed time round, and let τ be the number of rounds in the interval $[1,T]$ such that all the agents are active in D, i.e., $\tau = |\{t \in [1,T] : \forall_{v \in V}\ D(v,t) = 0\}|$. Then, for all vertices v, we have: $\mathbf{n}_v^{R[k]}(\tau) \leq \mathbf{n}_v^{D}(T) \leq \mathbf{n}_v^{R[k]}(T)$.*

Proof. The right inequality follows directly from Lemma 3.1. To prove the left inequality, we rewrite for round $t \geq 1$ the sets of recurrence equations (3.2) and (3.4) on the number of visits and exits to each node v for deployment D:

$$\begin{cases} \mathbf{e}_v^D(t) = \mathbf{n}_v^D(t-1) - D(v,t), \\ \mathbf{n}_v^D(t) = \mathbf{n}_v^D(0) + \sum_{w \in \Gamma(v)} \left\lceil \frac{\mathbf{e}_w^D(t-1) - port_w(v)}{\deg(w)} \right\rceil, \\ \mathbf{n}_v^D(0) = \mathbf{n}_v^{R[k]}(0), \quad \mathbf{e}_v^D(0) = 0. \end{cases}$$

Consider a function $f : [1,\tau] \to [1,T]$, with $f(i)$ being the i-th time round in which all agents are active in delayed deployment D. Denote by $F \subseteq [1,T]$ the image of f. Taking into account that $D(v,t) = 0$ for all $t \in F$ and that the counters $\mathbf{e}_v^D$ and $\mathbf{n}_v^D$ are always non-decreasing in time, we obtain the following set of inequalities by restricting evolution to moments of time $t = f(i)$, with $i \in [1,\tau]$:

$$\begin{cases} \mathbf{e}_v^D(f(i)) & = \mathbf{n}_v^D(f(i)-1) - D(v,f(i)) \geq \mathbf{n}_v^D(f(i-1)) - 0 = \mathbf{n}_v^D(f(i-1)), \\ \mathbf{n}_v^D(f(i)) & = \mathbf{n}_v^D(0) + \sum_{w \in \Gamma(v)} \left\lceil \frac{\mathbf{e}_w^D(f(i)-1) - port_w(v)}{\deg(w)} \right\rceil \geq \\ & \geq \mathbf{n}_v^D(f(0)) + \sum_{w \in \Gamma(v)} \left\lceil \frac{\mathbf{e}_w^D(f(i-1)) - port_w(v)}{\deg(w)} \right\rceil, \\ \mathbf{n}_v^D(f(0)) & = \mathbf{n}_v^{R[k]}(f(0)), \quad \mathbf{e}_v^D(f(0)) = 0, \end{cases}$$

where we put $f(0) = 0$ for convenience of notation. By comparing the above with the corresponding equations for the undelayed rotor-router $R[k]$, written for round $i \in [1,\tau]$:

$$\begin{cases} \mathbf{e}_v^{R[k]}(i) = \mathbf{n}_v^{R[k]}(i-1), \\ \mathbf{n}_v^{R[k]}(i) = \mathbf{n}_v^{R[k]}(0) + \sum_{w \in \Gamma(v)} \left\lceil \frac{\mathbf{e}_w^{R[k]}(i-1) - port_w(v)}{\deg(w)} \right\rceil, \\ \mathbf{e}_v^{R[k]}(0) = 0. \end{cases}$$

it follows by induction that $\mathbf{n}_v^D(f(i)) \geq \mathbf{n}_v^{R[k]}(f(i))$. Putting $i = \tau$, we obtain the sought inequality $\mathbf{n}_v^D(T) \geq \mathbf{n}_v^{R[k]}(\tau)$. □

Observe that by the above lemma, we have that if node v is visited for the first time after T rounds in a delayed deployment D, i.e., $\mathbf{n}_v^D(T) = 0$ and $\mathbf{n}_v^D(T+1) = 1$, then $\mathbf{n}_v^{R[k]}(\tau) = 0$ and $\mathbf{n}_v^{R[k]}(T+1) \geq 1$. From this, we directly obtain the key lemma for the approach we use to analysing the cover time of k-rotor-router systems in this chapter.

Lemma 3.3 (**the slow-down lemma**). *Let $R[k]$ be a k-rotor router system on any graph G with an arbitrarily chosen initialization, and let D be any delayed deployment of $R[k]$. Suppose that deployment D covers all the vertices of the graph after $T = C_{rr}^{D}(G)$ rounds, and in at least τ of these rounds, all agents were active in D. Then, the cover time $C_{rr}^{R[k]}(G)$ of the system can be bounded by:*

$$\tau \leq C_{rr}^{R[k]}(G) \leq T.$$

□

If the deployment D is defined so that agents in D are delayed in at most a constant proportion of the first $C_{rr}^{D}(G)$ rounds, then the above inequalities lead to an asymptotic bound on the value of the undelayed rotor-router, $C_{rr}^{R[k]}(G) = \Theta(C_{rr}^{D}(G))$. This is the case, e.g., in the proof of Theorem 3.15.

3.3 Upper bound on cover time for general graphs

In this section, we will show that a k-agent parallel rotor-router system explores a graph in $O(mD/\log k)$ steps, regardless of initialization. We start by providing an informal intuition of the main idea of the proof. After some initialization phase of duration t_0, but before exploration is completed at time $C_{rr}^{k}(G)$, we consider a shortest path connecting some arc of the graph which has already been visited many times at time t_0, with an arc which will remain unvisited at time $C_{rr}^{k}(G)$. We look at the number of visits to consecutive arcs on this path. It turns out that the rotor-router admits a property which can be informally stated as follows: if, up to some step t of exploration, an arc e_{l+1} of the considered path has been traversed more times than the next arc e_l on the path by some difference of δ, then in the next step $t+1$ of exploration, at least $\delta - O(1)$ agents will traverse arcs which have, so far, been visited not more often (up to a constant additive factor) than e_l. In this way, the larger the discrepancy between the number of visits to adjacent arcs, the more activity will the rotor-router perform to even out this discrepancy, by traversing under-visited arcs. This load-balancing behavior of the system will be shown to account for the $(\log k)$-speedup in cover time with respect to the case of a single agent.

We start by proving two structural lemmas which generalize the results of Yanovski *et al.* [144, Theorem 2]. The first lemma establishes a connection between the existence of an arc entering a subset of nodes $S \subseteq V$ that has been traversed more times than all arcs outgoing from S, and the number of agents currently located within set S.

Lemma 3.4. *For any time $t \in \mathbb{N}$ and $d \in \mathbb{N}$, consider the partition of the set of nodes $V = S \cup T$ such that each node in set S (set T) has completed at most d (more than d) full cycles of if its rotor, $S = V_d^{(t)}$ and $T = V \setminus S$. Suppose that for some nodes $v \in S$, $u \in T$, and some $\delta \in \mathbb{N}$, there exists an arc $u \to v$, such that $\mathbf{m}_{u\to v}(t) \geq d + \delta$. Then,*

the set of arcs having their tail at a node of S will be traversed by at least $\delta - 1$ agents in total in step $t + 1$.

Proof. Denote by $S \to T$ (resp., $T \to S$) the set of arcs connecting nodes from S with nodes from T (resp., nodes from T with nodes from S), and let $l = |S \to T| = |T \to S|$. By the basic property of the rotor-router process, all arcs outgoing from some node w have been traversed either $\mathbf{r}_w(t)$ or $\mathbf{r}_w(t) + 1$ times by the end of step t. It follows the definition of sets S and T that any arc outgoing from S was traversed at most $d+1$ times and any arc outgoing from T was traversed at least $d+1$ times. The arc $u \to v \in T \to S$ was traversed $d + \delta$ times. Hence:

$$\sum_{e \in S \to T} \mathbf{m}_e(t) \leq l \cdot (d+1),$$
$$\sum_{e \in T \to S} \mathbf{m}_e(t) \geq (l-1) \cdot (d+1) + d + \delta \geq \sum_{e \in S \to T} \mathbf{m}_e(t) + \delta - 1.$$

Thus, at least $\delta - 1$ more agents moved from T to S than in the opposite direction until the end of step t. So, at the end of time step t, we have at least $\delta - 1$ agents located at nodes from set S. It follows that during step $t + 1$, at least $\delta - 1$ agents traverse arcs outgoing from nodes from the set S. □

By an application of the above lemma, we obtain the key property of a pair of consecutive arcs which have a different number of traversals at time t.

Lemma 3.5. *Let $G = (V, E)$ be any undirected graph and let $e_2 = u \to v$, $e_1 = v \to w$ be two consecutive arcs of $\vec{G}$. Fix a time step $t \in \mathbb{N}_+$. Then, for any $x \geq \mathbf{m}_{e_1}(t) + 1$, the number of agents that traverse arcs from set $E(t, x)$ in time step $t + 1$ satisfies:*

$$\textstyle\sum_{e \in E(t,x)} \mathbf{a}_e(t) \geq \mathbf{m}_{e_2}(t) - \mathbf{m}_{e_1}(t) - 1.$$

Proof. We can assume that $\mathbf{m}_{e_2}(t) - \mathbf{m}_{e_1}(t) \geq 2$, otherwise the claim is trivial. By the definition of the rotor-router, we know that $0 \leq \mathbf{m}_{e_1}(t) - \mathbf{r}_v(t) \leq 1$ and $\mathbf{r}_u(t) \geq \mathbf{m}_{e_2}(t) - 1 \geq \mathbf{m}_{e_1}(t) + 1 \geq \mathbf{r}_v(t)$. We now apply Lemma 3.4 for $r = \mathbf{r}_v(t)$, putting $S = V(t, \mathbf{r}_v(t))$ and $T = V \setminus S$. Note that $v \in S$, $u \in T$, and $\mathbf{m}_{u \to v}(t) = r + \delta$ for $\delta = \mathbf{m}_{e_2}(t) - \mathbf{r}_v(t) \geq \mathbf{m}_{e_2}(t) - \mathbf{m}_{e_1}(t)$. It follows from the Lemma that during step $t + 1$, at least $\mathbf{m}_{e_2}(t) - \mathbf{m}_{e_1}(t) - 1$ agents traverse arcs outgoing from nodes from the set S. Since $S = V(t, \mathbf{r}_v(t))$, all arcs e^* outgoing from nodes from set S have a number of traversals which satisfies $\mathbf{m}_{e^*}(t) \leq \mathbf{r}_v(t) + 1 \leq \mathbf{m}_{e_1}(t)) + 1$, so $e^* \in E(\mathbf{m}_{e_1}(t) + 1, t)$. Thus, $\mathbf{m}_{e_2}(t) - \mathbf{m}_{e_1}(t) - 1$ agents in step $t + 1$ traverse edges in $E(t, \mathbf{m}_{e_1}(t) + 1)$, and moreover $E(t, \mathbf{m}_{e_1}(t) + 1) \subseteq E(t, x)$ for all $x \geq \mathbf{m}_{e_1}(t) + 1$. □

The property of the rotor-router captured by the above lemma is, in fact, sufficient to prove the main results of the section, following the general approach outlined at the beginning of the section. To show a bound of $C^k_{rr}(G) \in O(mD/\log k)$, we will apply two separate arguments, first one for the range of relative small k ($k \in 2^{O(D)}$, which

corresponds to $C^k_{rr}(G) \in \Omega(m)$), and then one for values of k which are larger, but polynomially bounded with respect to n.

Theorem 3.6. *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. If $k \leq 2^{16D}$, then a team of k agents performing in parallel the rotor-router movement explores G in less than $500mD/\log k$ steps, regardless of the initial positions of agents.*

Proof. First, assume that $k > 2^{160}$ and fix $b = \lfloor (\log k)/2 \rfloor$. Consider the first t_0 steps, where $t_0 = \lceil 2^{b+1}mD/k \rceil$. Since in every step exactly k arcs are traversed by agents, the total number of arc traversals during the first t_0 steps is at least $2^{b+1}mD$. We have $2m$ arcs in total. Thus, there exists an arc e' such that $\mathbf{m}_{e'}(t_0) \geq 2^b D$. These first t_0 steps we will call as a form of setup stage, after which we begin to analyze the behavior of the rotor-router process.

Denote by $C^k_{rr}(G)$ the cover time of G with k agents for a given initialization. We will assume that $C^k_{rr}(G) > t_0$, i.e., at least one arc of the graph has not been explored at time t_0; otherwise, $C^k_{rr}(G) \leq t_0 = \lceil 2^{b+1}mD/k \rceil \leq \lceil 2mD/\sqrt{k} \rceil$, since $b = \lfloor (\log k)/2 \rfloor$, and the claim of the theorem holds for all k.

Take $e'' \in \vec{E}$ to be an arc which is explored for the first time in step $C^k_{rr}(G)$, i.e., such that $d^{(C^k_{rr}(G)-1)}(e'') = 0$. Since the diameter of G is D, there exists a path $\mathcal{P} = \langle e'' = e_1, e_2, \dots e_{D'} = e' \rangle$ such that $D' \leq D+2$, and for each $l \in [1, D']$, $e_l = v_{l+1} \to v_l$ where $v_l, v_{l+1} \in V$.

Fix a time step $t \in [t_0, C^k_{rr}(G))$. We will place some of the arcs of path $\mathcal{P}$ in groups (buckets) $I_1, I_2, \dots, I_b$, such that all arcs in bucket I_i have been traversed between $2^{i-1}D$ and $2^i D$ times until step t. Formally, denote:

$$I_i = \left\{ l : \mathbf{m}_{e_l}(t) \in [2^{i-1}D, 2^i D) \right\} \subseteq [1, D'], \quad \text{for } i \in [1, b].$$

We now analyze which buckets successive arcs of the path $\mathcal{P}$ fall into. For $l \in [1, D')$, define

$$\Delta_l = \begin{cases} [\mathbf{m}_{e_l}(t), \mathbf{m}_{e_{l+1}}(t)), & \text{if } \mathbf{m}_{e_l}(t) < \mathbf{m}_{e_{l+1}}(t), \\ \emptyset, & \text{otherwise.} \end{cases}$$

Note that the union of all Δ_l covers the interval $[0, 2^b D)$, since for any $x \in [0, 2^b D)$ there exists $l^* \in [1, D')$ such that $x \in \Delta_{l^*}$ because $\mathbf{m}_{e_1}(t) = 0$ and $\mathbf{m}_{e_{D'}}(t) \geq 2^b D$ (see Fig. 3.1 for an illustration). The intuition of the proof is now as follows: Since there are at most D' non-empty intervals Δ_l spanning the total range $[0, 2^b D)$ of all buckets $I_1, I_2, \dots, I_b$, in a constant proportion of all buckets I_i, the average length of an intervals Δ_l starting in bucket I_i will be at least $|I_i|b/D = 2^{i-1}b$, up to a constant factor. The existence of such long intervals Δ_l beginning in I_i will allow us to exploit Lemma 3.5 to show that arcs e_l, e_{l+1} differ in the number of traversals by a constant times $2^{i-1}b$. This implies that for the considered bucket indices i, the number of agents active at time t on edges from buckets $I_1, \dots, I_i$ will be at least $2^{i-1}b$, up to constant factors and minor shifts at bucket boundaries. We now proceed to formalize the above arguments.

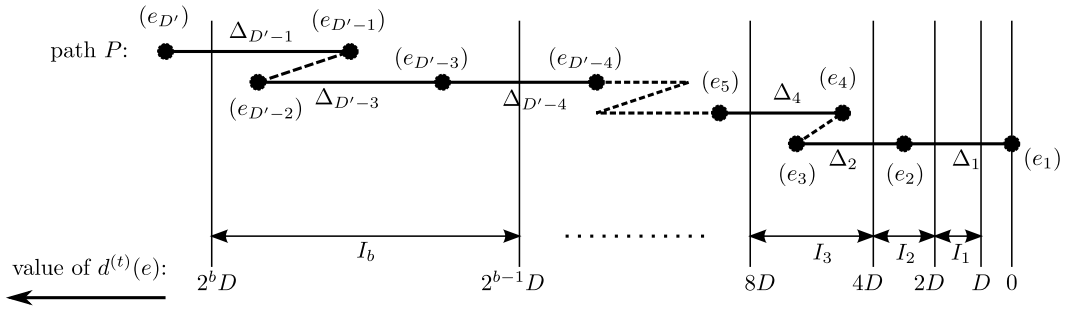

FIGURE 3.1: An illustration of sets I_i and Δ_l in the proof of Theorem 3.6.

For $i \in [1,b]$, denote by $\mathcal{X}_i$ the set of intervals Δ_l beginning in bucket I_i: $\mathcal{X}_i = \bigcup_{l\in I_i} \Delta_l$. Consider any $x \in [0, 2^b d)$, and let l^* be such that $x \in \Delta_{l^*}$. We have $\mathbf{m}_{e_{l^*}}(t) \leq x < 2^bD$, hence $l^* \in I_{i^*}$, for some $i^* \in [1,b]$, and $x \in \mathcal{X}_{i^*}$. It follows that:

$$[0, 2^bD) \subseteq \bigcup_{i\in[1,b]} \mathcal{X}_i. \tag{3.5}$$

For $i \in \mathbb{N}$, denote by $a_i(t)$ the number of agents that traverse arcs from set $E(t, 2^iD)$ in step $t+1$, $a_i(t) \equiv \sum_{e\in E(t,2^iD)} \mathbf{a}_e(t)$, and let $a_{-1}^{(t)} = 0$. (We remark that $E(t, 2^iD) \supseteq I_1 \cup \ldots \cup I_i$.) First, note that for all $i \in [1,b]$ and for $l \in I_i$, we have $\mathbf{m}_{e_l}(t) < 2^iD$. So, by Lemma 3.5:

$$a_i(t) \geq \mathbf{m}_{e_{l+1}}(t) - \mathbf{m}_{e_l}(t) - 1 = |\Delta_l| - 1 \quad \Longrightarrow \quad |\Delta_l| \leq a_i(t) + 1. \tag{3.6}$$

Now, observe that for any $i \in [1,b]$:

$$\max \mathcal{X}_i = \max_{l\in I_i} (\max \Delta_l) \leq \max_{l\in I_i} (\mathbf{m}_{e_l}(t) + |\Delta_l| - 1) < 2^iD + a_i(t), \tag{3.7}$$

where we took into account inequality (3.6) and that $\mathbf{m}_{e_l}(t) < 2^iD$ for $l \in I_i$.

Next, we will show that for all $i \in [1,b]$:

$$2^{i-1}D - a_{i-1}(t) \leq |\mathcal{X}_i| \leq |I_i|(a_i(t) + 1). \tag{3.8}$$

The right inequality in (3.8) is proved as follows: $|\mathcal{X}_i| \leq \sum_{l\in I_i} |\Delta_l| \leq |I_i|(a_i(t) + 1)$, where the latter inequality is a consequence of (3.6).

We now prove the left inequality in (3.8). If $a_{i-1}(t) \geq 2^{i-1}D$, then the bound is trivial. In the case when $a_{i-1}(t) < 2^{i-1}D$, we will first prove that:

$$[2^{i-1}D + a_{i-1}(t), 2^iD) \subseteq \mathcal{X}_i. \tag{3.9}$$

To this end, take any $x \in [2^{i-1}D + a_{i-1}(t), 2^iD)$ and observe that by (3.5), there exists some $j \in [1,b]$ such that $x \in \mathcal{X}_j$. Moreover, note that:

1. For any $j < i$, $x \notin \mathcal{X}_j$, because, by (3.7), $\max \mathcal{X}_j < 2^jD + a_j(t) \leq 2^{i-1}D + a_{i-1}(t) \leq x$.

2. For any $j > i$, $x \notin \mathcal{X}_j$, because: $\min \mathcal{X}_j = \min_{l \in I_j, \Delta_l \neq \emptyset} \min \Delta_l = \min_{l \in I_j, \Delta_l \neq \emptyset} \mathbf{m}_{e_l}(t) \geq 2^{j-1}D \geq 2^j D > x$.

Thus, $x \in \mathcal{X}_i$, and (3.9) follows. Equation (3.9) implies that $|\mathcal{X}_i| \geq 2^{i-1}D - a_{i-1}(t)$, which completes the proof of (3.8). Next, by (3.8),

$$|I_i| \geq \frac{2^{i-1}D - a_{i-1}(t)}{a_i(t)+1} \quad \text{for all } i \in [1,b].$$

The buckets $I_1, I_2, \ldots, I_b$ are pairwise disjoint by definition and contain at most D' elements altogether, which gives:

$$D+2 \geq D' \geq \sum_{i=1}^{b} |I_i| \geq \sum_{i=1}^{b} \frac{2^{i-1}D - a_{i-1}(t)}{a_i(t)+1} \geq \sum_{i=1}^{b} \frac{2^{i-1}D}{a_i(t)+1} - b,$$

where in the last inequality we used the fact that $a_i(t) \geq a_{i-1}(t)$ for $i \in [2,b]$. Dividing the sum in the last inequality by bD, we get the following expression for the arithmetic average:

$$\frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i(t)+1} \leq \frac{D+b+2}{bD} = \frac{1}{b} + \frac{1+2/b}{D} < \frac{9.2}{b},$$

where in the last inequality we took into account that $k \leq 2^{16D}$ and $b \leq (\log k)/2$ by assumption, hence $D \geq (\log k)/16 \geq b/8$, and that $b = \lfloor (\log k)/2 \rfloor \geq 80$. All the elements of the considered sum are positive, hence by Markov's inequality, there exists a subset of indices $S^{(t)} \subseteq [1,b]$, with $|S| \geq b/2$, such that for all $j \in S^{(t)}$ we have:

$$\frac{2^{j-1}}{a_j(t)+1} \leq 2 \cdot \frac{1}{b}\sum_{i=1}^{b} \frac{2^{i-1}}{a_i(t)+1} \leq \frac{18.4}{b}.$$

This implies that for all $j \in S^{(t)}$:

$$a_j(t) \geq \frac{b}{18.4} \cdot 2^{j-1} - 1 > \frac{b}{25} \cdot 2^{j-1}, \tag{3.10}$$

where we again took into account that $b \geq 80$.

Fix $t_1 = \lceil 100mD/b \rceil$. We now prove that

$$C^k_{rr}(G) \leq t_0 + 2t_1 + 4m. \tag{3.11}$$

Suppose, by contradiction, that $C^k_{rr}(G) > t_0 + 2t_1 + 4m$. We will say that an index $j \in [1,b]$ is *good* after time t if $j \in S^{(t)}$. Since for all $t \in [t_0, C^k_{rr}(G))$ we have $|S^{(t)}| \geq b/2$ and $S^{(t)} \subseteq [1,b]$, by the pigeon-hole principle there must exist an index j^* that is good in at least $(C^k_{rr}(G) - t_0)/2 = t_1 + 2m$ steps in $[t_0, C^k_{rr}(G))$; we will call these steps *good steps*.

For an arc e of the graph, we denote by t_e the so called *exit time step* for arc e, after which the total number of visits to arc e of the graph for the first time exceeds $2^{j^*}D$:

$\mathbf{m}_e(t_e) \leq 2^{j^*}D < \mathbf{m}_e(t_e+1)$. The set of all exit time steps, taken over all arcs of the graph, is denoted $\hat{T} = \{t_e : e \in \vec{E}\}$. Note that $e \in E(t, 2^{j^*}D)$ if and only if $t \leq t_e$, and therefore we may write:

$$\sum_{t\in[0,C^k_{rr}(G))\setminus\hat{T}} a_{j^*}(t) = \sum_{t\in[0,C^k_{rr}(G))\setminus\hat{T}} \sum_{e\in E(t,2^{j^*}D)} a_e(t) \leq \sum_{e\in\vec{E}} \sum_{t=0}^{t_e-1} a_e(t) = \sum_{e\in\vec{E}} \mathbf{m}_e(t_e) \leq 2m\cdot 2^{j^*}D. \tag{3.12}$$

Now, recall that there are at least $t_1 + 2m$ good time steps $t \in [t_0, C^k_{rr}(G))$ for which index j^* satisfies (3.10), and that $|\hat{T}| \leq 2m$. It follows that:

$$\sum_{t\in[0,C^k_{rr}(G))\setminus\hat{T}} a_{j^*}(t) > t_1 \cdot \frac{b}{25}\cdot 2^{j^*-1} = \left\lceil \frac{100mD}{b} \right\rceil \frac{b}{25}\cdot 2^{j^*-1} \geq 2m\cdot 2^{j^*}D,$$

a contradiction with (3.12). Thus, we have proved (3.11).

By (3.11), we obtain

$$\begin{aligned} C^k_{rr}(G) &\leq t_0 + 2t_1 + 4m = \left\lceil \frac{2^{b+1}mD}{k} \right\rceil + 2\left\lceil \frac{100mD}{b} \right\rceil + 4m \leq \\ &\leq \frac{mD}{\log k}\left(\frac{2^{b+1}\log k}{k} + \frac{200\log k}{b} + \frac{4\log k}{D} + \frac{3\log k}{mD}\right) \end{aligned} \tag{3.13}$$

Taking into account that $b = \lfloor(\log k)/2\rfloor$, $k \leq 2^{16D}$, and $k > 2^{160}$, we obtain that the expression in the above bracket can be bounded by a constant, giving: $C^k_{rr}(G) < 500\frac{mD}{\log k}$. This completes the proof for the case $k > 2^{160}$.

Suppose now that $k \leq 2^{160}$. Yanovski *et al.* [144] showed that a single agent explores the graph in at most $2mD$ steps regardless of the initialization, and moreover, that adding agents cannot decrease the number of traversals on any edge. We thus trivially obtain the claim: $C^k_{rr}(G) \leq 2mD < 500\frac{mD}{\log k}$. □

We now consider the case when $k \geq 2^{16D}$. Here, we first make the additional assumption that each agent starts from a distinct node. We show that additional assumption implies that no arc is traversed by more than one agent in a single step. The proof then proceeds along similar lines as that of Theorem 3.6, and we show that in many time steps t, there exists a pair of arcs e_{l+1}, e_l in $\mathcal{P}$ with a large difference in the number of traversals up to time t. However, instead of counting the number of long arcs on path $\mathcal{P}$ belonging to a bucket I_i, in this proof we take advantage of the fact that the length of the path $D' \leq D+2$ is small compared to $\log k$, which can be used to infer the existence of the sought arc pairs.

Lemma 3.7. *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. If $k \geq 2^{16D}$, then a team of k agents performing parallel rotor-router movement, with each agent starting from a distinct node of the graph, explores G in time $16mD/\log k$.*

Proof. We first prove that in every step $t \in \mathbb{N}$ of the exploration, every arc is traversed by at most one agent. Assume, to the contrary, that $t^* \in \mathbb{N}$ is the first step when two agents traverse the same arc, and let this arc be $e = u \to v$. Then, by virtue of the rotor-router principle, the number of agents located at u at the end of step $t^* - 1$ must have been at least $\deg(u) + 1$. This in particular implies that $t^* > 1$. Since there are exactly $\deg(u)$ incoming arcs to u, one of them was traversed by more than one agent in step $t^* - 1$. This contradicts the minimality of t^*. Thus, we have $a_e(t) \leq 1$, for all $e \in \vec{E}$ and $t \in \mathbb{N}$.

Denote by $C^k_{rr}(G)$ the cover time of graph G. For $i \in \mathbb{N}_+$, let $X = \lfloor k^{1/(2D+6)} \rfloor$ and let $Y_i = \sum_{j=0}^{i-1} X^j = \frac{X^i - 1}{X-1}$. Note that since $k \geq 2^{16D}$, we have:

$$X \geq 2 \quad \text{and} \quad Y_i < X^i \text{ for all } i \in \mathbb{N}. \tag{3.14}$$

Similarly as in proof of Theorem 3.6, we first consider a setup phase, consisting of steps $[1, t_0)$ of exploration, this time defining t_0 as:

$$t_0 = 2 \left\lceil mX^{2D+5}/k \right\rceil \leq 2\lceil m/X \rceil. \tag{3.15}$$

During the setup stage, the total number of edge traversals is at least $2mX^{2D+5}$. Thus, there exists an arc e' such that $\mathbf{m}_{e'}(t_0) \geq X^{2D+5}$. There also exists an arc e'' such that $\mathbf{m}_{e''}(C^k_{rr}(G) - 1) = 0$. Thus, for each $t \in [t_0, C^k_{rr}(G))$,

$$\mathbf{m}_{e''}(t) = 0 \text{ and } \mathbf{m}_{e'}(t) \geq X^{2D+5} > Y_{2D+5}. \tag{3.16}$$

Since D is the diameter of G, there exists a path $\mathcal{P} = \langle e'' = e_1, e_2, \ldots, e_{D'} = e' \rangle$, such that $D' \leq D + 2$ and for all $i \in [1, D')$, $e_i = v_i \to v_{i+1}$ where $v_i, v_{i+1} \in V$.

For each time step t and $i \geq 2$, let $a_i(t)$ be the number of agents that during step $t + 1$ traverse those arcs which were traversed at most Y_i times until the end of step t, $a_i(t) \equiv \sum_{e \in E(t, Y_i)} \mathbf{a}_e(t)$. We have for any $i \geq 2$:

$$\sum_{t=t_0}^{C^k_{rr}(G)-1} a_i(t) \leq 2m(Y_i + 1) < 3mY_i, \tag{3.17}$$

because otherwise we would have an arc e that contributes at least $Y_i + 2$ to the above sum. Then, since in each time step $t \in N$ each arc is traversed at most once, there exist steps $t_0 < t_1 < t_2 < \cdots < t_{Y_i+2} \leq C^k_{rr}(G)$ in which e is traversed, and moreover $e \in E(t_{Y_i+2} - 1, Y_i)$. However, till the end of step $t_{Y_i+2} - 1 \geq t_{Y_i+1}$ the arc e has been traversed $Y_i + 1$ times, so, $e \notin E(t_{Y_i+2} - 1, Y_i)$, and we obtain a contradiction, proving (3.17).

We now prove that

$$C^k_{rr}(G) \leq t_0 + 6 \left\lceil \frac{m}{X - 1} \right\rceil. \tag{3.18}$$

Suppose, by contradiction, that $C_{rr}^k(G) > t_0 + 6\lceil m/(X-1)\rceil$. For each time step t, we will call the set of arcs $E(t, Y_i) \setminus E(t, Y_{i-1})$ the *i-th zone* at time t, for $i \geq 2$.

Each zone that does not contain any arc of path $\mathcal{P}$ in a given time step is called *free*. The path $\mathcal{P}$ has at most D' arcs and hence at least D' zones with indices in the interval $[2, 2D'+1]$ are free in each time step. Thus, by the pigeonhole principle, during the time period $[t_0, C_{rr}^k(G))$ there must exist an index $i^* \in [2, 2D'+1]$ such that the i^*-th zone is free during a set of time steps $T \subseteq [t_0, C_{rr}^k(G))$, with:

$$|T| \geq (C_{rr}^k(G) - t_0)/2 > 3\lceil m/(X-1)\rceil.$$

By (3.16), the arc e' belongs to a zone with index at least $2D+6 \geq 2D'+2$ in each time step $t \in T$, while arc e'' belongs to zone 1. Since the i^*-th zone is free at time t, by following path $\mathcal{P}$ from arc e' to e'', we will necessarily encounter an index $j \in [1, D')$, such that $\mathbf{m}_{e_{j+1}}(t) \geq Y_{i^*+1} + 1$ and $\mathbf{m}_{e_j}(t) \leq Y_{i^*}$, which gives:

$$\mathbf{m}_{e_{j+1}}(t) - \mathbf{m}_{e_j}(t) \geq Y^{i^*+1} + 1 - Y^{i^*} = X^{i^*} + 1.$$

By Lemma 3.5, for each $t \in T$, at least X^{i^*} agents traverse arcs from set $E(t, Y_{i^*})$ in step $t+1$, i.e., $a_{i^*}(t) \geq X^{i^*}$. Thus,

$$\sum_{t \in T} a_{i^*}(t) \geq |T| X^{i^*} \geq 3 \left\lceil \frac{m}{X-1} \right\rceil X^{i^*} > 3mY_{i^*}.$$

This contradicts (3.17), completing the proof of (3.18). Note that:

$$X = \left\lfloor k^{1/(2D+6)} \right\rfloor$$

By (3.18), (3.15), and the definition of X, we have:

$$C_{rr}^k(G) \leq 2\left\lceil \frac{m}{X} \right\rceil + 6\left\lceil \frac{m}{X-1} \right\rceil \leq 8\frac{m}{X-1} + 8 \leq \frac{mD}{\log k}\left(\frac{\log k}{D(k^{1/(8D)} - 2)} + 8\right).$$

Observe that for fixed D, the expression in the above bracket is strictly decreasing with k for $k > 2^{8D}$, and for $k = 2^{16D}$ takes a value of 16. Knowing that $k \geq 2^{16D}$, we therefore obtain $C_{rr}^k(G) < 16\frac{mD}{\log k}$. □

It remains to consider the case not covered by the above lemma, when not all agents start from distinct positions. In fact, we will reduce such a case to the one already considered by making use of the concept of delayed deployments discussed in Section 3.1.1. Lemma 3.3 has the following direct corollary.

Lemma 3.8. *Let R and R' be two starting configurations of the k-agent rotor-router system with cover times $C_{rr}^k(G)$ and t'_C, respectively. Suppose that there exists a delayed deployment D of R whose execution transforms the starting configuration of R into the starting configuration of R' in $\hat{t}$ time steps. Then, $C_{rr}^k(G) \leq \hat{t} + t'_C$.*

Proof. Observe that the concatenation of the execution of deployment D for $\hat{t}$ steps and R' for t'_C steps is a delayed deployment of R which explores the graph in $C^k_{rr}(G) \leq \hat{t} + t'_C$ steps. The claim follows by Lemma 3.3. □

The next lemma provides an upper bound on the time of transforming a rotor-router configuration with at most n agents into one in which agents occupy distinct starting nodes.

Lemma 3.9. *For any initialization R of the rotor-router system with k agents, $k \leq n$, there exists a delayed deployment D of R which terminates in a configuration in which all agents occupy distinct positions after $\hat{t} \leq k^4$ steps.*

Proof. In deployment D, we release agents sequentially from their starting positions in R, moving one agent only at a time until it is located at a node unoccupied by another agent. Consider the phase in which we move a fixed agent a in this deployment. In the worst case, a has to explore the graph induced by all nodes occupied to date. The agent acts a single-agent rotor router system with respect to this graph. Recall that the cover time of a graph with m edges and diameter D by a single agent is at most $2mD$, regardless of the initial configuration [144]. Since in the considered system there are at most k occupied nodes with at most $k^2/2$ edges between them, and the graph of occupied nodes has diameter at most k, a finds an unoccupied node within $2 \cdot k^2/2 \cdot k = k^3$ steps. This has to be done by each of k agents, thus total time of all phases of the delayed deployment is $\hat{t} \leq k^4$. □

When $1 < k \leq \lceil n^{1/5} \rceil$, we can bound the time $\hat{t}$ in the above lemma as: $\hat{t} \leq k^4 \leq 32n/k \leq 64m/k \leq 128\frac{mD}{\log k}$.

Combining the above result with Lemmas 3.7 and 3.8, we obtain that for any rotor router initialization with k agents, $k \leq \lceil n^{1/5} \rceil$ and $k \geq 2^{16D}$, exploration is completed within time $C^k_{rr}(G) = \hat{t} + t'_C \leq 128\frac{mD}{\log k} + 16\frac{mD}{\log k} = 144\frac{mD}{\log k}$. On the other hand, when $k < 2^{16D}$, by Theorem 3.6, the cover time is $C^k_{rr}(G) \leq 500\frac{mD}{\log k}$. It follows that the bound $C^k_{rr}(G) \leq 500\frac{mD}{\log k}$ holds for all starting configurations with $k \leq \lceil n^{1/5} \rceil$.

When $k > \lceil n^{1/5} \rceil$, we can make use of a result of Yanovski *et al.* [144], stating that the worst-case initialization of a rotor-router system with k agents cannot have greater cover time than the worst-case initialization of a system with $k' < k$ agents. Putting $k' = \lceil n^{1/5} \rceil$, for any $k > \lceil n^{1/5} \rceil$ we obtain: $C^k_{rr}(G) \leq 500\frac{mD}{\log k'} \leq 2500\frac{mD}{\log n}$. Finally, combining the results for $k \leq \lceil n^{1/5} \rceil$ and $k > \lceil n^{1/5} \rceil$ gives the following theorem.

Theorem 3.10. *Let $G = (V, E)$ be any undirected graph with arbitrary initialization of pointers and let D be the diameter of G. A team of k agents performing in parallel the rotor-router movement explores G in time $\max\{500mD/\log k, 2500mD/\log n\}$, regardless of the initial positions of agents. In particular, if $k \leq n^c$ for some $c > 0$, then the cover time is at most $2500c \cdot mD/\log k$.* □

Theorems 3.6 and 3.10 imply that the cover time of the rotor-router is $O(mD/\log k)$ for all graphs, whenever $k \in 2^{O(D)}$ or $k \in O(poly(n))$. On the other hand, the cover time

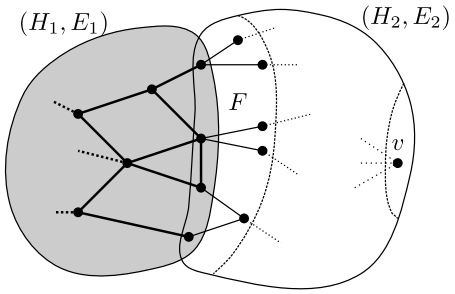

FIGURE 3.2: Graph decomposition used in the proof of Theorem 3.11.

of the rotor-router is trivially lower-bounded as $\Omega(D)$ for a team of agents starting from a single node, regardless of the number of agents. It follows that it is not possible to extend the bound of $O(mD/\log k)$ on cover time beyond the range $k \in 2^{O(n)}$. We leave as open the question of whether the considered bound can be achieved for the (rather special) range of values of k not covered by Theorems 3.6 and 3.10.

3.4 Lower bound on cover time for general graphs

In this section we show that for any graph G there exists an initialization of the k-agent rotor-router system that results in cover time of $\Omega(mD/k)$. This means that the rotor-router does not admit a synergy effect in the worst case initialization.

Theorem 3.11. *Let $G = (V, E)$ be any undirected graph of diameter D. There exists a port labeling of the edges of G, an initialization of pointers and an assignment of starting positions to a team of k agents, such that the exploration performed in parallel with the rotor-router movement has cover time $C_{rr}^k(G) \geq \frac{1}{4}mD/k$.*

Proof. If $k > m$, we make all agents start from an arbitrarily chosen single node, and choose an arbitrary pointer initialization. In such a scenario, the exploration will be completed after time at least $D > \frac{mD}{k}$. Thus, we can safely assume that $k \leq m$.

For any graph $G = (V, E)$, as shown in [15, Theorem 2], there exists a partition of the edge set $E = E_1 \cup E_2$, such that (see Fig. 3.2 for an illustration):

(i) $|E_1| \geq \frac{m}{2}$,

(ii) there exist $V_1 \subseteq V$ and $V_2 \subseteq V$ such that the subgraphs $H_1 = G[V_1]$ and $H_2 = G[V_2]$ are connected and their edge sets are E_1 and E_2, respectively,

(iii) there exists a node $v \in V_2$ being at distance at least $\frac{D}{2}$ from each node of H_1.

Denote by $F \subset E_2$ the set of edges incident to some node from H_1.

Now, let $C = \{e_1, e_2, \ldots, e_{2|E_1|}\}$ be a directed Eulerian cycle in $\vec{H}_1$ (the bidirected subgraph corresponding to H_1) traversing every edge in E_1 exactly once in each direction. To simplify notation, let $\Delta = \left\lfloor \frac{2|E_1|}{k} \right\rfloor$.

We choose an arbitrary set of indexes $1 = j_1 < j_2 < \ldots < j_k \leq 2|E_1|$ such that they are spread (almost-)equidistantly in $\{1, \ldots, 2|E_1|\}$, that is:

$$\forall_{1 \leq i < k} \quad j_{i+1} - j_i \in \{\Delta, \Delta + 1\} \quad \text{and} \quad j_1 - j_k + 2|E_1| \in \{\Delta, \Delta + 1\}.$$

This is possible because, due to (i), $2|E_1| \geq k$.
We partition the set of arcs $\vec{E}_1$ corresponding to edges from E_1 into Δ sets $\vec{S}_1, \ldots, \vec{S}_\Delta$ of size k:

$$\vec{S}_{i+1} = \{e_{j_1+i}, e_{j_2+i}, \ldots, e_{j_k+i}\}, \quad \text{for } 0 \leq i < \Delta,$$

and one set for all remaining edges: $\vec{R} = \vec{E}_1 \setminus \bigcup_{t=1}^{\Delta} \vec{S}_t$.

We choose the starting positions of k agents, the port assignment, and the initialization of pointers for the arcs in $\vec{E}_1$ such that in their first $\Delta + 1$ steps, the k agents traverse all arcs in $\vec{E}_1$ in the following delayed deployment: for each $t \in \{1, \ldots, \Delta\}$, in the t-th step, exactly the edges in $\vec{S}_t$ are traversed, whereas in the $(\Delta + 1)$-th step we delay some agents so that exactly the edges in $\vec{R}$ are traversed. We achieve this by setting outgoing ports so that, for every node u in H_1, we order the arcs in $\vec{E}_1$ incident to u by assigning smaller ports to edges in $\vec{S}_t$ than to the edges in $\vec{S}_{t+1}$, for each $t \in \{1, \ldots, \Delta\}$, where $\vec{S}_{\Delta+1} = \vec{R}$. Such a port ordering is enough to explore the graph H_1, with delayed deployment, with the property that every edge is visited once every $\Delta + 1$ steps.

Now we assign ports to the set of arcs $\vec{F}$ corresponding to edges from F. To this end, we consider the subgraph of G, denoted by $\widetilde{G}$, consisting of the edges in $E_1 \cup F$. In other words, we take H_1 (together with the port assignment obtained above) and we add the edges in F, obtaining $\widetilde{G}$. Note that, by (ii), each edge in F has one endpoint in V_1 and the other endpoint in $V \setminus V_1$. The ports on the arcs of F outgoing from V_1 are determined by analyzing the behavior of agents in the graph $\widetilde{G}$ in the delayed deployment described above. Whenever any set of agents are about to leave H_1 and traverse any arcs from $\vec{F}$, we select a single agent in a deterministic way (for example, by choosing the agent located on a node with the smallest index, having indexes assigned to nodes). We stop all other agents and perform traversals only with the selected agent, until it returns to H_1. We set the ports of the arcs in $\vec{F}$ so that whenever an agent leaves H_1 through an arc $(v \to u) \in \vec{F}$ $(v \in V_1, u \notin V_1)$, it returns to H_1 through the arc $(u \to v)$ (we call this property *the property of return*). Having the property of return, we achieve that the agents patrol E_1, and whenever an agent is about to leave H_1, the other agents are delayed until the agent returns to the same node. Since the selection of agents is done deterministically, the edges in F are always traversed in separated periods of time (when one agent is traversing edges from F, all other agents are stopped) in a cyclic fashion, i.e., the sequence of traversal of the arcs in $\vec{F}$ is $\left(f_1, f_1', f_2, f_2', \ldots, f_{|F|}, f_{|F|}'\right)^*$, where f' means the reversed arc to an arc f, i.e., if $f = (u \to v)$, then $f' = (v \to u)$. Denote $f_i = (u_i \to v_i)$ for each $i \in \{1, \ldots, |F|\}$.

It remains to assign port labels to the arcs in $\vec{E}_2 \setminus \vec{F}$, where $\vec{E}_2$ is the arc set of the set of edges E_2, and to initialize the pointers for the nodes in $V \setminus V(\widetilde{G})$. This is done

by first constructing a multigraph G' and then by analyzing a single agent movement in G'. The node set of G' is $\{h\} \cup (V \setminus V_1)$. For each $(u \to v) \in E_2 \setminus F$, let $(u \to v)$ be an edge of G', and for each $i \in \{1, \ldots, |F|\}$, let (h, v_i) and (v_i, h) be the edges of G'. In other words, we construct G' by taking G, leaving the edges in $E \setminus E_1$ untouched, and contracting (identifying) the nodes of H_1 into the single node h. (The loops at h formed by the edges in E_1 are discarded.) For each $i \in \{1, \ldots .|F|\}$, the ports of $(h \to v_i)$ and $(v_i \to h)$ equal the ports of (u_i, v_i) and (v_i, u_i), respectively.

We set the remaining ports in G' and pointer initialization so that a single agent that starts at h explores G' in the following way:

(a) The arcs from $\vec{F}$ are traversed according to the order

$$\big((h \to v_1), (v_1 \to h), (h \to v_2), (v_2 \to h), \ldots, (h \to v_{|F|}), (v_{|F|} \to h)\big) .$$

Later on, we use the port labeling of G' to assign port labels to the arcs in $\vec{E}_2$ in G, and the above allows us to maintain the return property in G.

(b) The agent requires at least $(D/2 - 1)$ traversals through each of the arcs in $\vec{F}$. This follows from the fact that, due to (iii), there exists a node in G' being at distance at least $D/2$ from h.

The above process assigns port labels to the arcs in $\vec{E}_2$ and sets initial values of all pointers in G', which completes the construction of G and the initial setup of the rotor-router.

Now we analyze the delayed deployment performed by the k agents in G. We divide the exploration of G into phases. The i-th phase starts in the step in which each edge in $\vec{S}_1$ is traversed for the i-th time, and ends in the step preceding the beginning of the $(i+1)$-th stage. Note that each stage contains at least Δ steps in which all agents move simultaneously. By (a), the property of return holds in G, and therefore each arc in $\vec{F}$ is traversed exactly once in each phase, except the first phase, when no arc from $\vec{F}$ is traversed. (This first phase comes from the fact that arcs from $\vec{E}_1$ have smaller port numbers than arcs from $\vec{F}$, in common vertices.) Thus, by (b), at least $D/2$ phases are required in the delayed deployment to explore G. This means that we need τ steps in which all agents move simultaneously to fully explore the graph G, where:

$$\tau \geq \Delta \cdot D/2 = \left\lfloor \frac{2|E_1|}{k} \right\rfloor \cdot D/2 \geq \left\lfloor \frac{m}{k} \right\rfloor \cdot D/2 \geq \frac{1}{4} mD/k$$

We can now apply Lemma 3.3 for the considered deployment, obtaining that the cover time of G is $C^k_{rr}(G) \geq \tau \geq \frac{1}{4}mD/k$. □

The bound in Theorem 3.11 is asymptotically tight for some graph classes, for example for stars. We leave the following simple observation without proof.

Proposition 3.12. *Let G be a star on n nodes. A team of $k \leq n$ agents covers G in time $C^k_{rr}(G) \leq 2\lceil n/k \rceil$, for any initialization of the rotor-router and any initial positions of agents.* □

3.5 Cover time on the ring

For a given (possibly delayed) deployment of the k-rotor-router system, such that no two agents ever occupy the same node at the same time, and a fixed round t, we consider the partition of the node set into so called *domains*. We set $V(t) = V_0(t) \cup V_1(t) \cup \ldots \cup V_k(t)$, where $V_0(t)$ denotes the set of nodes which have not yet been visited until round t, and $V_i(t)$, $1 \leq i \leq k$, is the set of all nodes such that the i-th agent was the last agent visiting the node until round t, inclusive. When the considered graph is a ring, we have the following simple characterization of the structure of the domains of particular agents in deployments in which agents never meet. We will denote the i-th agent by a_i. We state the following simple properties without proof.

Lemma 3.13. *Consider a deployment in which no two agents ever meet at a node, and let $v_i(t) \in V_i(t)$ be the location of the agent a_i at a given round t, $1 \leq i \leq k$. The following properties hold:*

- *$V_i(t)$ induces a sub-path of the ring.*
- *The pointers of all nodes $u \in V_i(t)$ point away from $v_i(t)$, i.e., not along the arc on the path leading from u to $v_i(t)$ in $V_i(t)$. In particular, if $v_i(t)$ is an end-point of the path induced by $V_i(t)$, then all the pointers of $V_i(t) \setminus \{v_i(t)\}$ point in the same direction.*
- *In each round, $V_i(t)$ loses or gains at most one node at each end of the path. In particular, $|V_i(t+1) \oplus V_i(t)| \leq 2$.* □

To provide an asymptotic description of the behavior of agent domains in time, we introduce the continuous-time approximation of the agents' behavior. This is useful under the assumption that the sizes of all the domains are sufficiently large, i.e., that the change of size of $V_i(t)$ in the number of rounds of the order $|V_i(t)|$ is negligible with respect to $|V_i(t)|$.

Suppose that the domains of the agents are ordered along the ring as $V_0(t), V_1(t), \ldots, V_k(t)$. Assuming that only the i-th agent is moving, the agent will reach each of the endpoints of its domain every $1/(2|V_i(t)|)$ rounds. Consequently, within T rounds, the agent enlarges its domain by approximately $T/(2|V_i(t)|)$ to the left, and $T/(2|V_i(t)|)$ to the right, thus by about $T/|V_i(t)|$ in total. This movement is counteracted by the moves of the adjacent agents occupying domains V_{i-1} and V_{i+1}. Consequently, we define the *continuous-time approximation* of the rotor-router through the set of differential equations:

$$\frac{d\nu_i(t)}{dt} = \frac{1}{\nu_i(t)} - \frac{1}{2\nu_{i-1}(t)} - \frac{1}{2\nu_{i+1}(t)}, \quad \text{for } 1 \leq i \leq k,$$

where $\nu_i(t) = |V_i(t)|$, for all $1 \leq i \leq k$. The interpretation of $\nu_0(t)$ and $\nu_{k+1}(t)$ depends on whether the whole ring has already been covered: if so, then $\nu_{k+1}(t) \equiv \nu_1(t)$ and $\nu_0(t) \equiv \nu_k(t)$; if not, i.e., if $|V_0(t)| > 0$, then we put $\nu_0(t) = \nu_{k+1}(t) = +\infty$.

Whereas the above differential model provides the basic intuition for many of the proofs, the main difficulty lies in taking into account the differences between the continuous-time model and the real rotor-router. In particular, we have to consider the position of the agent within its domain, the discrete changes of the domain size in time, and the initial pointer arrangement in the unvisited part of the ring.

The following lemma introduces a sequence $\{a_i\}_{i=0}^{k+1}$, useful in analyzing initial placements in which all agents start from the same point of the ring. It corresponds to a normalized solution to the continuous-time model of the rotor-router (i.e., $a_i(t) = \nu_i(t)/\sum_j \nu_j(t)$), subject to the constraint that the proportions of domain sizes do not change in time (i.e., $\frac{da_i(t)}{dt} = 0$), and specific boundary conditions.

Lemma 3.14. *For any $k > 3$ there exists a sequence of positive real numbers $(a_0, a_1, \ldots, a_k, a_{k+1})$ which satisfies the following properties:*

(1) $a_0 = +\infty$,

(2) $a_{k+1} = a_k < a_{k-1} < \ldots < a_1$,

(3) $\sum_{i=1}^{k} a_i = 1$,

(4) $a_i \cdot a_1 = \frac{2}{a_i} - \frac{1}{a_{i-1}} - \frac{1}{a_{i+1}}$, *for all* $1 \leq i \leq k$,

(5) $\frac{1}{4(H_k+1)} \leq a_1 \leq \frac{1}{H_k}$, *where* $H_k = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$ *denotes the k-th harmonic number,*

(6) $\frac{1}{4i(H_k+1)} \leq a_i$, *for all* $1 \leq i \leq k$.

Proof. For a fixed $c > 0$, consider the recursively defined sequence $\{b_i(c)\}_{i=0}^{+\infty}$: $b_0 = 0$, $b_1 = c$, $b_{i+1} = 2b_i - b_{i-1} - \frac{1}{b_i}$, where we write $b_i \equiv b_i(c)$ to simplify notation. Let $d_i = b_i - b_{i-1}$. Then, $d_1 = c$, and $d_{i+1} = d_i - \frac{1}{b_i}$. Expanding this recurrence, we have:

$$d_{i+1} = c - \left(\frac{1}{b_1} + \ldots \frac{1}{b_i}\right).$$

$$b_{i+1} = c - \left(\frac{1}{b_1} + \ldots \frac{1}{b_i}\right) + b_i = (i+1)c - \left(\frac{i}{b_1} + \frac{i-1}{b_2} + \ldots + \frac{1}{b_i}\right).$$

First, by a simple inductive argument we observe from the above that for sufficiently large values of $c = b_1$, arbitrarily many of the initial elements of sequences $\{b_i(c)\}$ and $\{d_i(c)\}$ are positive.

Next, fix $i \geq 3$ and suppose that $d_j > 0$, for all $1 \leq j \leq i$. Then:

$$b_i \leq ic.$$

Thus:

$$d_{i+1} \leq c - \left(\frac{1}{c} + \ldots \frac{1}{ic}\right) = c - \frac{H_i}{c}.$$

From the relation $d_{i+1} > 0$, we obtain $H_i < c^2$, so $i < e^{c^2+1}$. This implies that by adjusting $c \in (0, +\infty)$, we can arbitrarily choose the number of positive initial elements

of sequence $\{d_i(c)\}$. Taking into account that $d_i(c)$, for any fixed index i, is a continuous function of the parameter c, by the intermediate value theorem, there must exist a value of c such that $d_{k+1}(c) = 0$, or equivalently, that $b_{k+1} = b_k$. From now on, we use this value of c, only. Observe that $d_{k+1} = 0$ implies that $c = \sum_{i=1}^{k} 1/b_i$.

Now, define $a_i \equiv 1/(cb_i)$, for all $0 \leq i \leq k+1$. Such a sequence $\{a_i\}$ immediately satisfies conditions (1), (2), and (3). Condition (4) is obtained directly by observing that $a_1 = \frac{1}{c^2}$ and applying the replacement $b_i = 1/(ca_i)$ to the defining recursion of $\{b_i\}$.

Condition (5) may be restated as $H_k \leq c^2 \leq 4(H_k + 1)$. We have already established that the first of these relations holds, since otherwise we would have $d_{k+1} < 0$.

We will first show by induction that $d_i > c - \frac{2H_{i-1}}{c}$ for all $1 \leq i \leq e^{c^2/4}$. Indeed, the claim holds for $i = 1$. Suppose it holds for all $1 \leq j \leq i$. Then:

$$b_j = \sum_{l=1}^{j} d_l > cj - \frac{2}{c}\sum_{l=1}^{j} H_{l-1} = cj - \frac{2}{c}(jH_j - j),$$

$$d_{i+1} = c - \sum_{l=1}^{i} \frac{1}{b_l} > c - \sum_{l=1}^{i} \frac{1}{cl - \frac{2}{c}(lH_l - l)}.$$

Since $i < e^{c^2/4}$, then for any $l < i$ we have:

$$H_l < \log l + 1 < \frac{c^2}{4} + 1,$$

$$\frac{2}{c}(lH_l - l) < l\frac{c}{2}.$$

Thus

$$d_{i+1} > c - \sum_{l=1}^{i} \frac{1}{cl - l\frac{c}{2}} = c - \frac{2H_i}{c},$$

and the inductive claim holds. Now if $i < e^{\frac{c^2}{4}-1}$, then:

$$d_i > c - \frac{2H_{i-1}}{c} > c - \frac{2\log i + 2}{c} > c - \frac{c^2}{2c} = \frac{c}{2}.$$

Thus $k > e^{\frac{c^2}{4}-1}$, and we have:

$$c^2 \leq 4(\log k + 1) \leq 4(H_k + 1).$$

Since $b_i \leq ic$ then $a_i \geq 1/(ic^2)$ thus sequence $\{a_i\}$ satisfies condition (6). □

We are now ready to analyse a specific initialization, for which the k-agent rotor-router covers the ring particularly slowly.

Theorem 3.15. *In the case when all the agents are initially placed at the same node v, a group of k agents explores the ring of size n in time $\Theta(\frac{n^2}{\log k})$ when $k < n^{1/11}$, when all pointers are initialized along the shortest path to v.*

Proof. Consider a scenario with K agents on an N-node ring. Since $C_{rr}^{R[K-1]}(G) \geq C_{rr}^{R[K]}(G) \geq C_{rr}^{R[K+1]}(G)$, and the cover time is also monotonous with respect to the size of the ring, without affecting asymptotic bounds we can assume that K is even an N is odd, i.e., $K = 2k$ and $N = 2n - 1$. By induction, we can show that the number of agents at node v will be even at all times, and the arrangement of pointers on the ring (except for node v) is symmetric with respect to the axis of symmetry passing through v. Consequently, the cover time for the N-node ring with K agents is asymptotically the same as the cover time of a n-node path with k agents, starting from an initial placement of all agents on one of the end-points v of the path.

Let $R[k]$ be this deployment on the path P_n. We now propose a delayed deployment D of $R[k]$ in which, starting from a certain moment in time, the domains of all agents are separate. Let the domains be ordered along the path according to decreasing numbers, i.e., the agent with domain V_k is the one located closest to the starting point v, while the agent with domain V_1 is the furthest from v, i.e., it is the only agent to explore previously unvisited nodes of the path. The goal of the formalization below is to define the delayed deployment so that the ratios of domain sizes satisfy $|V_i| \sim a_i$, for $k \geq i \geq 1$, throughout time.

We will identify the path P_n with the integer interval $[1, n]$ (with $v = 1$), and domains with subsets of this interval. For $k \geq i \geq 1$, let $p_i = \sum_{j=i}^{k} a_i$. For a given value S, $n \geq S > 0$, we will call a configuration of agents and pointers on the path a *desirable configuration of length* S if it has the following properties:

- The position of the i-th agent on the path is $v_i = \lfloor p_i S \rfloor$.
- Each agent is at the right endpoint of its domain, i.e., $V_k = [1, v_k]$ and $V_i = [v_{i+1} + 1, v_i]$ for $k - 1 \geq i \geq 1$.
- For all the nodes on the path (including those containing agents), except for node 1, the pointer points to the left (towards node 1).

The evolution of the delayed deployment D is defined in two phases, as follows:

- *Phase* A. Form a desirable configuration with $S_0 = \frac{n}{\sqrt{k \log k}}$. To achieve this, release the agents one-by-one, starting from agent 1 to agent k, and perform exactly $(\lfloor p_i S_0 \rfloor - 1)^2$ moves with each agent, so that each agent i occupies position $\lfloor p_i S_0 \rfloor$ and all pointers on the path point to the left.
- *Phase* B. For successive $j = 0, 1, \ldots$, iterate the following procedure, until the path has been covered. Starting from an initial desirable configuration of some length S_j, form a new desirable configuration of length $S_{j+1} = S_j + \lceil k^4 a_1 a_k \rceil + 12k$ as follows:

  B1. Starting from the current desirable configuration, release all agents simultaneously for $\lceil 2k^4 a_k S_j \rceil$ rounds.

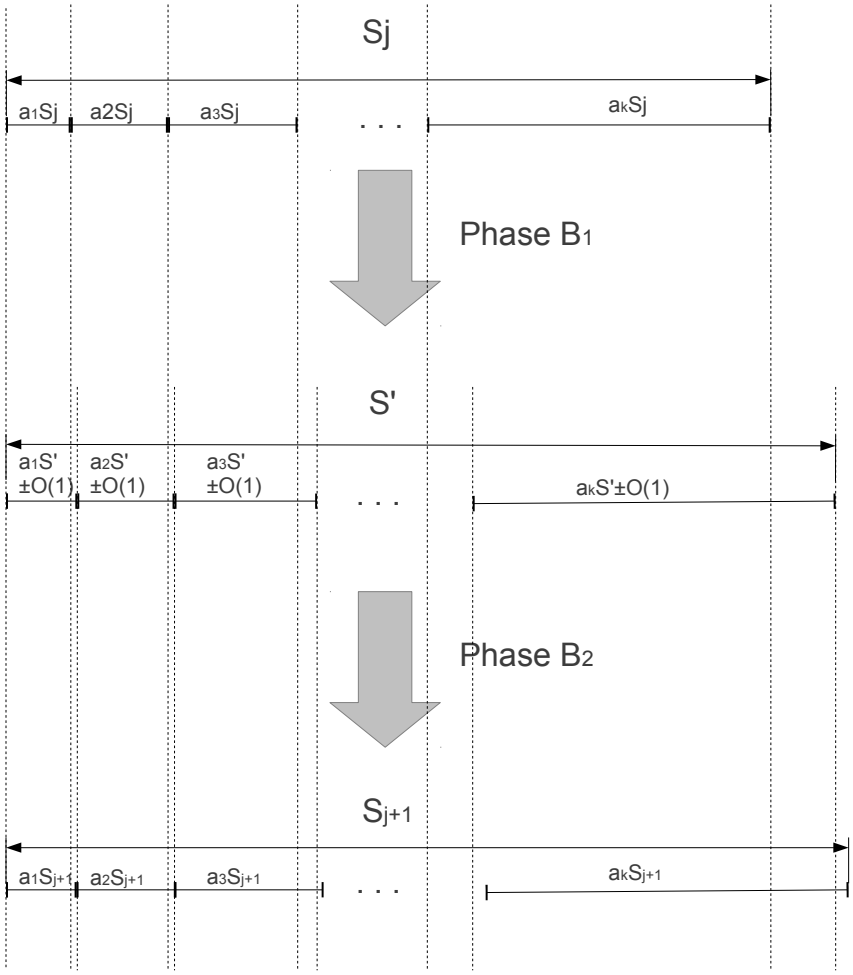

FIGURE 3.3: An iteration of Phase B of delayed deployment D (proof of Theorem 3.15)

B2. Adjust the positions of the agents, so as to reach the desirable configuration of length S_{j+1}. To achieve this, release the agents one-by-one, starting from agent 1 to agent k, allowing each agent i to move until it has reached position $\lfloor p_i S_{j+1} \rfloor$.

We denote by T the cover time of deployment D, by A, the total number of rounds of Phase A, by B_1, the total total number of rounds of Phase B1, and by B_2, the total number of rounds of Phase B2. We also remark that during Phase B1 none of the agents is delayed, hence, by Lemma 3.3 we have:

$$B_1 \leq C_{rr}^k(G) \leq T = A + B_1 + B_2.$$

We begin by bounding time A. The agents are released sequentially in Phase A. The number of rounds required for each agent to reach its position is less than $\frac{n^2}{k \log k}$. Thus, $A < \frac{n^2}{\log k}$.

We now proceed to Phase B (see Fig. 3.3 for an illustration). The size of the smallest domain in configuration S_0 is:

$$\left\lfloor \frac{n}{\sqrt{k \log k}} a_k \right\rfloor \geq \left\lfloor \frac{n}{\sqrt{k \log k}} \frac{1}{4(H_k+1)k} \right\rfloor \geq \left\lfloor \frac{k^{11}}{k^{3/2} \log k \, (4 \log k + 8)} \right\rfloor \geq k^9,$$

where the last inequality holds for $k \geq 10^6$. Consider now the j-th step of the phase, starting from length $S = S_j$, and the change of the configuration within part B1 of this step. The number of rounds used in part B1 of the step is $2a_k S k^4$. Let $|V_i|_j = \lfloor p_i S \rfloor - \lfloor p_{i+1} S \rfloor \geq a_i S - 1$ be the size of the domain of the i-th agent at the beginning of the j-th step, and let $|V_i|_j + g_i$ be its size after completion of part B1 of this step. In order to increase the size of its domain, the i-th agent needs to perform at least g_i traversals of its domain (such that during these traversals the size of this domain is at least $|V_i|_j$), where a traversal is understood as starting and ending at the right endpoint of the domain. These traversals require more than $a_i S g_i$ rounds, whereas the total duration of part B1 of the j-th step is $\lceil 2a_k S k^4 \rceil$, hence we obtain $g_i < 2k^4$. Since the total size of all domains is non-decreasing in time, it follows that $\sum_{i=1}^{k} g_i \geq 0$, and so:

$$-2k^5 \leq g_i < 2k^4.$$

We now proceed to refine this bound on g_i. Initially, the size of the i-th domain is between $a_i S - 1$ and $a_i S + 1$. Thus, for the i-th agent, the number of completed traversals c_i of its domain during the considered part B1 is:

$$\frac{2a_k S k^4}{a_i S + 1 + 2k^4} \leq c_i \leq \frac{2a_k S k^4 + 1}{a_i S - 1 - 2k^5}.$$

If the i-th node performed c_i complete traversals, then it reached each of the boundaries of its domain at lest c_i times and one boundary could be reached $c_i + 1$ times. Thus, considering the change in size of domain g_i during the traversals of agents i, $i-1$ and $i+1$, we have:

$$2c_i - c_{i-1} - c_{i+1} - 2 \leq g_i \leq 2c_i + 1 - c_{i-1} - c_{i+1}$$

and introducing the bounds on c_i, c_{i-1}, c_{i+1} to the

$$2a_k S k^4 \left(\frac{2}{a_i S + 1 + 2k^4} - \frac{1}{a_{i-1} S - 1 - 2k^5} - \frac{1}{a_{i+1} S - 1 - 2k^5} \right) - 2 \leq g_i$$

$$g_i \leq \left(2a_k S k^4 + 1\right) \left(\frac{2}{a_i S - 1 - 2k^5} - \frac{1}{a_{i-1} S + 1 + 2k^4} - \frac{1}{a_{i+1} S + 1 + 2k^4} \right) + 1.$$

$$2a_k S k^4 \left(\frac{2}{a_i S} \left(1 - \frac{2k^4+1}{a_i S + 1 + 2k^4}\right) - \frac{1}{a_{i-1} S} \left(1 + \frac{2k^5+1}{a_{i-1} S - 1 - 2k^5}\right) - \frac{1}{a_{i+1} S} \left(1 + \frac{2k^5+1}{a_{i+1} S - 1 - 2k^5}\right) \right) - 2 \leq g_i$$

$$g_i \leq 2a_k S k^4 \left(\frac{2}{a_i S} \left(1 + \frac{2k^5+1}{a_i S - 1 - 2k^5}\right) - \frac{1}{a_{i-1} S} \left(1 - \frac{2k^4+1}{a_{i-1} S + 1 + 2k^4}\right) - \frac{1}{a_{i+1} S} \left(1 - \frac{2k^4+1}{a_{i+1} S + 1 + 2k^4}\right) \right) + 2$$

We know that $a_i S \geq k^9$ and $a_k \leq a_i$

$$2a_k k^4 \left(\frac{2}{a_i}\left(1-\frac{2}{k^5}\right) - \frac{1}{a_{i-1}}\left(1+\frac{2}{k^4}\right) - \frac{1}{a_{i+1}}\left(1+\frac{2}{k^4}\right)\right) - 3 \leq g_i$$

$$g_i \leq 2a_k k^4 \left(\frac{2}{a_i}\left(1+\frac{2}{k^4}\right) - \frac{1}{a_{i-1}}\left(1-\frac{2}{k^5}\right) - \frac{1}{a_{i+1}}\left(1-\frac{2}{k^5}\right)\right) + 3$$

$$2a_k k^4 \left(\frac{2}{a_i} - \frac{1}{a_{i-1}} - \frac{1}{a_{i+1}}\right) - 11 - \frac{8}{k} \leq g_i \leq 2a_k k^4 \left(\frac{2}{a_i} - \frac{1}{a_{i-1}} - \frac{1}{a_{i+1}}\right) + 11 + \frac{8}{k}$$

$$2a_i a_k k^4 a_1 - 11 - \frac{8}{k} \leq g_i \leq 2a_i a_k k^4 a_1 + 11 + \frac{8}{k}$$

The above analysis shows that the position of the i-th agent after Phase B1 is upper-bounded by:

$$\lfloor p_i S \rfloor + \sum_{l=i}^{k} g_l < \lfloor p_i S \rfloor + \sum_{l=i}^{k} (a_l a_1 a_k k^4 + 11 + 8/k) \leq$$
$$\leq p_i S + p_i a_1 a_k k^4 + 11k + 8 \leq \lfloor p_i S_{j+1} \rfloor - k + 9 < \lfloor p_i S_{j+1} \rfloor.$$

and lower-bounded by:

$$\lfloor p_i S \rfloor + \sum_{l=i}^{k} g_l \geq \lfloor p_i S \rfloor + \sum_{l=i}^{k} (a_l a_1 a_k k^4 - 11 - 8/k) \geq p_i S + p_i a_1 a_k k^4 a_1 - 11k - 9 \geq$$
$$\geq \lfloor p_i S_{j+1} \rfloor - 23k - 9 > \lfloor p_i S_{j+1} \rfloor - 24k.$$

Now, consider the duration of the Phase B2. Each agent must adjust its position to the right, by a distance of at most $24k$. First, the right-most agent (agent 1) has to perform at most $24k$ traversals of its domain. As a result, the size of the domain of the penultimate agent (agent 2) can decrease by at most $24k$, hence it must perform at most $24k + 24k = 48k$ traversals to reach its position at the end of the step. In general, agent i has to perform at most $24ki \leq 24k^2$ traversals of its domain. The size of the i-th domain during Phase B2 is at most $a_i S + 2k^4 + 48k^2$. Thus, the duration of Phase B2 is bounded by:

$$\sum_{i=1}^{k} \left(a_i S + 2k^4 + 48k^2\right) 24k^2 < 24Sk^2 + 48k^7 + 1152k^5.$$

Observe that the duration of part B1 of the step was $2a_k S k^4 \geq \frac{2}{4k(H_k+1)} S k^4 > 24Sk^2$ for $k > 10^3$, because $a_k \geq \frac{1}{4k(H_k+1)}$ from Lemma 3.14. Thus, overall we have that the execution of B1 dominates the complexity of the algorithm, $B_1 \in \Omega(B_2)$ and $B_1 \in \Omega(A)$. It follows that $C_{rr}^{R[k]}(G) = \Theta(B_1)$. Now, in order to bound time B_1, observe that the j-th step of Phase B results in the increase of S_j, the number of already covered nodes, by $\Theta(k^4 a_1 a_k)$, which means that Phase B consists of $\Theta\left(\frac{n}{k^4 a_1 a_k}\right)$ steps. Since more than half of these steps are performed for $n/2 < S_j < n$, we obtain a tight bound on the

cover time $B_1 \in \Theta(\frac{n^2}{a_1})$. Noting that $a_1 = \Theta(\frac{1}{H_k})$ by Lemma 3.14, we eventually obtain $B_1 \in \Theta(\frac{n^2}{\log k})$. Thus, $C_{rr}^{R[k]}(G) \in \Theta(\frac{n^2}{\log k})$. □

We now show that the initialization considered above, with all agents starting from one node and all ports pointing to the left, is indeed asymptotically the worst possible. The proof of this theorem proceeds in two steps, first by considering agents starting from one node with an arbitrary placement of pointers on the ring, and then by extending this result to the general case through the application of delayed deployments.

Lemma 3.16. *In the case when all the agents are initially placed at the same node v, a group of k agents explores the ring of size n in time $O(\frac{n^2}{\log k})$ when $k < n^{1/11}$, regardless of the initial placement of pointers.*

Proof. We extend the proof of the upper bound from Theorem 3.15 to different initializations of pointers. We consider the case of the rotor-router deployment $R[k]$ on the n-node path with all agents initially positioned at the left endpoint of the path (but with arbitrary pointer initialization along the path). As in the proof of Theorem 3.15, we consider a delayed deployment with similarly defined Phases A and B, using the same set of desirable configurations of length S_j. Note that in a desirable configuration, all the pointers along the path point to the left for all nodes which have already been visited by an agent at least once. In Phase A, agents are released one-by-one, until the i-th agent reaches position $\lfloor p_i S_0 \rfloor$, after which the agent is stopped (this may happen after a smaller number of steps than in the proof of Theorem 3.15). In the j-th step of Phase B, the only difference concerns the definition of part B1, where we add the condition that, upon reaching position $\lfloor p_i S_{j+1} \rfloor$ for the first time, the i-th agent stops and waits for the other agents to complete part B1 of the step. By induction, one can show that for $i > 1$, agent i will only stop moving in part B1 after agent $i-1$ has stopped moving, and consequently, it may never happen that a moving agent meets a stationary agent. The analysis of the time spent within parts A, B1 and B2 is performed as before, and we obtain $C_{rr}^{R[k]}(G) = A + B_1 + B_2 = O(\frac{n^2}{\log k})$.

The analysis on the ring proceeds by a modification of the argument for a path, treating the ring as two sub-paths connected at the common node 1. In Phase B, the deployments on both sub-paths are synchronized so that the agents a_k of the respective deployments arrive at node 1 simultaneously. If agent a_k of one of the sub-paths, say the left one, arrives before the agent a_k of the right sub-path, then all the agents of the left sub-path are stopped at their current locations until the other agent a_k arrives at node 1. (Note that the two sub-paths do not have to be performing the same step j of Phase B at the same time.) This transformation of the deployment on the path does not affect asymptotic analysis, hence the cover time of the deployment on the ring is also $O(\frac{n^2}{\log k})$. □

Theorem 3.17. *For any initialization of the k-agent rotor-router system on the ring, the cover time is $O(\frac{n^2}{\log k})$, for $k < n^{1/11}$.*

Proof. Let $R[k]$ be a deployment of the rotor-router on the ring. Fix a subset $P \subset V$ of $P = k^{2/3}$ points on the ring which are evenly spaced, i.e., $G[V \setminus P]$ is a set of disjoint paths of length at most $n/k^{2/3}$. Consider a delayed deployment of $R[k]$, which begins with a Phase in which the agents of $R[k]$ are activated and moved one by one, stopping each agent as soon as it has reached a node from P. Since the cover time of a path of length $O(n/k^{2/3})$ for a single agent is $O(n^2/k^{4/3})$, the duration of this Phase is at most $O(n^2/k^{1/3})$. After this initial phase, by the pigeon-hole principle, there must exist a node $v \in P$ which contains $k' \geq k^{1/3}$ agents. We now continue the delayed deployment by releasing $k'' = \min\{k^{1/3}, n^{1/11}\}$ agents which are located at v, and permanently stopping (removing) all other agents. By Theorem 3.15, the path will be covered by the delayed deployment within $O\left(\frac{n^2}{\log k''}\right)$ rounds. By summing the duration of the two phases and using the slow-down lemma, we obtain the claim: $C_{rr}^{R[k]}(G) \in O\left(\frac{n^2}{k^{1/3}} + \frac{n^2}{\log k''}\right) = O(\frac{n^2}{\log k})$, for $k < n^{1/11}$. □

The following three lemmas provide a partial characterization of the changes of size of domains during the runtime of a deployment. We first define the borders and interiors of the domains. If we consider the dynamics of the set of domains $V_i(t)$ in time then we can observe that an agent i making a cycle in his domain V_i will always capture one node of both the neighboring domains $i-1$ and $i+1$. Thus some nodes will frequently change their membership in domains. The goal of defining borders and interiors is to obtain a more "stable" process which will allow us to analyze its behaviour in time. Borders are defined in a fixed moment in time T_{bor} and for any time moment $t > T_{bor}$ are defined recursively based on the positions of borders in step $t-1$ and positions of agents in step t.

Definition 3.18. (1) For time T_{bor} the *border* $B_{a_i,a_{a+1}}(T_{bor})$ between agents a_i and a_{i+1} is defined as a set of two nodes: such a node from $V_i(T_{bor})$ that has a neighbor in $V_{i+1}(T_{bor})$ (since we work on ring and we have more than 2 agents then there can be only one such node) and a node from $V_{i+1}(T_{bor})$ that has a neighbor in $V_i(T_{bor})$.

(2) For time $t > T_{bor}$

(i) If $v_i(t) \in V_{i+1}(t-1) \setminus B_{a_i,a_{i+1}}(t-1)$ (i.e. a_i captured a node v belonging in time $t-1$ to the domain of a_{i+1} and node v was not an element of the border $B_{a_i,a_{i+1}}(t-1)$) then the border $B_{a_i,a_{i+1}}(t)$ moves. It is reset according to the rule (1).

(ii) If $v_{i+1}(t) \in V_i(t-1) \setminus B_{a_i,a_{i+1}}(t-1)$ then the border $B_{a_i,a_{i+1}}(t)$ moves in the opposite direction. It is reset according to the rule (1).

(iii) If none of (i), (ii) happened in time t then the border does not move $B_{a_i,a_{i+1}}(t) = B_{a_i,a_{i+1}}(t-1)$.

Borders are defined between all pairs of consecutive domains. We have borders $B_{a_1,a_2}(t), B_{a_2,a_3}(t), \ldots B_{a_{k-1},a_k}(t)$. If time T_{exp} is a moment of exploration of the ring

(i.e. $V_0(T_{exp}) = \emptyset$ and $V_0(T_{exp} - 1) \neq \emptyset$) then the border B_{a_k,a_1} between agent a_1 and agent a_k is defined as in Definition 3.18(1). Thus we have $B_{a_k,a_1}(t) = \emptyset$ for $t = T_{bor}, T_{bor} + 1, \ldots, T_{exp} - 1$ and in step T_{exp} border B_{a_k,a_1} is defined and from that moment of time on it can be moved according to Definition 3.18. Denote by $B(t) = \bigcup_{i=1}^{k-1} B_{a_i,a_{i+1}}(t) \cup B_{a_k,a_1}(t)$ the set of all border nodes in step t.

We will say that agent visits the border $B_{a_i,a_{i+1}}(t)$ if it visits at least one of the nodes of the border. We will say that if the rule $(2)(i)$ from the definition 3.18 is applied then the border between agents a_i and a_{i+1} is moved by agent a_i. Symmetrically if the rule $(2)(ii)$ is applied then we say that the border is moved by agent a_{i+1}. In both cases the border is moved by two nodes. Observe that border $B_{a_i,a_{i+1}}$ can be moved by a_i only if the agent visits it twice and between these two visits there was no visit by a_{i+1}. This observation will be used in the following analysis of the evolution of the domains.

Definition 3.19. Define the *interior* $I_{a_i}(t)$ of the domain of agent a_i as the domain without border nodes $I_{a_i}(t) := V_i(t) \setminus B(t)$ for $i = 1, 2, \ldots k$.

Every domain V_i of an agent a_i consists of an interior I_{a_i} and potentially the left border B_{a_{i-1},a_i} and the right border $B_{a_i,a_{i+1}}$. Thus, the difference between the size of the domain and the size of the interior is at most 4.

Lemma 3.20. *Assume that unexplored part of the ring V_0 has negatively initialized pointers (i.e. the first agent entering from a node u to a node $v \in V_0$ will be sent back to u). If $k \geq 6$ and at time T_{bor} the interior of every domain has size at least $22k$, then for any t such that $T_{bor} \leq t < T_{exp}$:*

(1) if a, b, c are any three agents with consecutive domains (i.e. $(a, b, c) = (a_i, a_{i+1}, a_{i+2})$ or $(a, b, c) = (a_{i+2}, a_{i+1}, a_i)$ for any $1 \leq i \leq k - 2$), if $|I_a(t)| - 7 > |I_b(t)|$ and $|I_b(t)| \leq 2|I_c(t)|$, then in step $t + 1$ the border $B_{a,b}$ will not be moved by agent a,

(2) if $(a, b) = (a_{k-1}, a_k)$ or $(a, b) = (a_2, a_1)$ if $|I_a(t)| - 7 > |I_b(t)|$, then in step $t + 1$ the border $B_{a,b}$ will not be moved by agent a,

(3) the size of the interior of any domain is at least $11k$.

Proof. We will prove this lemma by induction on time. First, take time T_{bor}. In time $T_{bor} + 1$ no border can move, because T_{bor} is the time of initialization of the borders and an agent has to visit the border twice to move it.

We will prove three implications to prove the lemma. Fix any $t > T_{bor}$. Assume that conditions $(1), (2)$ are true for $T_{bor}, T_{bor} + 1, \ldots, t$ and initially every interior has size at least $22k$. We want to prove that condition (3) is true for $t + 1$. We define for any $t^* \in [T_{bor}, t]$ a function $i_{min}(t^*) = \min_{\alpha \in \mathcal{A}}\{|I_\alpha(t^*)|\}$ and $j_\alpha(t^*) = \min\{|I_\alpha(t^*)|, 22k\}$ for all $\alpha \in \mathcal{A}$. We want to show, that $i_{min}(t + 1) \geq 11k$.

Take any step $t^* \in [T_{bor}, t]$. By the inductive claim $i_{min}(t^*) \geq 11k$ thus if for some agents a, b with adjacent domains $j_a(t^*) - j_b(t^*) \geq 8$, then $|I_b(t^*)| < 22k - 8 \leq 2|I_c(t^*)|$, because by the inductive claim $|I_c(t^*)| \geq 11k$. Thus in step $t^* + 1$ the border between a

and b cannot move in the direction of the smaller interior. If $|j_i(t^*) - j_{i+1}(t^*)| = 7$, then the border can still move in the direction of the smaller interior thus the difference can increase up to 11. Since initially for every $\alpha \in \mathcal{A}$, $|I_\alpha(T_{bor})| \geq 22k$, then $j_\alpha(T_{bor}) = 22k$. Consider the configuration that yields the minimum possible value of function j_α for some agent α. The configuration is $j_{a_1}(t^*) = 22k, j_{a_2}(t^*) = 22k - 11, \ldots j_{a_k}(t^*) = 11k + 11$. It is true for any $t^* \in [T_{bor}, t+1]$. Thus the minimum size of the interior of any domain in step $t+1$ is at least $11k$.

Now we will prove the second implication. Take any time step $t \geq T_{bor}$. We want to prove, that the condition (3) for $T_{bor}, T_{bor}+1, \ldots, t$ implies conditions (1) and (2) for t. Assume, by contradiction, that agent a moves the border between B_i in step $t+1$. So, in step t, agent a is located at the extremal point of the border of domains a and b, having completed a cyclic exploration of its domain. Let us denote the mentioned extremal point of border by v_b. Let $t^* < t$ be time step, when a previously visited v_b. If such t^* does not exist, then either a is making first cycle after domains were defined or node v_b was not visited by a in previous cycle. In both cases a cannot move the border in step $t+1$. Thus such t^* exists. Note, that since in time $t+1$ agent a moves the border then b had not visited v_b in time interval $[t^*+1, t]$. It is however possible, that in time t^* both a and b were located in v_b.

Consider what is the minimum time $t_b > t^*$ for agent b to arrive at the border $B_{a_i,a_{i+1}}$. Agent b has to get to the border with c (in time $|I_b(t^*)|$ or less) then it can move the border $B_{a_{i+1},a_{i+2}}$ at most once which takes time 5 and then again $|I_b(t^*)|$ to arrive at node v_b. Thus $t_b \leq t^* + 2|I_b(t^*)| + 6$. Since b had not arrived at v_b until time t then $t_b > t$ and $t - t^* \leq 2|I_b(t^*)| + 5$.

On the other hand agent a made a full cycle in the interval $[t^*, t]$. In this cycle a had to visit all nodes from $I_a(t)$ twice and both nodes from border $B_{a_i,a_{i+1}}$ twice and at least one node from the other border once. If $i = 1$ then there is no other border but then case agent a captures at least one previously unexplored node. Thus $t - t^* \geq 2|I_a(t)| + 4$.

We obtained lower and upper bound on $t - t^*$ but in the upper bound we have size of the I_b in time t^*. In the interval $[t^*, t]$ agent c could capture some nodes of the interior of b. In the following we want to bound the number of nodes that could be captured by c in the interval $[t^*, t]$. We denote $i_c = \min_{s \in [t^*,t]}\{|I_c(s)|\}$ which is the minimum size of interior of domain of agent c in time interval $[t^*, t]$. We will consider two cases. First assume, that $i_c > |I_b(t^*)|/3$, thus $t - t^* \leq 2|I_b(t^*)| + 5 \leq 6i_c + 5$ and c can make at most 3 complete cycles of its domain. Thus b can lose at most 6 nodes to c in time interval $[t^*, t]$, thus $|I_b(t^*)| - 6 \leq |I_b(t)|$. We have,

$$t - t^* \leq 2|I_b(t^*)| + 5 \leq 2|I_b(t)| + 17 \leq 2|I_a(t)| + 3 < t - t^*,$$

which leads to a contradiction.

Now consider case, when $i_c \leq |I_b(t^*)|/3$. This means that agent c during interval $[t^*, t]$ increased size of his interior from at most $|I_b(t^*)|/3$ to at least $|I_b(t)|/2$. To increase size of the interior from i_c to $i_c + 2$ agent c has to make at least one full

cycle of his interior and visit border twice thus $2i_c + 4$ steps are needed. Similarly to increase from i_c to $i_c + 2\delta$, we need at least $2\delta(i_c + \delta + 1)$. Thus to increase from i_c to $|I_c(t)|$ at least $2\left\lfloor\frac{|I_c(t)|-i_c}{2}\right\rfloor\left(i_c + \left\lfloor\frac{|I_c(t)|-i_c}{2}\right\rfloor + 1\right)$ steps are needed. Thus $t - t^* \geq 2\left\lfloor\frac{|I_c(t)|-i_c}{2}\right\rfloor\left(i_c + \left\lfloor\frac{|I_c(t)|-i_c}{2}\right\rfloor + 1\right)$. On the other hand since $t - t^* \leq 2|I_b(t^*)| + 5$. We also assume in the condition (1) that $|I_b(t)| \leq 2|I_c(t)|$ We have

$$\begin{aligned}
2|I_b(t^*)| + 5 &\geq t - t^* \\
&\geq 2\left\lfloor\frac{|I_c(t)| - i_c}{2}\right\rfloor\left(i_c + \left\lfloor\frac{|I_c(t)| - i_c}{2}\right\rfloor + 1\right) \\
&\geq 2\left(\frac{|I_c(t)| - i_c}{2} - 1\right)\left(i_c + \frac{|I_c(t)| - i_c}{2}\right) \\
&= |I_c(t)|(|I_c(t)| - 2)/2 - i_c^2/2 - i_c \\
&\geq 16/33|I_c(t)|^2 - i_c^2/2 - i_c \\
&\geq 4/33|I_b(t)|^2 - |I_b(t^*)|^2/18 - |I_b(t^*)|/3
\end{aligned}$$

Where we used the fact that $|I_c(t)| \geq 11k \geq 66$ thus $2 \leq 1/33|I_c(t)|$. Thus we have

$$|I_b(t^*)|^2/6 + 7|I_b(t^*)| + 15 \geq 4/11|I_b(t)|^2 \tag{3.19}$$

Since a made one cycle in time $[t^*, t]$ then all nodes that b lost during this interval were taken by agent c. Thus agent c had to make at least $2i_c(|I_b(t^*)| - |I_b(t)|)$ steps. Since, by the inductive assumption $i_c \geq 11k$:

$$2|I_b(t^*)| + 5 \geq t - t^* \geq 2i_c(|I_b(t^*)| - |I_b(t)|) \geq 132(|I_b(t^*)| - |I_b(t)|),$$

Which gives us

$$132|I_b(t)| + 5 \geq 134|I_b(t^*)|. \tag{3.20}$$

By combining inequalities (3.19) and (3.20) and the fact that $|I_b(t^*)| \geq 66$ we obtain a contradiction. Thus the second implication is also true. The last implication, that the condition (3) for $T_{bor}, T_{bor} + 1, \ldots, t$ implies condition (2) for t can be proven similarly as the second implication. Since the unexplored region has negatively initialized pointers, it can be seen as a domain of an agent c who never moves any border of its domain. □

Now we will formulate an analogue of Lemma 3.20 for the case of $t \geq T_{exp}$.

Lemma 3.21. *If $k \geq 6$ and at time T_{bor} the interior of every domain has size at least $22k$, then for any $t \geq T_{exp}$:*

(1) if a, b, c are any three agents with consecutive domains and if $|I_a(t)| - 7 > |I_b(t)|$ and $|I_b(t)| \leq 2|I_c(t)|$, then in step $t + 1$ the border $B_{a,b}$ will not be moved by agent a,

(2) the size of the interior of any domain is at least $11k$.

Proof. Analogical to the proof of Lemma 3.20 (1) and (3). □

We now show two auxiliary lemmas which allow us to conclude that the sizes of all domains will eventually even out in time.

Lemma 3.22. *Assume that the unexplored part of the ring V_0 has negatively initialized pointers or is empty. Let a and b be two agents with consecutive domains (i.e. $(a,b) = (a_i, a_{i+1})$ or $(a,b) = (a_{i+1}, a_i)$). If $k \geq 6$ and initially the interior of every domain has size at least $22k$ and $|I_a(t)| > 1.1|I_b(t)|$, then for any $t \geq T_{bor}$, in step $t+1$ the border $B_{a,b}$ will not be moved by agent a.*

Proof. Assume by contradiction, that agent a moves the borders in step $t+1$. Let t^* be the last time step, when agent b visited its other border. Using the same argument as in the proof of Lemma 3.20, we have $2|I_a(t)| + 4 \leq t - t^* \leq 2|I_b(t^*)| + 5$. It is possible, that $|I_b(t^*)| > |I_b(t)|$ if b lost some nodes during the time interval $[t^*, t]$. But this number of nodes is limited since the size of every domain is at least $11k$ (by Lemmas 3.20,3.21). Thus agent b loses at most 2 nodes once every $22k$ time steps. Thus during $2|I_b(t^*)| + 5$ time steps, b lost at most $\frac{2|I_b(t^*)|+5}{11k} + 2$ nodes. Thus

$$|I_b(t)| \geq |I_b(t^*)| \left(1 - \frac{2}{11k}\right) - \frac{5}{11k} - 2,$$

$$|I_b(t^*)| \leq \left(|I_b(t)| + \frac{5}{11k} + 2\right) \left(1 + \frac{2}{11k-2}\right).$$

Since $k \geq 6$, and $|I_b(t)| \geq 11k \geq 66$, then $|I_b(t^*)| \leq |I_b(t)| \left(1 + \frac{5}{11k|I_b(t)|} + \frac{2}{|I_b(t)|}\right) \left(1 + \frac{2}{11k-2}\right) < 1.08|I_b(t)|$. Thus

$$t - t^* \leq 2|I_b(t^*)| + 5 < 2.16|I_b(t)| + 5 < 2.18|I_b(t)| + 4 < 2|I_a(t)| + 4 \leq t - t^*.$$

And we obtain a contradiction. □

Lemma 3.23. *Assume that unexplored part of the ring V_0 has negatively initialized pointers or is empty. If $|I_a(t^*)| - 4 > |I_b(t^*)|$ holds for $2n^2$ consecutive time steps $t^* = t, t+1, t+2, \ldots, t+2n^2-1$ for some $t \geq T_{bor}$, then border between a and b will be moved by b once in the interval $[t, t+2n^2-1]$.*

Proof. Any full cycle of agent a takes at least $2|I_a(t^*)|$. Any full cycle of b takes at most $2|I_b(t^*)| + 6$ (visiting the whole interior twice and both borders) time steps. Thus if $2|I_a(t^*)| > 2|I_b(t^*)| + 8$, then the cycle of b is shorter by at least two steps. Thus after a sufficiently large number of time steps (after time at most $2n^2$), b will visit the border twice in some time interval $[t_1, t_2]$ and a will not visit the border in this time interval. Thus, b will move the border towards a and gain two nodes. □

From our considerations, we obtain the lemma which will prove crucial in characterizing the limit behavior of the rotor-router on the ring.

Lemma 3.24 (**agent domains**). *If at some time step t every domain has size at least $22k+2$ and $k \geq 6$, then after a sufficiently large number of steps the interiors of adjacent domains will differ by at most 7.*

Proof. If domains have sizes at least $22k + 2$ in step $T_{bor} = t$, then we can define interiors and borders so that every interior has size at least $22k$. We already know from Lemmas 3.20 3.21 that if initially every interior has size at least $22k$, then during the deployment every domain will have size at least $11k$. If a and b are neighbors and at time t^* $|I_a(t^*)| \geq 2|I_b(t^*)|$, then we will say that there is a *significant difference* between a and b. If there is a significant difference between two adjacent domains of a and b, then by Lemma 3.22, the border will never move towards the smaller domain and by Lemma 3.23, the border will eventually move towards the bigger domain. Thus, significant differences will eventually disappear. Now, if there is no significant difference between any interiors of neighboring domains, then by Lemmas 3.20 3.21, the border can move in the wrong direction (towards the smaller domain) only if the difference is at most 7. On the other hand, by Lemma 3.23, if the difference is at least 4 for a sufficiently large number of consecutive time steps, then the border will move towards the bigger domain. Thus, if the difference between the sizes of two adjacent interiors is at least 8, then the border cannot move in the wrong direction and will eventually move in the correct direction. Thus, finally if the sizes of each domain are initially at least $22k + 2$, then after some number of time steps, the interiors of adjacent domains will differ by at most 7. □

3.5.1 Best-case initial placement

We start by proposing the initialization with agents equally spaced along the path as a candidate for (asymptotically) best-case initial placement with $O\left((\frac{n}{k})^2\right)$ cover time. The proof is straightforward in the case if we assume that the adversary initially directs all pointers towards the nearest agent, so as to block it. However, the adversary may apply a different strategy, and there do indeed exist port arrangements which deflect agents from some section of the ring, leading to a larger value of cover time. In our proof we show such actions of the adversary do not affect the asymptotics of the cover time.

Theorem 3.25. *Consider an initialization of the rotor-router system on the ring with agents starting on a set of points $P = \{p_1, p_2, \ldots, p_k\}$, such that $G[V \setminus P]$ is a set of paths of length at most n/k. Then, the system covers all of the nodes of the ring in time $O\left((\frac{n}{k})^2\right)$, regardless of the initial pointer arrangement.*

Proof. W.l.o.g, let $1 \leq p_1 < p_2 < \ldots < p_k \leq n$. Given a fixed initial pointer arrangement, let $x \in [1, n]$ be the node which is visited last by the rotor-router. To prove the claim, by the slow-down lemma, it suffices to construct a delayed deployment D of the rotor-router such that point x is visited by some agent within $O\left((\frac{n}{k})^2\right)$ rounds. We define deployment D as follows. Initially, we release all agents simultaneously, so that each agent moves left while the pointer of its current node points to the left, and stops as soon as it encounters a node whose pointer points to the right. Let q_i denote the position of the agent starting

from p_i after this phase is complete; we have $p_i - n/k \leq q_i \leq p_i$, hence the duration of this phase is at most n/k. We also have $|q_{i+1} - q_i| \leq 2\lceil n/k \rceil$. After this initialization phase, the deployment proceeds in steps of duration $4\lceil n/k \rceil$. The deployment is defined so that at the start of each step, agent i is located at point q_i. We describe the deployment through the following procedure, performed simultaneously by each agent i. The agent moves (to the right), stopping when it has either reached point q_{i+1}, or a node whose pointer points to the left. It then waits until the end of the $2\lceil n/k \rceil$-th round of the step to synchronize with other agents, and then returns to node q_i, where it waits until the end of the step.

We observe that in each step such that agent i does not reach q_{i+1}, it reaches a node on the path $[q_i, q_{i+1}]$ which has not previously been visited by any agent. Suppose that i is such that $q_i < x < q_{i+1}$. It follows that node x will be visited by agent i within $|q_{i+1} - q_i| \leq 2\lceil n/k \rceil$ steps. Since the duration of each step is $4\lceil n/k \rceil$, the second phase of the delayed deployment takes at most $8\lceil n/k \rceil^2$ round. Overall, point x is covered within $O\left((\frac{n}{k})^2\right)$ rounds from the start of the process, and the claim follows. □

To prove that the equally-spaced initialization is the best possible, we provide a general case lower-bound of $\Omega\left((\frac{n}{k})^2\right)$ on cover time for all initializations. To do this, we introduce an auxiliary notion of a *good vertex* for an initialization of the rotor-router. Such vertices are shown to always exist (in fact, to be in the majority in the vertex set) and take a long time to cover, regardless of the initial placement of agents.

Definition 3.26. For any placement of the k agents let $S = \{s_1, s_2, \ldots, s_k\}$ be the k not necessarily distinct starting vertices. We will consider the subset of *good vertices* of the cycle, defined as all nodes v which satisfy the following two constraints:

1. For all $1 \leq r \leq k$, $\left|\left[v, v + r\frac{n}{10k}\right] \cap S\right| \leq r$.
2. For all $1 \leq r \leq k$, $\left|\left[v, v - r\frac{n}{10k}\right] \cap S\right| \leq r$.

The following lemma concerning the relation between good vertices and the starting positions of the agents, and proves useful in the analysis of the k-agent rotor-router, as well as the k-agent random walk.

Lemma 3.27. *For any initial placement $S = \{s_1, s_2, \ldots, s_k\}$ of the k agents, there are at least $0.8n - o(n)$ good vertices.*

Proof. Let V_1 and V_2 be the sets of vertices which satisfy constraints 1 and 2 above, respectively. We first show that $|V_1| \geq 0.9n - o(n)$. Consider an algorithm which starts from vertex 0 and scans the cycle in the increasing order of vertex numbers, as follows:

$v \leftarrow 0$
$B \leftarrow \emptyset$
while $(v < n - (n/10k))$ **do**
  if $v \notin V_1$ **then**
    Let r be the smallest positive integer such that $|[v, v + r(n/k)) \cap S| > r$
    $B \leftarrow B \cup [v, v + r(n/10k))]$
    $v \leftarrow v + r(n/k)$
  else $v \leftarrow v + 1$
  end if
end while

By the construction of set B, each new interval of the form $[v, v + r(n/10k))$, of length $r(n/10k)$ which is added to it, contains more than r elements of set S. Consequently: $|B \cap S| > 10k|B|/n$, so $|B| < 0.1n|S|/k = 0.1n$. On the other hand, we observe that for $v < n - (n/10k)$, $v \notin B \implies v \in V_1$, and so $|V_1| \geq n - o(n) - |B| \geq 0.9n - o(n)$. By a similar argument, we show that $|V_2| \geq 0.9n - o(n)$. From here, we obtain the sought bound on the number of good vertices: $|V_1 \cap V_2| \geq 0.8n - o(n)$. □

Theorem 3.28. *If $n \geq 440k^2 + 40$ and $k \geq 6$, then for any set of initial locations of k agents, there exists an initial arrangement of pointers on the ring such that the cover time of the rotor-router system is $\Omega\left((\frac{n}{k})^2\right)$.*

Proof. Without affecting the asymptotic claim, we assume $k \geq 5$. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the k not necessarily distinct starting vertices. Let r_i be the number of vertices initially between s_i and s_{i+1}, and r_k be the number of vertices between s_k and s_1. Obviously $\sum_i r_i \geq n - k$. Thus $\sum_{\{i:r_i \geq \frac{n-k}{2k}\}} r_i \geq \frac{n-k}{2}$. If we take two middle quarters from each interval of length at least $\frac{n-k}{2k}$ then totally we will obtain at least $\frac{n-k}{4}$ nodes. Thus at least $n/4 - o(n)$ nodes are at distance at least $\frac{n-k}{8k}$ to the closest agent. If $n \geq 9k$ then $\frac{n-k}{8k} \geq \frac{n}{9k}$. Thus from Lemma 3.27 there are at least $0.05n - o(n)$ good nodes at distance at least $\frac{n}{9k}$ to the closest starting point of an agent. For sufficiently large n such node will exist. We will call this node v. Now we will use Lemma 3.1 and construct a delayed deployment $D1$. We will block all but one or two agents to ensure that each agent will have a domain of size at least $\frac{n}{20k}$ and at least $\frac{n}{10k}$ nodes will not be explored. We initiate all pointers negatively – in each node the pointer points away from the closest agent. We will describe the procedure in one direction. In the other direction procedure will be the same. Firstly we release the closet agent at the left of v until it reaches the node at distance $\frac{n}{20k}$ from v. Then we block the agent. Since the closest node to v is at distance at least $\frac{n}{9k}$ then after this procedure in interval $[v + \frac{n}{20k}, v + \frac{n}{10k}]$ there will be only one agent. Then we take the next closest agent at the left of v and release it until it reaches node $v + \frac{n}{10k}$. Again since v is a good node there will be only one agent in interval $[v + \frac{n}{10k}, v + \frac{n}{5k}]$. Then for i-th closest agent at the left of v for $i \geq 2$ we release it until it reaches node $v + (i-1)\frac{n}{10k}$. It is possible, that the agent will go to the other side of the ring. Then we block it at the node $v + \frac{n}{2}$ and continue procedure. We do the same

procedure to the left and right from v. We end up with some agents at node $v + \frac{n}{2}$. We release them one-by-one. Assume that such agent a went to the left from v. We block him, when he is at distance $\frac{n}{10k}$ from the last agent placed to the left of v. Now each agent has a domain of size at least $\frac{n}{20k}$. Now we release all agents simultaneously. By Lemmas 3.20 and 3.21 size of any domain will not drop below $\Omega(\frac{n}{20k})$. Assumptions of Lemmas 3.20 and 3.21 are satisfied, because $\frac{n}{20k} \geq 22k+2$ and the pointers are initialized negatively. We also have a group of $\frac{n}{20k}$ not explored nodes. Since this group will be explored by agents having domains of sizes at least $\Omega\left(\frac{n}{20k}\right)$ it will take at least $\Omega\left(\frac{n^2}{400k^2}\right)$ time steps. Number of steps in $D1$ when all agents are released simultaneously is $\Omega\left(\frac{n^2}{k^2}\right)$, thus from Lemma 3.1 the cover time of not delayed k agents in the rotor-router model in this case will also be $\Omega\left(\frac{n^2}{k^2}\right)$. □

3.5.2 Comparison with the Random Walk

The question of the cover time of random walks starting from a worst-case initial placement has already been resolved in the literature. On the one hand, it is known that the speedup of cover time for a k-agent random walk with respect to the single agent case is $\Omega(\log k)$ for any graph whose cover time is asymptotically equal to the maximum hitting time [7], regardless of the initial placement of agents. Since this is clearly the case for the ring [5], we have that the cover time of the k-agent random walk is $O(n^2/\log k)$. On the other hand, the adversary may choose to place all agents at one node of the ring. Such an all-one-one initialization has a cover time of precisely $\Theta(n^2/\log k)$ [7]. Thus, the cover time for k random walks on the ring with worst-case initialization is $\Theta(n^2/\log k)$. In order to give a complete picture of the relation between multi-agent rotor-router and random walk we would like to compare the cover time in the best-case initial placement of agents. The remaining part of this section will focus on analyzing the best-case cover time for k random walks on a ring.

To establish an upper bound for the best-case scenario, we consider k random walks with initial positions given with equal spacing, i.e., with offsets $0, n/k, 2(n/k), \ldots, (k-1)(n/k)$ relative to some node. (For simplicity, we assume here that k divides n.) The following lemma implies that in this case the cover time is $O((n/k)^2 \log^2 k)$.

Lemma 3.29. *Let $\alpha \geq 20$, $k \geq 2$ and let $t := \alpha^2 (n/k)^2 \log^2 k$. Then, with probability at least $1 - k^{1-\alpha/20}$, k random walks starting from initial positions with equal spacing cover all the vertices of the ring within t steps.*

Proof. Recall that $t = \alpha^2 \cdot (n/k)^2 \cdot \log^2(k)$. Since the maximum hitting time of a single random walk on a path with $\frac{1}{5}\sqrt{t}+1$ nodes is at most $\frac{1}{25}t$ (cf. [123]), we conclude from Markov's inequality that a single random walks on the ring with n vertices visits a vertex which is at least $\frac{1}{5} \cdot \sqrt{t}$ to the right of its starting vertex within t steps with probability at least $1/4$. Note that for any vertex $u \in V = \{0, \ldots, n-1\}$, there are at least $x-1$ random walks with distance between (n/k) and at most $x \cdot (n/k)$ to u. Putting $x = \frac{1}{10} \cdot \sqrt{t}/(n/k)$

we obtain the following upper bound for the event that u will *not* be covered:

$$\left(1-\frac{1}{4}\right)^{\frac{1}{10}\cdot\sqrt{t}/(n/k)-1} = \left(1-\frac{1}{4}\right)^{\frac{\alpha}{10}\cdot\log(k)-1} \leq k^{-\alpha/20}.$$

Now note that if for any vertex u of the set $S := \{0, n/k, 2(n/k), \ldots, (k-1)\,(n/k)\}$ there is a random walk that is initially placed to the left of u with distance at most $\frac{1}{10}\cdot\sqrt{t}/(n/k)$ and which traverses at least $\frac{1}{5}\cdot\sqrt{t}$ steps to the right within the first t steps, then all vertices of the ring are covered after t steps. Hence by taking the union bound over the set S we conclude that all vertices of the ring are covered with probability at least

$$1 - k\cdot k^{-\alpha/20} = 1 - k^{1-\alpha/20}.$$

□

We now prove a corresponding lower bound on the cover time in the best-case scenario, showing that the position with equal spacing is asymptotically the best possible. We first prove an auxiliary result which relies on the notion of good vertices introduced in the previous section.

Lemma 3.30. *Let $t = 10^{-4}\cdot(n/k)^2\cdot\log^2(k)$, $k = \omega(1)$, and let u be any good vertex at distance at least $\frac{n}{10k}$ from the starting points of all random walks. Then, with probability at least $k^{-1/2}$, u is not covered after t steps by any of the k random walks.*

Proof. Consider first a random walk with distance $(n/k)/10 \leq d \leq 4\cdot\sqrt{t}$ to u. The probability that the random walk reaches a point with distance at least $4\cdot\sqrt{t}$ to u without visiting u before is equal to

$$\frac{d}{4\cdot\sqrt{t}}.$$

Once the random walk has distance $4\cdot\sqrt{t}$ to u, the probability that it does not visit u within t steps is at least $1/2$. Combining these insights, we obtain that a random walk with distance $d \leq 4\cdot\sqrt{t}$ does not visit the vertex u within t steps with probability at least

$$\frac{d}{4\cdot\sqrt{t}}\cdot\frac{1}{2}.$$

Consider now all random walks with distance less than $4\cdot\sqrt{t}$. The number of these random walks is $4\cdot\sqrt{t}/(n/k) = (1/25)\log k$. The probability that none of these random

walks covers u is at least

$$\begin{aligned}
\prod_{j=0}^{(1/25)\log k-1} \frac{\frac{n}{10k}+j\cdot\frac{n}{10k}}{4\cdot\sqrt{t}}\cdot\frac{1}{2} &\geq 4^{-(1/25)\log k}\prod_{j=1}^{(1/25)\log k}\frac{j\cdot\frac{n}{10k}}{\sqrt{t}}\\
&\geq 4^{-(1/25)\log k}\cdot\prod_{j=1}^{(1/25)\log k}\frac{j}{(1/10)\log k}\\
&\geq 10^{-(1/25)\log k}\cdot\prod_{j=1}^{(1/25)\log k}\frac{j}{(1/25)\log k}\\
&\geq 10^{-(1/25)\log k}\cdot\frac{((1/25)\log k)!}{((1/25)\log k)^{(1/25)\log k}}\\
&\geq 10^{-(1/25)\log k}\cdot e^{-(1/25)\log k},
\end{aligned}$$

where the last line follows from Stirling's approximation.

For a random walk with distance $d = c\cdot\sqrt{t}$, $c \geq 4$ to u, the probability to visit u is at most $e^{-c/2}$. Hence the probability that u is visited by none of the random walks with distance at least $4\cdot\sqrt{t}$ is lower bounded by

$$\begin{aligned}
\prod_{j=(1/25)\log k}^{k}\left(1-e^{-\frac{j\cdot\frac{n}{k}}{\sqrt{t}}\cdot\frac{1}{2}}\right) &= \prod_{j=(1/25)\log k}^{k}\left(1-e^{-\frac{j}{\frac{1}{100}\log(k)}\cdot\frac{1}{2}}\right)\\
&= \prod_{j=(1/25)\log k}^{k}\left(1-e^{-\frac{50j}{\log(k1)}}\right)\\
&= \prod_{j=(1/25)\log k}^{k}\left(1-e^{-\frac{50j}{\log(k)}}\right)^{e^{\frac{50j}{\log(k)}}\cdot e^{-\frac{50j}{\log(k)}}}\\
&\geq e^{-\sum_{j=(1/25)\log k}^{k}e^{-\frac{50j}{\log(k)}}}\\
&\geq e^{-\frac{e^{-\frac{2}{\log(k)}}}{1-e^{-50/\log(k)}}}\\
&\geq e^{-\frac{\log(k)}{50}} = k^{-1/50}.
\end{aligned}$$

Hence none of the k random walks will visit u with probability at least

$$10^{-(1/25)\log k}\cdot e^{-(1/25)\log k}\cdot k^{-1/50} \geq k^{-1/2}$$

□

The lower bound on cover time is completed when we prove the existence of a good vertex satisfying the conditions of Lemma 3.30. We do this taking into account Lemma 3.27.

Lemma 3.31. *For arbitrary starting positions of k random walks, we need at least $\Omega((n/k)^2\log^2 k)$ steps to visit all n vertices with probability at least $1/2$.*

Proof. Let $S = \{s_1, s_2, \ldots, s_k\}$ be the k not necessarily distinct starting vertices. Fix $t = 10^{-4} \cdot (n/k)^2 \cdot \log^2 k$. We define intervals $I_i, 0 \leq i < k/\log^2 k$, of the form $I_i = [(i-1)(n/k)\log^2 k, i(n/k)\log^2 k)$. The length of the union of all intervals with even indices is $I = \bigcup_{0 \leq j < k/2\log^2 k} I_{2j}$, and $|I| = 0.5n - o(n)$.

If F is the set of good vertices, then by Lemma 3.27, $|F| \geq 0.8n - o(n)$. Let H be the set of all nodes at distance at least $(n/k)/10$ to node from S; we have $|H| \geq 0.8n$. Thus $|I \cap F \cap H| \geq 0.1n - o(n)$ and $|I \cap F \cap H| \geq 0.09n$ for sufficiently large n. Since each interval I_i is of length $(n/k)\log^2 k$, at least $0.09k/\log^2 k$ intervals with even indices must contain a good vertex satisfying assumptions of Lemma 3.30. We pick one such vertex from each interval. In this way, we obtain set S of $0.09k/\log^2 k$ vertices, at pairwise distances of at least $(n/k)\log^2 k$ from each other.

We denote by Y the event that none of random walks reached a distance more than $40 \cdot \sqrt{t}\log k$ to its origin. We note that $\mathbf{Pr}[Y] \geq 1 - k^{-40}$. We also denote by X the event, that every vertex in S is explored in time t by k random walks. Note, that $\mathbf{Pr}[X|Y] \leq \left(1 - k^{-1/2}\right)^{\frac{k}{\frac{1}{10}\log^2 k}}$ because if event Y happened, then each vertex $s \in S$ remains uncovered with probability at least $k^{-1/2}$ and these events are independent for different vertices in S. Hence

$$\mathbf{Pr}[X] \leq \mathbf{Pr}[Y]\,\mathbf{Pr}[X|Y] + 1 - \mathbf{Pr}[Y] \leq \left(1 - k^{-1/2}\right)^{\frac{k}{\frac{9}{100}\log^2 k}} + k^{-40} \leq 1/2$$

The last inequality holds for $k > 1$. □

Now, the characterization of the cover time of k random walks in the best-case scenario follows directly from Lemmas 3.29 and 3.31.

Theorem 3.32. *The cover time of k random walks on the ring for best-case initial placement is $\Theta((n/k)^2 \log^2 k)$.* □

3.6 Return time on the ring

The considerations of the rotor-router in the previous section concerned the time required to cover all nodes in the initialization phase. As a deterministic system with a finite number of states, the rotor-router eventually reaches its limit behavior, cycling through a finite number of configurations. In this section, we characterize this limit behavior of the rotor-router on the ring using the concept of *return time*, i.e. the maximum over $v \in V$ of the length of the longest time interval during which v is not visited by any agent of the rotor-router system in its limit behavior. We show that this performance parameter of the rotor-router on the ring achieves the best possible value of $\Theta(\frac{n}{k})$, regardless of the initial placement of the agents.

Theorem 3.33. *If $k \in O\left(n^{1/6}\right)$ then after a sufficiently large number of time steps, the k-agent rotor-router system will visit every node of the n vertex ring once every $\Theta(\frac{n}{k})$ time steps.*

Proof. In this proof we will make use of delayed deployments of agents. When analyzing delayed deployments, we apply a different definition of a domain: for each agent a_i, we define its domain $V(a_i)$ as the union of the two maximal sub-paths of the ring adjacent to the location $v(a_i)$, consisting only of nodes whose pointers point towards $v(a_i)$. Note that, once the whole ring has been explored, for $k > 1$, this definition is equivalent to the definition of domains for the undelayed deployment $R[k]$, and is indistinguishable from the point of view of its future evolution in time. In particular, all the lemmas from Section 3.5 bounding evolution of domains hold unchanged.

We will denote the undelayed deployment by $R[k]$ and a specific delayed deployment by D. The considered deployment D consists of two phases. In the first phase, we selectively release (and delay) agents until the whole ring has been covered, and each agent has a domain of size at least $22k + 2$; details of the construction are provided later. In the second phase, we release all agents. By Lemma 3.24, the size of every domain will eventually converge to $O\left(\frac{n}{k}\right)$. Thus, in deployment D every node will eventually be visited once every $O\left(\frac{n}{k}\right)$ time steps.

It remains to be shown that the same holds for the undelayed deployment $R[k]$. Let θ be the total number of rounds during which not all agents are active in D. The construction of D will be such that $\theta \in O\left(\frac{n}{k}\right)$. Now, let t be a time step such that after t every node is visited at least once every $c\frac{n}{k}$ time steps by some agent following deployment D, for some constant $c > 0$. Take any $t^* > t$. We have that in D, every node is visited at least once in the time interval $\left[t^*, t^* + c\frac{n}{k}\right]$. By Lemma 3.3 we have that for any v, $\mathbf{n}_v^{R[k]}\left(t^* - \theta\right) \leq \mathbf{n}_v^D\left(t^*\right)$, and $\mathbf{n}_v^{R[k]}\left(t^* + c\frac{n}{k}\right) \geq \mathbf{n}_v^D\left(t^* + c\frac{n}{k}\right) > \mathbf{n}_v^D\left(t^*\right)$. Thus, for any time t^*, some agent following $R[k]$ visits v within the time interval $[t^* - \theta, t^* + c\frac{n}{k}]$, which contains t^* and is of duration $O\left(\frac{n}{k}\right)$. It follows directly that the refresh time of $R[k]$ is $O\left(\frac{n}{k}\right)$.

It remains to describe the construction for the first phase of deployment D to achieve domains of size at least $22k + 2$, so that not all agents are active at the same time in at most $O\left(\frac{n}{k}\right)$ rounds. We proceed as follows. First, we release all agents until all nodes of the ring have been covered. Next, we release agents one by one, progressing the agent until it has reached a point located a distance of at least $44k + 6$ from the nearest agent. Since the longest sub-path consisting of agents, which do not have a gap of length at least $2(44k + 6) + 1$ between them, is $88k^2 + 13k$, and the moving agent is equivalent to a single-agent rotor-router system, the agent will reach the endpoint of such a sub-path (and complete its movements) within $(88k^2 + 13k)^2$ steps. In total, the moves of all agents in this stage of the construction require $O(k^5)$ rounds. In the next stage, we deploy the agents one by one, so that each agent is moved until it is located at a distance of precisely $22k + 2$ from its location at the beginning of this stage. By a similar analysis, the duration of this stage is $O(k^3)$. Note that during this stage no two agents meet or pass each other on an edge, and so each agent is adjacent to a path of length $22k + 2$ with ports arranged towards its current location. Hence, we have achieved $|V(a_i)| \geq 22k + 2$ within a total of $O(k^5)$ steps, which is $O(n/k)$ for $k \in O\left(n^{1/6}\right)$. □

No strong analogue of the above theorem holds for a system with k random walks. The only property which can be bounded is the expected time between two successive visits to a node, which is precisely equal to n/k on the ring (since the stationary distribution of each of the k walks is uniform with probability $1/n$ on each node). However, the random variable which describes the expected time between successive visits to a node has high variance.

3.7 Discrepancy between the rotor-router and random walk

In the two previous sections we studied the behavior of the k-agent rotor-router on the ring. Now we would like to analyze the asymptotic cover time for different graph classes. However multi-agent rotor-router on more complex graphs probably does not admit such structural behavior (domains, continuous time approximation) as for the case of the ring. Thus in the next few sections we will try a different approach. We will bound the discrepancy between rotor-router and a well-known process of *continuous diffusion*.

In contrast to the case of parallel random walks, in the rotor-router system multiple agents interact with the same set of pointers at nodes, and the agents cannot be considered independent. However, the link between the multi-agent rotor-router and the parallel random walk processes becomes more apparent when the number of agents is extremely large ($k \gg n$), so that multiple agents are located at each node of the graph. Then, a fixed node v of degree d in the graph, which contains $\mathbf{a}_v(t)$ agents at a given moment of time t, will send them out along outgoing links in the next step of the rotor-router process, propagating the pointer at each step, so that each of its neighbours receives either $\lfloor \mathbf{a}_v(t)/d \rfloor$ or $\lceil \mathbf{a}_v(t)/d \rceil$ agents. In an analogous parallel random walk process, the *expected* number of agents following each of the outgoing links of a node v containing $a_t(v)$ agents will be $\mathbf{a}_v(t)/d$. In fact, both the random walk and the rotor-router can be seen as different forms of discretization of the *continuous diffusion process*, in which a node having real-valued load $\mathbf{a}_v(t)$ sends out precisely $\mathbf{a}_v(t)/d$ load to each of its neighbours in the given time step. Discrete diffusion processes appear in research areas including statistical physics and distributed load balancing problems, and some studies of rotor-router-type systems have also been devoted to their diffusive properties. It is known, in particular, that, at any moment time, the difference of the number of agents located at a node between the rotor-router system and that in continuous diffusion is bounded by $\Theta(d \log n\mu^{-1})$ for d-regular graphs with eigenvalue gap μ [138], given identical initialization. This difference can even be bounded by constant for the case of lines [38] and grids [62]. Some other results in the area can also be found in [3, 105]. In this chapter, we observe that, somewhat counter-intuitively, the link between continuous diffusion and the rotor-router can also be exploited for small values of k ($k \ll n$), for which agents as a rule occupy distinct nodes ($\mathbf{a}_v(t) = 1$), and rounding $\mathbf{a}_v(t)/d$ up or down to the nearest integer makes a major difference.

A variant of the multi-agent rotor-router mechanism has been extensively studied in a different setting, in the context of balancing the workload in a network. The single agent is replaced with a number of agents, referred to as *tokens*. Cooper and Spencer [39] study d-dimensional grid graphs and show a constant bound on the discrepancy, defined as the difference between the number of tokens at a given node v in the rotor-router model and the expected number of tokens at v in the random-walk model. Subsequently, Doerr and Friedrich [62] analyze in more detail the distribution of tokens in the rotor-router mechanism on the 2-dimensional grid. Akbari and Berenbrink [3] showed an upper bound of $O(\log^{3/2} n)$ on the discrepancy for hypercubes and a bound of $O(1)$ for a constant-dimensional torus.

Notation. We introduce some auxiliary notation related to random walks and diffusion on the graph. We will denote by $P_{v,u}(t)$ the probability that a simple random walk, starting at node v of the graph, is located at u after exactly t steps of the walk, $t \geq 0$. The transition matrix of the random walk will be denoted by $\mathbf{M}$. For a node $u \in V$, $\mathbf{u}$ will denote a vector of length n with $\mathbf{u}(u) = 1$ and all other entries 0. We recall that the cells of the t-th power of this matrix satisfy the following relation: $\mathbf{u}^\intercal \mathbf{M}^t \mathbf{v} = P_{v,u}(t)$ [5]. The mixing time after which the random walk on the graph G reaches a total variation distance of at most $1/4$ from its stationary distribution will be denoted by $\mathsf{MIX}_{1/4}(G)$.

$$\mathsf{MIX}_{1/4}(G) = \max_{v \in V} \min \left\{ t : \|P_{v,\cdot}(t) - \pi\|_{TV} \leq \frac{1}{4} \right\},$$

where π denotes the vector of the stationary distribution of the random walk. By $P_{v,\cdot}(t)$ we denote the vector of probability distribution of the t-step random walk starting from v. For vector $P_{v,\cdot}(t) - \pi$, the value $\|P_{v,\cdot}(t) - \pi\|_{TV}$ is the total variation distance defined as follows:

$$\|P_{v,\cdot}(t) - \pi\|_{TV} = \frac{1}{2} \sum_{u \in V} |P_{v,u}(t) - \pi_u|$$

This definition can be compared with the following definition of the mixing time used in [71]. We will denote the mixing time defined according to this second definition by $\mathsf{MIX}^*_{1/2}(G)$.

$$\mathsf{MIX}^*_{1/2}(G) = \max_{v \in V} \min \left\{ t : \forall_{u \in V} \frac{3\pi_u}{2} \geq P_{v,u}(t) \geq \frac{\pi_u}{2} \right\}$$

In our considerations we will use a similar value $t_{1/2}(G)$, which satisfies slightly relaxed constraints:

$$t_{1/2}(G) = \max_{v \in V} \min \left\{ t : \forall_{u \in V} P_{v,u}(t) \geq \frac{\pi_u}{2} \right\},$$

which denotes time after which probability of being at any node is at least half of the stationary probability regardless of the starting node of the random walk.

Clearly $t_{1/2}(G) \leq \mathsf{MIX}^*_{1/2}(G)$, and thus we can use results from [71], where authors present upper bounds on the value of $\mathsf{MIX}^*_{1/2}(G)$ for some graph classes.

3.7.1 The main technique

To bound the cover time of the rotor-router, for any moment of time t, we will estimate the difference between the number of visits of the rotor-router to a node $x \in V$ up to time t, and the corresponding *expected* number of visits of parallel random walks, starting from the same initial placement of agents in the graph, to the same node x. (The latter notion can be equivalently interpreted as the total amount of load arriving in rounds 1 to t in a similarly initialized continuous diffusion process in load balancing.) It turns out that the difference (discrepancy) between these two processes is bounded. As soon as the expected total number of visits of parallel random walks to x up to t has exceeded the maximum possible discrepancy with respect to the rotor-router, we can be sure that node x has been visited by the rotor-router at least once up to time t. This is captured by the following lemma.

Lemma 3.34. *Take any graph G. Let t^* be such a time moment that*

$$\forall_{x \in V} \sum_{\tau=0}^{t^*} \left(\mathbf{M}^\tau \mathbf{n}_0\right)_x > \Psi^{(t^*)}$$

where

$$\Psi^{(t)} = \max_{v \in V} \sum_{\tau=0}^{t} \sum_{(u_1,u_2) \in \overrightarrow{E}} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|.$$

Then, the cover time of the k-agent rotor-router with arbitrary initialization on graph G satisfies $C^k_{rr}(G) \leq t^*$.

Before proceeding to prove the lemma, we remark that $\mathbf{M}^\tau \mathbf{n}_0$ is a vector describing the expected number of agents at nodes after τ steps of independent random walks on G. Vector $\mathbf{M}^\tau \mathbf{n}_0$ is of size n, and by $(\mathbf{M}^\tau \mathbf{n}_0)_x$ we denote its x-th coordinate, for any $x \in V$. The expression $\sum_{\tau=0}^{t^*} (\mathbf{M}^\tau \mathbf{n}_0)_x$ on the left-hand side of the inequality is the before-mentioned expected total number of visits of random walks to x up to time t starting from initial agent placement. The expression $\Psi^{(t^*)}$ is a generalization of the so-called 1-*discrepancy* Ψ of the graph, $\Psi = \lim_{t \to +\infty} \Psi^{(t)}$, introduced in [138]. The measure of 1-discrepancy is often applied when comparing a continuous and discrete process at a fixed moment of time t [18,90], whereas herein we compare the *total* distance of two processes over all steps up to time t.

Proof. Consider the total number of visits $\mathbf{n}_u(t)$ at vertex u until step t by the rotor-router. It may be expressed as the sum of the number of agents initially located in u and the number of agents that entered to u from its neighbors (see Section 3.1.1 for details of the argument):

$$\mathbf{n}_u(t) = \mathbf{n}_u(0) + \sum_{v \in \Gamma(u)} \left\lceil \frac{\mathbf{n}_v(t-1) - port(v,u)}{\deg(v)} \right\rceil, \tag{3.21}$$

where $port(v,u) \in \{0, 1, \ldots \deg(v) - 1\}$ denotes the label of the port leading from v to u.

We can rewrite equation (3.21) as follows

$$\mathbf{n}_u(t) = \sum_{v \in \Gamma(u)} \frac{\mathbf{n}_v(t-1)}{\deg(v)} + \mathbf{n}_u(0) + \xi_u(t), \tag{3.22}$$

where $\xi(t)$ is an "error vector" defined as:

$$\xi_u(t) = \sum_{v \in \Gamma(u)} \alpha_{v,u}(t), \tag{3.23}$$

with

$$\alpha_{v,u}(t) = \left(\left\lceil \frac{\mathbf{n}_v(t-1) - port(v,u)}{\deg(v)} \right\rceil - \frac{\mathbf{n}_v(t-1)}{\deg(v)} \right).$$

Note that the values $\alpha_t^{(v,u)}$ are defined over directed arcs of the graph, $(v,u) \in \overrightarrow{E}$, satisfying $|\alpha_{v,u}(t)| \leq 1$ and $\sum_{u \in \Gamma(v)} \alpha_{v,u}(t) = 0$. Consequently, we have $\sum_{(v,u) \in \overrightarrow{E}} \alpha_{v,u}(t)\mathbf{v} = \mathbf{0}$, and:

$$\xi_t = \sum_{(v,u) \in \overrightarrow{E}} \alpha_t^{(v,u)} \mathbf{u} = \sum_{(v,u) \in \overrightarrow{E}} \alpha_t^{(v,u)} \cdot (\mathbf{u} - \mathbf{v}).$$

Now, we rewrite (3.22) as follows:

$$\mathbf{n}(t) = \mathbf{M} \cdot \mathbf{n}(t-1) + (\mathbf{n}(0) + \xi_t), \tag{3.24}$$

where $\mathbf{M}$ is the transition matrix of the random walk on G. Expanding (3.24) we have:

$$\mathbf{n}(t) = \sum_{\tau=0}^{t} \mathbf{M}^\tau \mathbf{n}(0) + \sum_{\tau=0}^{t} \mathbf{M}^\tau \xi_{t-\tau}. \tag{3.25}$$

We will now bound the absolute value of the maximum element of the vector $\sum_{\tau=0}^{t} \mathbf{M}^\tau \xi_{\tau-t}$.

We have

$$\begin{aligned} \left\| \sum_{\tau=0}^{t} \mathbf{M}^\tau \xi_{\tau-t} \right\|_\infty &= \left\| \sum_{\tau=0}^{t} \left(\mathbf{M}^\tau \cdot \sum_{(v,u) \in \overrightarrow{E}} \alpha_{t-\tau}^{(v,u)} \cdot (\mathbf{u} - \mathbf{v}) \right) \right\|_\infty \leq \\ &\leq \left\| \sum_{\tau=0}^{t} \sum_{(v,u) \in \overrightarrow{E}} \alpha_{t-\tau}^{(v,u)} \mathbf{M}^\tau \cdot (\mathbf{u} - \mathbf{v}) \right\|_\infty \end{aligned}$$

Note that since $|\alpha_{t-\tau}^{(u,v)}| \leq 1$

$$\left\| \sum_{\tau=0}^{t} \mathbf{M}^\tau \xi_{\tau-t} \right\|_\infty \leq \left\| \sum_{\tau=0}^{t} \sum_{(v,u) \in \overrightarrow{E}} |\mathbf{M}^\tau \cdot (\mathbf{u} - \mathbf{v})| \right\|_\infty. \tag{3.26}$$

We rewrite the above in terms of probability distributions of of random walk on G after τ steps:

$$\left(\mathbf{M}^\tau \cdot (\mathbf{u}-\mathbf{v})\right)_w = P_{u,w}(\tau) - P_{v,u}(\tau), \qquad (v,u) \in \overrightarrow{E}. \tag{3.27}$$

In this way, we obtain for any $x \in V$:

$$\left|\mathbf{n}_x(t) - \sum_{\tau=0}^{t} \left(\mathbf{M}^\tau \mathbf{n}(0)\right)_x\right| = \left|\sum_{\tau=0}^{t} \left(\mathbf{M}^\tau \xi_{\tau-t}\right)_x\right| \leq \max_{w\in V} \sum_{\tau=0}^{t} \sum_{(v,u)\in\overrightarrow{E}} |P_{u,w}(\tau) - P_{v,w}(\tau)| = \Psi^{(t)}. \tag{3.28}$$

Thus, at time t any node the total number of visits in multi-agent rotor-router deviates from expected number of visits by multiple random walks by at most $\Psi^{(t)}$. Since at time t^* at any node the expected number of visits by random walk is more than $\Psi^{(t^*)}$ by assumption, all nodes have been visited at least once by the rotor-router. □

3.8 Cover time on graphs with small mixing time

Theorem 3.35. *The cover time $C_{rr}^k(G)$ of a k-agent rotor-router with arbitrary initialization on any non-bipartite graph G satisfies*

$$C_{rr}^k(G) \leq t_{1/2}(G) + \frac{2\Delta}{\delta}\frac{n}{k}\Psi.$$

Proof. In order to apply Lemma 3.34, we want to find such a time step t that for any $x \in V$, $\sum_{\tau=0}^{t} \left(\mathbf{M}^\tau \mathbf{n}_0\right)_x > \Psi \geq \Psi^{(t)}$.

Since G is not bipartite then $P_{u,v}(t)$ converges to π_v as t goes to infinity. Thus since $P_{u,v}(t_{1/2}(G)) \geq \pi_v/2$, we have for $\tau \geq t_{1/2}(G)$

$$P_\tau(u,v) \geq \frac{\pi_v}{2} \geq \frac{\deg(v)}{4m} \geq \frac{\delta}{2\Delta n},$$

where π is the stationary distribution of the random walk on G (recall $\pi_v = \deg(v)/2m$).

When considering k independent random walks ($\|\mathbf{n}_0\|_1 = k$), for $\tau \geq t_{1/2}(G)$ we obtain $\left(\mathbf{M}^\tau \mathbf{n}_0\right)(x) \geq \frac{k\delta}{2\Delta n}$. Thus

$$\sum_{t=0}^{t_{1/2}(G)+\frac{2\Delta n}{k\delta}\Psi} \left(\mathbf{M}^\tau \mathbf{n}_0\right)_x > \Psi.$$

Thus, by Lemma 3.34, within time $t_{1/2}(G) + \frac{2\Delta n}{k\delta}\Psi$ all nodes of G have been visited by the k-agent rotor-router. □

In order to apply Theorem 3.35 to special graph classes, we provide convenient bounds on the value of Ψ which hold for regular graphs.

Proposition 3.36. *For any d-regular graph G:*

(i) $\Psi \leq 4\sum_{t=0}^{\mathsf{MIX}_{1/4}(G)} \max_{v\in V} \sum_{\{u_1,u_2\}\in E} |P_{u_1,v}(t) - P_{u_2,v}(t)|$

(ii) $\Psi = O(d\mathsf{MIX}_{1/4}(G))$.

Proof. We want to approximate the value $|P_{u_1,v}(t) - P_{u_2,v}(t)|$. Let $\mathbf{M}$ be the transition matrix for the random walk on G and for any node u let $\mathbf{u}$ be a vector of length n with $\mathbf{u}(u) = 1$ and all other entries 0 and let $\mathbf{u}^\intercal$ be its transposition. We have

$$|P_{u_1,v}(t) - P_{u_2,v}(t)| = \left|\mathbf{u}_1^\intercal \mathbf{M}^t \mathbf{v} - \mathbf{u}_2^\intercal \mathbf{M}^t \mathbf{v}\right| = \left|(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^t \mathbf{v}\right|. \tag{3.29}$$

Set $t = \tau + a\mathsf{MIX}_{1/4}(G)$ in the equation (3.29).

$$|P_{u_1,v}(t) - P_{u_2,v}(t)| = \left|(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau \cdot \left(\mathbf{M}^{a\mathsf{MIX}_{1/4}(G)} \cdot \mathbf{v}\right)\right|. \tag{3.30}$$

Vector $\mathbf{M}^{a\mathsf{MIX}_{1/4}(G)} \cdot \mathbf{v}$ is the distribution of position of random walk starting from v after $a\mathsf{MIX}_{1/4}(G)$. Let $\frac{\mathbf{1}}{n}$ be vector of size n with all values $1/n$. Then

$$\mathbf{M}^{a\mathsf{MIX}_{1/4}(G)} \cdot \mathbf{v} = \tfrac{\mathbf{1}}{n} + \mathbf{err}_a^v,$$

where $\mathbf{err}_a^v$ is the vector of deviations from stationary distribution for random walk of length $a\mathsf{MIX}_{1/4}(G)$ starting at v. Since G is regular then $\frac{\mathbf{1}}{n}$ is its stationary distribution. From the properties of mixing time of random walk [121] we have that $\sum_{w\in V} |\mathbf{err}_a^v(w)| \leq 2^{-a+1}$. We transform the equation (3.30)

$$\begin{aligned}|P_{u_1,v}(t) - P_{u_2,v}(t)| &= \left|(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau \cdot \left(\tfrac{\mathbf{1}}{n} + \mathbf{err}_a^v\right)\right| \\ &= \left|(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau \cdot \tfrac{\mathbf{1}}{n} + (\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau \cdot \mathbf{err}_a^v\right|.\end{aligned}$$

Vector $\frac{\mathbf{1}}{n}$ is an eigenvector of matrix $\mathbf{M}$ thus $\mathbf{M}^\tau \cdot \frac{\mathbf{1}}{n} = \frac{\mathbf{1}}{n}$. Clearly $(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \frac{\mathbf{1}}{n} = 0$. We have

$$\begin{aligned}\sum_{\{u_1,u_2\}\in E} |P_{u_1,v}(t) - P_{u_2,v}(t)| &= \sum_{\{u_1,u_2\}\in E} |(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau \cdot \mathbf{err}_a^v| \\ &\leq \left(\sum_{\{u_1,u_2\}\in E} |(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau|\right) \cdot |\mathbf{err}_a^v| \\ &\leq \left\|\sum_{\{u_1,u_2\}\in E} |(\mathbf{u}_1 - \mathbf{u}_2)^\intercal \cdot \mathbf{M}^\tau|\right\|_\infty \cdot \sum_{w\in V} |\mathbf{err}_a^v(w)| \\ &\leq 2^{-a+1} \max_{v\in V} \sum_{\{u_1,u_2\}\in E} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|,\end{aligned}$$

where $|\mathbf{err}|$ for a vector $\mathbf{err}$ denotes the vector of absolute values of elements. Now we can bound the value of Ψ

$$\begin{aligned}
\Psi &= \max_{v\in V}\sum_{t=0}^{\infty}\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(t)-P_{u_2,v}(t)| \le \sum_{t=0}^{\infty}\max_{v\in V}\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(t)-P_{u_2,v}(t)| \\
&\le \sum_{a=0}^{\infty}\sum_{\tau=0}^{\mathsf{MIX}_{1/4}(G)-1}\max_{v\in V}\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(\tau+a\mathsf{MIX}_{1/4}(G))-P_{u_2,v}(\tau+a\mathsf{MIX}_{1/4})| \\
&\le \sum_{a=0}^{\infty}\sum_{\tau=0}^{\mathsf{MIX}_{1/4}(G)-1}2^{-a+1}\max_{v\in V}\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(\tau)-P_{u_2,v}(\tau)| \\
&= 4\sum_{\tau=0}^{\mathsf{MIX}_{1/4}(G)-1}\max_{v\in V}\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(\tau)-P_{u_2,v}(\tau)|,
\end{aligned}$$

which finishes the proof of (i). To prove (ii) observe that

$$\sum_{\{u_1,u_2\}\in E}|P_{u_1,v}(\tau)-P_{u_2,v}(\tau)| \le \sum_{\{u_1,u_2\}\in E}(P_{u_1,v}(\tau)+P_{u_2,v}(\tau)) = \sum_{u\in V}dP_{u,v}(\tau) = d,$$

because for regular graphs, $P_{u,v}(\tau) = P_{v,u}(\tau)$. □

By combining Theorem 3.35 and Proposition 3.36, we will obtain upper bounds on the cover time of the rotor-router in regular graphs. At this point we provide an auxiliary result, which allows us to extend all our considerations to almost-regular graphs, as well as to show that our bounds on cover time hold regardless of whether the considered graph has self-loops or not. The proof relies on a variant of the *delayed deployment* technique for the rotor-router, introduced in Section 3.1.1.

Proposition 3.37. *Consider a graph G' constructed from G by adding self-loops to vertices, so that in the port ordering at any vertex there are at most x consecutive self-loops. Then, $C^k_{rr}(G')/(x+1) \le C^k_{rr}(G) \le C^k_{rr}(G')$.*

Proof. It suffices to compare a pair of rotor-router systems in which all agents are initialized at the same positions in G and G', and the port orderings of G and G' are identical, when disregarding self-loops of G', establishing the relation between cover times of such a pair of systems.

The proof that $C^k_{rr}(G') \le C^k_{rr}(G)$ follows directly from Lemma 3.3, since we can construct a delayed deployment D for graph G which simulates the self-loops as in G'. Movements of agents in D on G will be exactly the same as in undelayed k-agent rotor-router operating on G'.

To prove the bound $C^k_{rr}(G')/(x+1) \le C^k_{rr}(G)$, we prove by induction, that more generally for any time t and vertex v

$$\mathbf{n}_v(t) \le \mathbf{n'}_v\left((x+1)t\right),$$

where $\mathbf{n}_v(t)$ and $\mathbf{n'}_v((x+1)t)$ is the total number of visits at vertex v until time t for rotor-router on graphs G and G' respectively (where visits of an agent coming in from a self-loop of v do not count towards $\mathbf{n'}_v(t)$).

Since agents are initialized at the same positions then the claim is true for $t = 0$. Assume that it is true for some $t \geq 0$. Let for any $e \in \overrightarrow{E}$ denote by $r_t(e)$ and $r'_t(e)$ the total number of traversals of edge e for rotor-router in G and G' respectively until time t. In G all agents that entered some node v until step t left v until step $t+1$. On the other hand in G' all agents that entered v until step $(x+1)t$, left until step $(x+1)(t+1)$. Since the order of pointers when considering only arcs from $\overrightarrow{E}$ is the same in both graphs, for any arc $e \in \overrightarrow{E}$ we have

$$r_{t+1}(e) \leq r'_{(x+1)(t+1)}(e)$$

Since this holds for every arc $e \in \overrightarrow{E}$, we have

$$\mathbf{n}_v(t+1) \leq \mathbf{n'}_v((x+1)(t+1)),$$

which completes the inductive proof. The above relation immediately implies that $C^k_{rr}(G) \geq (x+1) \leq C^k_{rr}(G')$, which completes the proof. □

Taking into account Theorem 3.35 and Propositions 3.36 and 3.37, we obtain an upper bound of $O(mD/k)$ on the cover time of the rotor-router in a wide class of almost regular graphs with small mixing time. The complementary lower bound $C^k_{rr}(G) = \Omega(mD/k)$ was shown in Section 3.4. These bounds hold for all k, until the trivial bound $C^k_{rr}(G) = \Omega(D)$ is reached, for $k = \Omega(m)$.

Theorem 3.38. *For any graph G such that $t_{1/2}(G) = O(D)$, $\mathsf{MIX}_{1/4}(G) = O(D)$, and $\Delta/\delta = O(1)$, the cover time of the k-agent rotor-router in the worst-case initialization of the system is:*

$$C^k_{rr}(G) = \Theta\left(\max\left\{\frac{mD}{k}, D\right\}\right).$$

Proof. Note that we only need to consider the case of $k = O(m)$ and show $C^k_{rr}(G) = \Theta(mD/k)$. The lower bound $C^k_{rr}(G) = \Omega(mD/k)$ was shown in Section 3.4. We will focus on the upper bound. Consider a Δ-regular graph G' constructed from G by adding self-loops to vertices. Since $\Delta/\delta = 1$ then adding self-loops to G increases the mixing time by no more than a constant factor, thus $\mathsf{MIX}_{1/4}(G') = O(D)$ and $t_{1/2}(G') = O(D)$.

By Theorem 3.35 and Proposition 3.36 cover time $C^k_{rr}(G')$ of rotor-router on G' is

$$C^k_{rr}(G') \leq t_{1/2}(G) + O\left(\frac{\Delta n}{k\delta}\Psi(G')\right) \leq O(D) + O\left(\frac{ndD}{k}\right) = O\left(\max\left\{\frac{mD}{k}, D\right\}\right).$$

Thus, taking into account Proposition 3.37, deployment D will also cover graph G in time $O\left(\max\left\{\frac{mD}{k}, D\right\}\right)$. □

Theorem 3.38 immediately implies the results stated in Table 3.2 for the case of complete graphs, degree-constrained expanders, and Erdős-Renyi graphs with edge

probability $p > (1+\varepsilon)\frac{\log n}{n}$. For cliques it is easy to see that $t_{1/2}(G) = O(1)$. For degree-constrained expanders, and Erdős-Renyi graphs, a bound on value $t_{1/2}(G)$ can be found in [71] ($t_{1/2}(G)$ is upper-bounded by the mixing time $\mathsf{MIX}^*_{1/2}(G)$).

The classes of tori, cycles, and hypercubes require more careful analysis; we consider them in the following sections.

3.9 Cover time on the ring revisited

The general case result shown in Section 3.3 allows us to upper-bound the cover time of the k-rotor-router system on the cycle by $O\left(\max\{\frac{n^2}{\log k}, n\}\right)$, for any $k \geq 1$. On the other hand, the structural result from Section 3.5 is showing lower bound of $\Omega\left(\frac{n^2}{\log k}\right)$ only for $k < n^{1/11}$. In the following, we extend this lower bound to arbitrary values of k. The proof relies on a modification of the approach used in the proof of Lemma 3.34: whereas Lemma 3.34 can only be used to upper bound cover time, this time we perform a different transformation of (3.25) for a specific initialization of agents starting from a single node on the ring, for which we can show that the "error term" associated with vector $\xi_{t-\tau}$ is negative. Intuitively, this behaviour is due to an initialization of pointers which delays progress of the agents going along the path to the most distant node of the ring. We eventually obtain the following result.

Theorem 3.39. *If G is a cycle of size n then cover time of k-agent rotor-router is*

$$C^k_{rr}(G) = \Theta\left(\max\left\{\frac{n^2}{\log k}, n\right\}\right).$$

Proof. The upper bound is a direct consequence of Theorem 3.6 and the fact that adding more agents cannot slow down exploration. The lower bound is for $k \leq n^{1/11}$ is shown in Theorem 3.15.

To prove the lower bound for $k > n^{1/11}$, we consider for simplicity a cycle of even length, with $n' = n/2$, and we divide the cycle into two subpaths of length n' along an axis of symmetry crossing a pair of edges. We then perform an initialization of the rotor-router system which is symmetric with respect to this axis (see Section 3.5 for an explanation why this argument is correct and sufficient). In all further considerations, we will restrict our attention to only one of the subpaths of the cycle. We will number its nodes $u_1, \ldots, u_{n'}$, with $u_1 = v$ and $u_{n'} = w$ being its endpoints. We now initialize the rotor-router on the considered path so that all agents are located at vertex v, all ports along the shortest path to v get label 0, and the port leading away from v gets label 1. We will show that it takes the agents in the considered system a long time to reach node w.

Fix a moment of time t, and suppose that none of the agents has reached w until the end of round $t-1$ inclusive. We will now show that if the condition $t < n'^2/(12\log k)$ is satisfied, then none of the agents will reach w at time t either, i.e., $\mathbf{n}_w(t) = 0$.

We rely on some of the techniques from the proof of Lemma 3.34. We have from (3.25):

$$\mathbf{n}_t(w) = \sum_{\tau=0}^{t} \mathbf{w}^\intercal \mathbf{M}^\tau \mathbf{n}(0) + \sum_{\tau=0}^{t} \mathbf{w}^\intercal \mathbf{M}^\tau \xi_{t-\tau}. \tag{3.31}$$

Consider a pair of nodes u_i, u_{i+1}, where we recall that u_{i+1} is the neighbor of u_i that is further from v (and closer to w). Then, due to the chosen port initialization, we have

$$\alpha_{t-\tau}^{(u_i,u_{i+1})} = (\lceil (\mathbf{n}_{u_i}(t-\tau-1)-1)/2 \rceil - \mathbf{n}_{u_i}(t-\tau-1)/2) \leq 0. \tag{3.32}$$

Since the considered graph has degree 2, we have $\alpha_{t-\tau}^{(u_i,u_{i-1})} + \alpha_{t-\tau}^{(u_i,u_{i+1})} = 0$, and we may write by rearranging the definition of $\xi_{t-\tau}$:

$$\xi_{t-\tau} = \sum_{i=1}^{n'-1} (\alpha_{t-\tau}^{(u_i,u_{i-1})} \mathbf{u}_{i-1} + \alpha_{t-\tau}^{(u_i,u_{i+1})} \mathbf{u}_{i+1}) = \sum_{i=1}^{n'-1} \alpha_{t-\tau}^{(u_i,u_{i+1})} (\mathbf{u}_{i+1} - \mathbf{u}_{i-1}).$$

In the above sum, u_0 should be interpreted as the mirror reflection of u_1 in the other subpath of G, whereas index n' was discarded from the sum since node $w = u_{n'}$ was not visited before time t by assumption.

Introducing the above into (3.31), taking into account that $\mathbf{n}_0 = k\mathbf{v}$, and expanding, we obtain:

$$\begin{aligned} \mathbf{n}_w(t) &= \sum_{\tau=0}^{t} \mathbf{w}^\intercal \mathbf{M}^\tau k\mathbf{v} + \sum_{\tau=0}^{t} \left(\mathbf{w}^\intercal \mathbf{M}^\tau \sum\nolimits_{i=1}^{n'-1} \alpha_{t-\tau}^{(u_i,u_{i+1})} (\mathbf{u}_{i+1} - \mathbf{u}_{i-1}) \right) = \\ &= k \sum_{\tau=0}^{t} P_{w,v}(\tau) + \sum_{\tau=0}^{t} \sum_{i=1}^{n'-1} \alpha_{t-\tau}^{(u_i,u_{i+1})} (P_{w,u_{i+1}}(t-\tau) - P_{w,u_{i-1}}(t-\tau)) \leq k \sum_{\tau=0}^{t} P_{w,v}(\tau), \end{aligned} \tag{3.33}$$

where the last inequality holds because $\alpha_{t-\tau}^{(u_i,u_{i+1})} \leq 0$ by equation (3.32), whereas $P_{w,u_{i+1}}(T) \geq P_{w,u_{i-1}}(T)$ holds for any time moment T, by the basic properties of a random walk on the cycle starting from vertex w.

In order to show that $\mathbf{n}_t(w) = 0$, it suffices to show that $\mathbf{n}_t(w) < 1$, since this value is an integer. Taking into account (3.33) we only need to show that $\sum_{\tau=0}^{t} P_{w,v}(\tau) < 1/k$. We apply the following standard bound based on normal approximation of $P_{w,v}(\tau)$, recalling that the distance between w and v is $n'-1$, $t < n'^2/(12 \log k)$, and $k > n'^{1/11}$:

$$\sum_{\tau=0}^{t} P_{w,v}(\tau) < t \cdot \frac{1}{\sqrt{t}} e^{-n'^2/t} = \sqrt{t} \cdot e^{-n'^2/t} < n' k^{-12} = k^{-1}(n' k^{-11}) < k^{-1},$$

which completes the proof. □

3.10 Cover time on the torus

For the d-dimensional torus, Theorem 3.38 is not applicable, since the mixing time of the torus is $\mathsf{MIX}_{1/4}(G) = \Theta(n^{2/d})$ [121], for constant d, whereas its diameter is $D = \Theta(n^{1/d})$.

To bound the cover time for $k > n^{1-1/d}$, in view of Proposition 3.37, we can equivalently consider the torus with d self-loops added on each node. We will now rely on Lemma 3.34, taking into account tighter bounds on $\Psi^{(t)}$ for small values of t. The following bound can be shown by a straightforward Markovian coupling argument.

Lemma 3.40. *If graph G is a d-dimensional torus with d self-loops at each node, then $\Psi^{(t)} \leq 24d\sqrt{t}$.*

Proof. We apply the Markovian coupling technique to bound values $|P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|$, where u_1 and u_2 are neighbors. To construct a coupling on G', consider three random walks. Let walk W_1 start from u_1, and let walks W_2 and W_3 start from u_2. We will view the random walk on G' in every step as choosing one among $2d$ edges and traversing it with probability $1/2$.

Walk W_1 is a standard random walk on G' and W_2 will be constructed based on W_1. When W_1 in a step chooses an edge, W_2 in the same step chooses the same edge.

Nodes u_1 and u_2 have the same coordinates in $d-1$ dimensions and differ by 1 in one dimension. Denote by d^* the dimension on which u_1 and u_2 differ.

If W_1 chooses a dimension different from d^* then W_2 makes the same choice whether to traverse the chosen edge or not. Thus, the positions of walks W_1 and W_2 will never differ on a dimension different from d^*. Consider the distance between W_1 and W_2 in dimension d^*. If W_1 chooses an edge from d^* then when choosing whether to traverse it, W_2 makes the opposite choice (if W_1 traverses it, W_2 does not). Thus, whenever W_1 chooses dimension d^*, the distance between these walks decreases by 1 with probability $1/2$ and increases by 1 with probability $1/2$. Denote by T the random variable denoting the time of meeting of walks W_1 and W_2. Walk W_3 follows W_2 in steps $0, 1, \ldots, T$ and then follows W_1. The pair (W_1, W_3) forms a coupling. Using the theory of coupling [121, Theorem 5.2], since $P\{T > \tau\}$ is the probability that walks W_1 and W_3 have coupled after time τ, we obtain

$$\frac{1}{2} \left\| P_{u_1,\cdot}(\tau) - P_{u_2,\cdot}(\tau) \right\|_1 \leq P\{T > \tau\}, \tag{3.34}$$

where $\left\| P_{u_1,\cdot}(\tau) - P_{u_2,\cdot}(\tau) \right\|_1 = \sum_{v \in V} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|$.

Since the initial distance between the walks is 1, by [121, Theorem 2.17] we obtain for $\tau > 0$

$$P\{T > \tau\} \leq 12/\sqrt{\tau} \tag{3.35}$$

By equations (3.34), (3.35) we have

$$\sum_{v \in V} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)| \leq \frac{24}{\sqrt{\tau}} \tag{3.36}$$

Let G' be a d-dimensional torus with n vertices and d self-loops at each node. We have

$$\frac{12dn}{\sqrt{\tau}} \geq \sum_{\{u_1,u_2\} \in E} \sum_{v \in V} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)| = \sum_{v \in V} \sum_{\{u_1,u_2\} \in E} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|$$

Observe that by the symmetry of G', for every v the value $\sum_{\{u_1,u_2\}\in E} |P_\tau(u_1,v) - P_\tau(u_2,v)|$ is the same, thus for any v

$$\frac{12dn}{\sqrt{\tau}} \geq n \sum_{\{u_1,u_2\}\in E} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)|$$

We obtain

$$\begin{aligned}\Psi^{(t)} &= \max_{v\in V} \sum_{\tau=0}^{t} \sum_{\{u_1,u_2\}\in E} |P_{u_1,v}(\tau) - P_{u_2,v}(\tau)| \leq d/2 + 12d \sum_{\tau=1}^{t} \tau^{-1/2} \\ &\leq d/2 + 12d \left(1 + \int_1^t x^{-1/2} dx\right) \leq 24d\sqrt{t}\end{aligned}$$

which completes the proof. □

Introducing the above bound into Lemma 3.34 and taking into account properties of the random walk in the torus, for $k = k'n^{1-1/d}$ ($k' > 1$), we eventually obtain a bound on cover time of the form $O\left(\frac{D^2}{\log k'}\right)$, Somewhat surprisingly, this bound is tight, and we propose an initialization of the rotor-router system which achieves this bound precisely. The proof of tightness relies on the bound on cover time for the ring introduced in Sections 3.5 and 3.9. In this way, we obtain a complete characterization of the speedup of the rotor-router on the torus.

Theorem 3.41. *If G is a torus of constant dimension then cover time of k-agent rotor-router is*

(i) $C_{rr}^k(G) = \Theta\left(\frac{mD}{k}\right)$, *for* $k \leq n^{1-1/d}$,

(ii) $C_{rr}^k(G) = \Theta\left(\max\{\frac{D^2}{\log k'}, D\}\right)$, *for* $k = k'n^{1-1/d}$, $k > n^{1-1/d}$.

Proof. We start by proving (i). We want to show that for tori $t_{1/2}(G) = O(n^{2/d})$. For sufficiently large n, the distribution of the random walk on the infinite d-dimensional grid can by approximated by the normal distribution $\mathcal{N}_d(0,\sigma^2)$, where $\mathcal{N}_d$ is a product of d independent normal distributions and $\sigma = \sqrt{t}/d$ is the standard deviation in each dimension. Hitting probabilities $P_{u,v}(t)$ on the infinite grid lower-bound the hitting probabilities for the corresponding pair u, v on torus G. Thus, it is sufficient to bound the hitting probabilities on the infinite grid for points at distance at most $\Theta(n^{1/d})$ in every dimension from the starting point.

In the infinite grid, the minimum probability will be achieved for the point which is at the maximum distance in every dimension from the starting point. This probability satisfies

$$P_{u,v}(t) \geq (1 + o(1)) \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-n^{2/d}}{\sigma^2}}\right)^d = \Theta(1) t^{-d/2} e^{\frac{-d^3 n^{2/d}}{t}}, \tag{3.37}$$

where $P_{u,v}(t)$ is the probability of being in v after t steps of a random walk on the torus G starting from u. If we set $t = cn^{2/d}d^2/\log(d/\sqrt{2\pi})$, where c is an appropriately chosen

constant, we obtain $P_t(u,v) \geq n^{-1}/2$. This means that for tori we have $t_{1/2}(G) = O(n^{2/d})$. In the range of $k \leq n^{1-1/d}$, we can apply Theorem 3.35, taking advantage of a known tight bound on $\Psi = \Theta(n^{1/d})$ [138]. In this way, we obtain: $C^k_{rr}(G) = O\left(n^{2/d} + \frac{n^{1+1/d}}{k}\right) = O\left(\frac{mD}{k}\right)$. Moreover, the complementary lower bound $C^k_{rr}(G) = \Omega(mD/k)$ holds for all graphs by Theorem 3.11. This resolves the case of $k \leq n^{1-1/d}$.

Now we want to prove (ii). We first show the lower bound. We want to construct an initialization of the rotor-router that will lead to the desired cover time. Take a set of all nodes that have coordinate 0 in the first dimension. There are $n^{1-1/d}$ such nodes. Assume that k is divisible by $n^{1-1/d}$ and place all agents evenly on these nodes. In each node we have $k/n^{1-1/d}$ agents. Now we initiate the pointers. The initial position of the pointer is on the edge in the first dimension that is along the shortest path to the closest node with coordinate 0 on the first dimension. Initialization of the cyclic order of the arcs can be arbitrary but the same on every node. With such initialization if we take all nodes with the same coordinate in the first dimension then we will always have the same number of agents on these nodes. Thus we can see the exploration of the torus as the exploration of the ring. We know that the exploration of a cycle of length $n^{1/d}$ using $k/n^{1-1/d}$ takes time $\Theta\left(\frac{n^{2/d}}{\log(k/n^{1-1/d})}\right)$. If the number of agents k is not divisible by $n^{1-1/d}$ we can again use Lemma 3.3 to observe that exploration with k agents will not be faster than exploration with $k' = n^{1-1/d}\lceil k/n^{1-1/d}\rceil$ agents.

Now we want to prove the upper bound. By Lemma 3.40 we know that $\Psi^{(t)} \leq 24d\sqrt{t}$. Thus we can use Lemma 3.34 if we find such t^*, that

$$\forall_{x\in V} \sum_{\tau=0}^{t^*} (\mathbf{M}^\tau \mathbf{n}(0))_x \geq 48d\sqrt{t^*}$$

It is sufficient to find such t^* such that for all $t \in [t^*/2, t^*]$ all elements of vector $\mathbf{M}^t\mathbf{n}(0)$ are at least equal to $96d/\sqrt{t^*}$. Vector n_0 is non-negative with sum k thus it is sufficient to find t^*, such that all elements of matrix $\mathbf{M}^{t^*/2}$ are at least $96d/(k\sqrt{t^*})$. We want to find $t' = t^*/2$ such that for any $u, v \in V$

$$P_{u,v}(t') \geq 96d/(k\sqrt{2t'}).$$

We can use equation (3.37) again. We want to find t such that

$$\Theta(1)t^{-d/2}e^{-\frac{d^3n^{2/d}}{t}} \geq \Theta(1)t^{-1/2}/k$$

If we take $k = n^{1-1/d}k'$ and $t = n^{2/d}/x$ then we obtain

$$x^{(d-1)/2}e^{-d^3x} \geq \Theta(1/k')$$

$$\log(k'/c) \geq d^3x - (d-1)/2 \log x,$$

where c is a constant. Thus, if $x = \log(k'/c)/d$ and $k' > c$, then the inequality is

satisfied. Thus, for $k > cn^{1-1/d}$ agents, we have the cover time $\Theta\left(\frac{d^3n^{2/d}}{\log(k/(cn^{1-1/d}))}\right) = \Theta\left(\frac{D^2}{\log(k/n^{1-1/d}))}\right)$, which completes the proof. □

3.11 Cover time on the hypercube

For the hypercube with $n = 2^d$ vertices, the value of Ψ has been precisely derived in [18]. The corresponding asymptotic formula is $\Psi = \Theta(\log^2 n)$. The value of $t_{1/2}(G)$ was shown to satisfy $t_{1/2}(G) = O(\log n \log\log n)$ [71, Lemma 5.2]. Using these results in combination with Theorem 3.35, we obtain the following corollary.

Corollary 3.42. *If G is a hypercube with n vertices then $C^k_{rr}(G) = \Theta\left(\frac{n\log^2 n}{k}\right) = \Theta\left(\frac{mD}{k}\right)$, for $k \leq n\frac{\log n}{\log\log n}$.*

The behavior of the rotor-router on the hypercube for $k > k_1 = n\frac{\log n}{\log\log n}$ is not completely understood. For $k = k_1$, the value of cover time is $O(\log n \log\log n)$. Interestingly, we can show that there exists a flat "plateau" region above k_1 in which the asymptotic cover time of the hypercube is precisely $\Theta(\log n \log\log n)$. The proof proceeds along slightly more complex lines than the proof of Theorem 3.39. We show that in the considered range of k, $\Theta(\log n \log\log n)$ time is required for k agents starting at one corner of the hypercube to reach the opposite corner, given an arrangement of ports at each node in which the pointer first traverses all ports leading the agent towards the starting vertex.

Theorem 3.43. *If G is a hypercube of size $n = 2^d$ then the cover time of k-agent rotor-router with $k \leq n \cdot 2^{\log^{1-\varepsilon} n}$ agents is $C^k_{rr}(G) > \frac{\varepsilon}{10}\log n \log\log n$, where $\varepsilon \in (0,1)$ is an arbitrary fixed constant.*

Proof. We identify each vertex with a $d = \log_2 n$ bit vector of coordinates, with $v = 0^d$ and $w = 1^d$ being antipodal vertices. We partition set V into layers $L_0, \ldots, L_d$, such that all vertices belonging to layer L_i have exactly i ones in their binary representation. Now, for each vertex $u \in L_i$, $0 \leq i \leq d$, the port labeling of u is set so that ports $0, 1, \ldots, (i-1)$ point to the i neighbours of u belonging to layer L_{i-1} in arbitrary order, while ports $i, (i+1), \ldots, (d-1)$ point to the $d-i$ neighbours of u belonging to layer L_{i+1} in arbitrary order. The system is initialized with k agents placed on node v.

Acting in a similar way as in the proof of Theorem 3.39, we will show that for all $t \leq \frac{\epsilon}{10}\log n \log\log n$, we have $\mathbf{n}_w(t) = 0$. Once again, we consider equality (3.31), and we prove that each of the summed expressions $\mathbf{w}^\intercal \mathbf{M}^\tau \xi_{t-\tau}$ is negative, for all $0 \leq \tau \leq t$. We can represent vector $\xi_{t-\tau}$ as follows:

$$\xi_{t-\tau} = \sum_{i=0}^{d} \sum_{u \in L_i} \left(\sum_{\substack{(u,u^-) \in \overrightarrow{E} \\ u^- \in L_{i-1}}} \alpha^{(u,u^-)}_{t-\tau} \mathbf{u}^- + \sum_{\substack{(u,u^+) \in \overrightarrow{E} \\ u^+ \in L_{i+1}}} \alpha^{(u,u^+)}_{t-\tau} \mathbf{u}^+ \right).$$

Due to the fact that for each node u from layer L_i, all the ports pointing to layer L_{i-1} are always visited before those in layer L_{i+1}, and that there are no other ports adjacent to u, we have:

$$\sum_{\substack{(u,u^+)\in\overrightarrow{E}\\ u^+\in L_{i+1}}} \alpha_{t-\tau}^{(u,u^+)} = -\sum_{\substack{(u,u^-)\in\overrightarrow{E}\\ u^-\in L_{i-1}}} \alpha_{t-\tau}^{(u,u^-)} \leq 0.$$

Moreover, by the symmetry of the random walk with respect to coordinates of the binary vector representation of the node, we have that for all nodes x belonging to a layer j, the probability that a walk starting at w is located at x after τ steps is the same, and denoted as $P_{w,x}(\tau) = P_w^{(j)}(\tau)$.

Combining the above observations, we obtain:

$$\begin{aligned}
\mathbf{w}^\intercal \mathbf{M}^\tau \xi_{t-\tau} &= \sum_{i=0}^{d}\sum_{u\in L_i}\left(\sum_{\substack{(u,u^-)\in\overrightarrow{E}\\ u^-\in L_{i-1}}} \alpha_{t-\tau}^{(u,u^-)} P_{w,u^-}(\tau) + \sum_{\substack{(u,u^+)\in\overrightarrow{E}\\ u^+\in L_{i+1}}} \alpha_{t-\tau}^{(u,u^+)} P_{w,u^+}(\tau)\right) = \\
&= \sum_{i=0}^{d}\sum_{u\in L_i}\left(P_w^{(i-1)}(\tau)\sum_{\substack{(u,u^-)\in\overrightarrow{E}\\ u^-\in L_{i-1}}} \alpha_{t-\tau}^{(u,u^-)} + P_w^{(i+1)}(\tau)\sum_{\substack{(u,u^+)\in\overrightarrow{E}\\ u^+\in L_{i+1}}} \alpha_{t-\tau}^{(u,u^+)}\right) = \\
&= \sum_{i=0}^{d}\sum_{u\in L_i}\left(\left(P_w^{(i+1)}(\tau) - P_w^{(i-1)}(\tau)\right)\sum_{\substack{(u,u^+)\in\overrightarrow{E}\\ u^+\in L_{i+1}}} \alpha_{t-\tau}^{(u,u^+)}\right) \leq 0,
\end{aligned}$$

where in the last inequality we took into account that for all moments of time T, $P_w^{(i+1)}(T) \geq P_w^{(i-1)}(T)$, by the properties of the random walk on the hypercube (recall that $w = 1^d$).

Acting as in the derivation of (3.33), we obtain:

$$\mathbf{n}_w(t) \leq k\sum_{\tau=0}^{t} P_{w,v}(\tau) = k\sum_{\tau=0}^{t} P_w^{(0)}(\tau).$$

A derivation of an upper bound on $\sum_{\tau=0}^{t} P_w^{(0)}(\tau)$ is obtained in [71] (the authors of [71] consider the lazy random walk on a hypercube with d self-loops at each node, but the same result can be applied to the hypercube without self-loops, after relaxing time bounds by a constant factor of 4):

$$\mathbf{n}_w(t) \leq k\sum_{\tau=0}^{t} P_w^{(0)}(\tau) \leq \frac{4tk}{n}\left(1 - \left(1 - \frac{1}{\log n}\right)^{4t}\right)^{\log n}.$$

Substituting $k \leq n \cdot 2^{\log^{1-\varepsilon} n}$ and $t \leq \frac{\varepsilon}{10}\log n \log\log n$, we obtain $\mathbf{n}_w(t) < 1$, which completes the proof. □

We leave the question of the cover time of the rotor-router on the hypercube for $k > n \cdot 2^{\log^{1-\varepsilon} n}$ as open, noting that, in view of our results, it can be bounded as between $O(\log n \log\log n)$ and $\Omega(\log n)$.

3.12 Conclusions

We have shown that the worst-case speedup on many graph classes is equal for both the k-agent random walk and the k-agent rotor-router, even though this speed up has a different explanation in both cases. For the random walk, it is a consequence of the properties of probability distributions of independent Markovian processes, while for the rotor-router, it results directly from the interactions between different agents and the pointers in the graph. It is particularly interesting that the range of speedups for the multi-agent rotor-router turned out to correspond precisely to the conjectured range for multiple random walks.

In this chapter we also formalized the intuition about similarities between rotor-router and the continuous diffusion process. Intuitively, the rotor-router can be seen as discretization of the continuous diffusion process. We showed that it is possible to bound the difference in cumulative loads between these two processes and obtain asymptotically tight values of the cover time of the rotor-router for many graph classes.

An interesting question related to this chapter is the following generalization. Consider a problem of exploration with oblivious mobile agents of a port-labelled graphs with whiteboards. Agents are oblivious, thus cannot carry any memory while traversing edges but can leave some information on the whiteboards. Such agents clearly can simulate the rotor-router process. We may ask the following questions:

- Is there any more efficient algorithm in such generalized model than the rotor-router?
- What happens if we allow agents to carry, for example, a constant number of memory bits while traversing an edge?

Finally, we remark that most of our considerations are done for the worst-case initialization and a possible line of study is the case of randomly initialized pointers. Note that such a scenario has been studied for the single-agent rotor-router [63]. It would also be interesting to see examples of graphs where the rotor-router achieves speedup different than $\log k$ or k even for a small number of agents.

Chapter 4

Collision-free exploration

In this chapter we consider graph exploration with multiple agents in a different setting. The agents are required to synchronously move along the network edges in a *collision-free* way, i.e., in no round may two agents occupy the same node. In each round, an agent may choose to stay at its currently occupied node or to move to one of its neighbors. An agent has no knowledge of the number and initial positions of the other agents. We are looking for the shortest possible time required to complete the *collision-free network exploration*, i.e., to reach a configuration in which each agent is guaranteed to have visited all network nodes and has returned to its starting location. It is a different objective than that in previous chapters where agents were supposed to explore the graph collaboratively.

In this chapter we will consider two scenarios. We first consider the scenario when each mobile agent knows the map of the network, as well as its own initial position. In the second scenario, the network is unknown to the agents. In both scenarios we propose algorithms for collision-free exploration of trees and general graphs. The results presented in this chapter were published in [T1].

We want to study the graph exploration problem in which two agents may never visit the same node of the graph at the same time. This property of the model, which we call *collision avoidance*, is motivated by the fact that the processes executed by mobile agents (software agents or physical robots) sometimes require exclusive access to network resources. Such a setting can be motivated, for example by applications related to mobile software agents which may need exclusive access to a node's resources when updating its data. In another application area, robots (or nano-robots) distributing interacting chemical or pharmacological agents within a battlefield or a human body must avoid being simultaneously present at a small distance from each other. Likewise, individuals, one of which is highly infectious or socially conflicting should avoid a meeting.

In our considerations, time is divided into synchronous rounds. Initially, each agent is placed at a different node and in each round it may choose to move to a neighboring node or to stay motionless. The agents are independent in the sense that they cannot communicate and none of them knows the number of other agents, their initial placement in the graph, nor is aware of the current location of the other agents. The agents move independently, and each of them executes the same algorithm. The effectiveness of the

algorithm is measured in terms of the *collision-free exploration time*, i.e., the number of rounds until all potentially existing agents are certain to have completed the exploration and returned to their initial location. Details of our model are discussed in Section 4.1.

The only results on exactly the same problem are due to Herman and Masuzawa [98], who obtained an asymptotically tight bound for trees (with a priori known topology). Herein, we provide our own analysis of this case for the sake of completeness. Our analysis for trees allows us to derive an exact number of required steps.

The offline setting of our question is related to the following problem (cf. [8]), which was studied in the context of routing. Each vertex of a given graph is initially occupied by a "pebble", which has to be moved to a destination, so that the destinations of different pebbles are different. In every synchronous round a set of edges is selected and the pebbles at each edge endpoints are interchanged. [8] attempts to minimize the number of rounds so that all pebbles reach their destination, giving lower and upper bounds for different classes of graphs. The routing model of [8] inherently implies the usage of matchings - the technique that we choose to apply in some results of this chapter. The $3n$ upper bound for trees given in [8] was improved to $\frac{1}{2}n + O(\log n)$ in [145]. [117] and [132] independently extended this model to allow more than one pebble per origin and destination node. Although the matching model of routing was also considered in the online setting (e.g. [132]), this is unrelated to this chapter. Indeed, in online routing the distributed decisions are made by the network nodes on the basis of local information concerning incoming packets, while in exploration, the mobile agents determine their subsequent moves while learning a piece of information about the network structure.

We consider two scenarios, differing in the amount of global information about the network topology which is available to each agent. Our results are summarized in Table 4.1.

For the first scenario, considered in Section 4.2, we assume that a map of the network is *a priori* known to the agents. We show that a collision-free exploration strategy exists for any graph, and provide efficient solutions for trees and general graphs. We start by considering the case of trees, proposing a strategy which involves the simultaneous activation of agents located at the endpoints forming a matching in some optimal edge-coloring of the tree. This strategy is shown to yield optimal exploration time.

We then extend this approach from the case of trees to the case of general graphs, by requiring that the agents perform exploration using only the edges of a well-chosen spanning tree of the graph. Somewhat surprisingly, it turns out that this approach is asymptotically the best possible, i.e., within a constant factor of the optimum. To prove the corresponding lower bound on the collision-free exploration time in graphs, we establish a tight connection between our problem and the fractional relaxation of the LP formulation of the minimum-degree spanning tree problem.

In the second scenario, discussed in Section 4.3, we deal with synchronous agents possessing only local knowledge about the graph to explore. In particular, no knowledge of the size of the graph is assumed. We suppose that each agent executes a local, distributed

Scenario	*Tree*	*General graph*
With complete map:	$n\Delta(G)$ Thm. 4.4	$\Theta(n\Delta^*(G))$ Thm. 4.6
With local knowledge:	$O(n^2)$ Thm. 4.12	$O(n^5 \log n)$ Thm. 4.13

TABLE 4.1: The time of optimal collision-free graph exploration. $\Delta(G)$ denotes the maximum degree of a node in graph G, and $\Delta^*(G) = \Delta(T)$, where T is a minimum-degree spanning tree of G.

algorithm, in every round making a decision based on the information concerning the currently occupied node and the identifiers of the neighboring nodes. For this scenario, we show that a collision-free exploration is always feasible in finite time and we give algorithms for trees and general graphs. Our collision-free exploration strategies are of length $O(n^2)$ for trees and $O(n^5 \log n)$ for arbitrary graphs, and make use of the application of universal exploration sequences. Throughout the chapter, we assume that the strategies for collision-free exploration are required to return the agents to their initial location. This assumption allows us to see our strategies as an analogue of the classical Travelling Salesman Problem with mutually-exclusive salesmen on an unweighted graph, and also allows the agents to engage in perpetual (periodic) exploration of the graph. After minor modification of the proofs, all the results presented in Table 4.1 also hold up to constant factors for the variant of the problem in which agents may end exploration at an arbitrary node of the graph.

The problem of graph exploration without collisions was also studied in the case when two agents also collide when traversing one edge in opposite directions. In that case exploration is not always possible. In [14] the authors study the maximal number of agents that can explore graph without collisions in synchronous setting. The asynchronous Look-Compute-Move model is considered in [21] where the authors study the maximal and minimal number of agents that are necessary and sufficient to solve the problem for a ring. In both these papers it is assumed that each agent can observe (or compute) the positions of the other agents. Collision avoidance in the Look-Compute-Move was also studied in the context of graph searching [20].

4.1 Model and definitions

We assume that the nodes of each n-node network have unique identifiers in $\{1, \ldots, n\}$. The identifier of a node v is denoted by $id(v)$. Several agents are initially located at pairwise different nodes of the network. The initial position of each agent λ is denoted by $home(\lambda)$. Each agent is unaware of the number and initial positions of the other agents, and all agents are given the same algorithm that determines their behavior in the subsequent rounds.

Each agent can perceive the identifier $id(v)$ of the currently occupied node v and can perceive the identifiers of all neighbors of v. Moreover, the agent can distinguish the edges incident to v according to the identifiers of the nodes located at the endpoints of the edges. The latter assumption is necessary to properly perform the navigation in a node labeled network.

The agents are synchronous and hence the time is divided into rounds of equal duration. Each round is divided into two stages. In the first stage each agent λ makes a decision (by executing its algorithm) that determines its behavior in the second stage of the round. The decision can be three-fold: it may decide to stay in this particular round at the currently occupied node, to move from the currently occupied node to one of its neighbors, or decide that its exploration is completed. In the second stage of the round, all agents simultaneously perform the action corresponding to their decision. If, as a result, two agents located at some adjacent nodes u and v decide to move from u to v and from v to u, respectively, then they traverse the same edge in this round, but remain unaware of this event, i.e., the two agents do not communicate and do not perceive each other. We require that the algorithm given to the agents ensures the following:

- at the end of each round no two agents are present on the same node of the network,
- by the end of some round $t \geq 0$, all the agents have decided that the exploration is completed,
- each agent has visited each node of the network in one of the rounds $1, \ldots, t$,
- each agent λ is present at $home(\lambda)$ at the end of round t.

Note that, in this setting, the execution of the agent's algorithm (and thus the behavior of the agent) only depends on the input to the algorithm and on the identifiers of the nodes visited by the agent. Thus, in particular, an agent is unable to ever discover the initial or current position of any other agent or the number of agents in the network.

With respect to additional information available to the agents, we study two scenarios in this work: either the agents have no prior knowledge of network topology and no knowledge of global parameters, or the complete map of the network is given to all agents. In the latter case the map consists of node identifiers, but provides no information on the locations of other agents. Note that if, together with a complete map of the network, all agents receive as an input information on the initial positions of all agents, then our exploration problem becomes similar to the off-line routing problems considered e.g. in [8, 132, 145].

Let us introduce the notation used in this chapter. Let $G = (V(G), E(G))$ be any graph. For any node v of G let $\Gamma_G(v)$ be the set of neighbors of v. We use the symbol $\Delta(G)$ to denote the *degree* of G, defined as $\Delta(G) = \max\{|\Gamma_G(v)| \colon v \in V(G)\}$. ($|\Gamma_G(v)|$ is called the *degree* of v.) Given a set of edges $X \subseteq E(G)$, define $G[X]$ to be the network with nodes in $V(G)$ and edges in X, $G[X] = (V(G), X)$. Note that $G[X]$ is not necessarily connected. A connected network H such that $V(H) \subseteq V(G)$ and

$E(H) \subseteq E(G)$ is called a *connected component* of G if there exists no connected network H' such that $V(H) \subseteq V(H') \subseteq V(G)$ and $E(H) \subseteq E(H') \subseteq E(G)$ and $H \neq H'$.

Any sequence $\mathcal{R} = (v_0, v_1, \ldots, v_l)$ of nodes of a network G is called a *route in* G if $v_i = v_{i-1}$ or $\{v_i, v_{i-1}\}$ is an edge of G for each $i = 1, \ldots, l$. We say that l is the *length* of $\mathcal{R}$ and we write $\mathcal{R}_i = v_i$ for each $i = 0, \ldots, l$. The route $\mathcal{R}$ *covers* G if for each node v of G there exists $i \in \{0, \ldots, l\}$ such that $v = \mathcal{R}_i$. The route $\mathcal{R}$ is *closed* if $\mathcal{R}_0 = \mathcal{R}_l$, where l is the length of $\mathcal{R}$. Let λ be an agent. We say that the route $\mathcal{R}$ of length l is a *route of* λ if: (i) $\mathcal{R}_0 = \mathit{home}(\lambda)$ and λ is present at $\mathcal{R}_i$ at the end of round i, $i = 1, \ldots, l$, and (ii) λ does not move in any round $r > l$.

We say that a route $\mathcal{R}$ of length l is an *exploration strategy for* λ if (i) $\mathcal{R}$ is a route of λ, (ii) $\mathcal{R}$ is closed, (iii) $\mathcal{R}$ covers G. Two routes $\mathcal{R}$ and $\mathcal{R}'$ of length l are *collision-free* if $\mathcal{R}_i \neq \mathcal{R}'_i$ for each $i = 0, \ldots, l$. Let $\mathcal{A} = \{\lambda_1, \ldots, \lambda_k\}$, $1 \leq k \leq n$, be the set of agents that are initially located at the nodes of G. Let $\mathcal{R}(\lambda)$ be the exploration strategy for each agent $\lambda \in \mathcal{A}$. We say that $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_k)$ are *collision-free* if $\mathcal{R}(\lambda_i)$ and $\mathcal{R}(\lambda_j)$ are collision-free for each $i, j \in \{1, \ldots, k\}$, $i \neq j$. Let t be the minimum integer such that for each set of agents placed arbitrarily on the nodes of G there exist collision-free exploration strategies, each of length at most t, for the agents. Then, t is called the *collision-free exploration time of* G.

4.2 Network exploration with a map

In this section we consider the problem of collision-free exploration in the case when each agent is given a complete map of the network to be explored. We start by discussing the simpler case of tree networks and then we generalize our approach from trees to arbitrary networks, showing its asymptotic optimality by proving a corresponding lower bound.

We start with some additional notation and two lemmas that are the main tool in the analysis of the algorithm given in this section.

Given a tree network T, we say that a function $c\colon E(T) \to \{1, \ldots, d\}$ is a *d-edge-coloring* of T if $c(e) \neq c(e')$ for any two adjacent edges in T.

Let d be an integer, let c be a d-edge-coloring of G, and let v be any node of G. Define $\mathcal{T}(v, d, c) = (v_0, v_1, v_2, \ldots)$ to be an infinite route in G starting at v such that:

(i) if $c(\{v_{i-1}, u\}) \neq 1 + (i-1) \bmod d$ for each neighbor u of v_{i-1} in G, then $v_i = v_{i-1}$,

(ii) if $c(\{v_{i-1}, u\}) = 1 + (i-1) \bmod d$ for some neighbor u of v_{i-1}, then $v_i = u$.

Then, define $\mathcal{T}^l(v, d, c)$, $l \geq 0$, to be the prefix of $\mathcal{T}(v, d, c)$ of length l, and $\mathcal{T}^l_i(v, d, c)$ to be v_i for each $i = 0, \ldots, l$.

(See Figure 4.1 for an example of $\mathcal{T}^{dn}(v, d, c)$ computed for a tree network T on $n = 15$ nodes. Figure 4.1(a) gives T and a 6-edge-coloring c of T, thus, $d = 6$ in this example.)

We now give two lemmas in which we prove that if u and v are two distinct nodes of T, then the routes $\mathcal{T}^{dn}(u, d, c)$ and $\mathcal{T}^{dn}(v, d, c)$ are collision-free, and each of them is closed and covers the tree network.

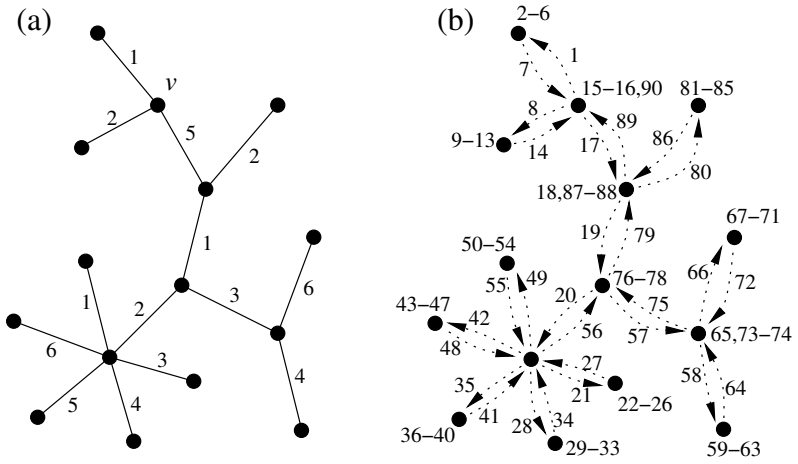

FIGURE 4.1: (a) a 15-node tree network T with a 6-edge-coloring c; (b) the route $\mathcal{T}^{dn}(v,d,c) = \mathcal{T}^{90}(v,6,c) = (v_0,\ldots,v_{90})$ encoded as follows: an integer i that is a label of an arc going from u to v indicates that $v_i = v$ and $v_{i-1} = u$; an integer label i of a node u of T means that $v_i = v_{i-1} = u$.

Lemma 4.1. *Let T be a tree network. If c is a d-edge-coloring of T, then for any two distinct nodes u and v of T the routes $\mathcal{T}^l(u,d,c)$ and $\mathcal{T}^l(v,d,c)$ are collision-free for each $l \geq 0$.*

Proof. Let $l \geq 0$ be fixed arbitrarily. Denote $\mathcal{T}^l(u,d,c) = (u_0,u_1,\ldots,u_l)$ and $\mathcal{T}^l(v,d,c) = (v_0,v_1,\ldots,v_l)$.

We prove by induction on $i = 0,\ldots,l$ that $\mathcal{T}^i(u,d,c)$ and $\mathcal{T}^i(v,d,c)$ are collision-free. By assumption, $u \neq v$ and by definition, $\mathcal{T}^0(u,d,c) = (u)$ and $\mathcal{T}^0(v,d,c) = (v)$. Hence, the claim follows for $i = 0$.

Assume that the claim holds for some $i \in \{0,\ldots,l-1\}$ and we prove it for $i+1$. If $u_i = u_{i+1}$ and $v_i = v_{i+1}$, then the proof is completed. Thus, $u_i \neq u_{i+1}$ or $v_i \neq v_{i+1}$ and assume without loss of generality that the former occurs. By construction of $\mathcal{T}(u,d,c)$, $c(\{u_i,u_{i+1}\}) = 1 + (i \bmod d)$. If $u_{i+1} = v_i$, then from the construction of $\mathcal{T}(v,d,c)$ we obtain that $v_{i+1} = u_i$ and consequently $u_{i+1} \neq v_{i+1}$ as required. If $u_{i+1} \neq v_i$, then $v_{i+1} \neq u_{i+1}$, because $u_i \neq v_i$ and c is an edge-coloring of T. □

Lemma 4.2. *Let T be a tree network and let d be an integer. If c is a d-edge-coloring of T and v is a node of T, then the route $\mathcal{T}^{dn}(v,d,c)$ is closed and covers T.*

Proof. Consider T to be rooted at v and let T_u to be a subtree of T induced by u and all its descendants for each $u \in V(T)$. Assume without loss of generality that T consists of at least two nodes. Denote $\mathcal{T}^{dn}(v,d,c) = (v_0,v_1,\ldots,v_{dn})$.

We prove by induction on the subtree size that for each $u \in V(T)$, if $i \in \{0,\ldots,dn\}$ is the minimum index such that $v_i = u$, then $(v_i,\ldots,v_{i+s})$ is closed and covers T_u, where $s = |V(T_u)|d - 1$.

We first consider the case of a single node tree T_u. Since $|V(T)| > 1$, u has a parent u'. By the definition of $\mathcal{T}(v,d,c)$, $c(\{u,u'\}) = 1 + (i-1) \bmod d$. Let $s \geq 0$ be the maximum index such that $v_i = v_{i+1} = \cdots = v_{i+s}$. Clearly, $v_{i+s+1} = u'$. We obtain that $s \geq d-1$,

because otherwise $1 + (i + s) \bmod d = c(\{v_{i+s}, v_{i+s+1}\}) = c(\{u, u'\}) = 1 + (i - 1) \bmod d$, and hence $(s + 1) \bmod d = 0$, which is a contradiction. Moreover, $v_{i+d} = u'$, because $1 + (i + d - 1) \bmod d = 1 + (i - 1) \bmod d$. Hence, $s = d - 1$ as required.

Consider any rooted subtree T_u, $|V(T_u)| > 1$, and suppose that the claim holds for any subtree with less than $|V(T_u)|$ nodes. We first consider the case when $u \neq v$.

Let $u_1, \ldots, u_p$, $p \geq 1$, be the children of u in T_u. By the definition of edge-coloring, $d \geq \Delta(T)$, and hence $p \leq d$. Let u' be the parent of u in T, and assume without loss of generality that for some $1 \leq q \leq p$

$$c(\{u, u'\}) < c(\{u, u_1\}) < \cdots < c(\{u, u_q\}) \text{ and } c(\{u, u_{q+1}\}) < \cdots < c(\{u, u_p\}) < c(\{u, u'\}). \tag{4.1}$$

Let i_j be the minimum index such that $v_{i_j} = u_j$, $j = 1, \ldots, p$. By the induction hypothesis, $\mathcal{R}(u_j) = (v_{i_j}, \ldots, v_{i_j+s_j})$ is closed and covers T_{u_j}, where $s_j = |V(T_{u_j})|d - 1$, for each $j = 1, \ldots, p$.

First we prove that for each $j = 1, \ldots, p$ it holds $v_{i_j+s_j+1} = u$. Let $j \in \{1, \ldots, p\}$ be selected arbitrarily. By the choice of i_j, $v_{i_j} \neq v_{i_j-1}$. Since v_0 is the root of T, $u = v_{i_j-1}$. This implies that

$$c(\{u, u_j\}) = c(\{v_{i_j-1}, v_{i_j}\}) = 1 + (i_j - 1) \bmod d. \tag{4.2}$$

Note that

$$1 + (i_j + s_j) \bmod d = 1 + (i_j + |V(T_{u_j})|d - 1) \bmod d = 1 + (i_j - 1) \bmod d. \tag{4.3}$$

By the fact that $\mathcal{R}(u_j)$ is closed, $u_j = v_{i_j+s_j}$. Hence, by (4.2) and (4.3),

$$c(\{u_j, u\}) = c(\{v_{i_j+s_j}, v_{i_j+s_j+1}\}) = c(\{u_j, v_{i_j+s_j+1}\}).$$

Since, c is an edge-coloring of T, $v_{i_j+s_j+1} = u$ as required.

Let $\mathcal{C}(u_j)$ be the maximal subsequence of $\mathcal{T}^{dn}(v, d, c)$ starting with $v_{i_j+s_j+1}$ and with all elements equal u, $j = 1, \ldots, p$. By (4.1) and by construction of $\mathcal{T}(v, d, c)$,

$$\mathcal{C}(u_j) = (v_{i_j+s_j+1}, \ldots, v_{i_{j+1}-1}) \text{ for each } j = 1, \ldots, p - 1. \tag{4.4}$$

Define $\mathcal{C}(u_0)$ and $\mathcal{C}(u_p)$ to be the maximal subsequences of $\mathcal{T}^{dn}(v, d, c)$ starting with v_i and v_{i_p} respectively, with all elements equal u. Note that the definition of $\mathcal{C}(u_0)$ is correct, because $v_i = u$. By (4.4),

$$(v_i, \ldots, v_{i+s}) = \mathcal{C}(u_0), \mathcal{R}(u_1), \mathcal{C}(u_1), \mathcal{R}(u_2), \mathcal{C}(u_2), \ldots, \mathcal{R}(u_p), \mathcal{C}(u_p),$$

where

$$s = \sum_{j=0}^{p} |\mathcal{C}(u_j)| + \sum_{j=1}^{p} |\mathcal{R}(u_j)| - 1 = 2p + \sum_{j=0}^{p} (|\mathcal{C}(u_j)| - 1) + \sum_{j=1}^{p} (|V(T_{u_j})|d - 1). \tag{4.5}$$

The sum $\sum_{j=0}^{p}(|\mathcal{C}(u_j)|-1)$ equals, informally speaking, the number of two consecutive appearances of u in $(v_{i+1},\ldots,v_{i+s})$. By (4.1), this sum equals $d-p-1$, because c is a d-edge-coloring of T and u is not the root of T. Hence, by (4.5),

$$s = d-1+\sum_{j=1}^{p}|V(T_{u_j})|d = d(1+\sum_{j=1}^{p}|V(T_{u_j})|)-1 = |V(T_u)|d-1.$$

Finally, if $u=v$, then the proof is analogous, and the fact that u has no parent implies that $\sum_{j=0}^{p}(|\mathcal{C}(u_j)|-1) = d-p$. Hence, we obtain that $s=dn$ when u is the root, which completes the proof. □

It remains to observe that the considered routes can be implemented as exploration strategies. Indeed, each agent λ is able to construct some d-edge-coloring c of T (the same for all agents, e.g., lexicographically first with respect to some chosen ordering of all colorings) with $d=\Delta(T)$, and hence it is able to 'follow' $\mathcal{T}^{n\Delta(T)}(home(\lambda),\Delta(T),c)$. We formulate this strategy in the form of the algorithm below.

Algorithm Tree-Exploration(T)
Input: A node-labeled tree network T.
begin
Let v be the initial position of the executing agent.
Compute the lexicographically first $\Delta(T)$-edge-coloring c of T
for each round $r \leftarrow 1$ **to** $n\Delta(T)$ **do**
if there exists an edge $\{v,u\}$ such that $c(\{v,u\}) = 1+(r-1) \bmod \Delta(T)$
then move from v to u in round r, set $v \leftarrow u$.
else stay at v in round r.
end Tree-Exploration

For an agent λ following Algorithm Tree-Exploration, its route is of length $n\Delta(T)$, and given as $\mathcal{R}^{n\Delta(T)}(\lambda) = \mathcal{T}^{n\Delta(T)}(home(\lambda),\Delta(T),c)$, where c is the $\Delta(T)$-edge-coloring computed in the Algorithm. Consequently, taking into account Lemmas 4.1 and 4.2, we have the following.

Proposition 4.3. *Let T be a tree network and let $\lambda_1,\ldots,\lambda_k$, $1\le k\le n$, be the agents initially located at pairwise different nodes of T. Suppose that the agent λ_i uses Algorithm* Tree-Exploration *to compute its route $\mathcal{R}^{n\Delta(T)}(\lambda_i)$, for each $i=1,\ldots,k$. Then, $\mathcal{R}^{n\Delta(T)}(\lambda_1),\ldots,\mathcal{R}^{n\Delta(T)}(\lambda_k)$ are exploration strategies, and are collision-free.*

It turns out that there exist no shorter collision-free exploration strategies than those constructed with Algorithm Tree-Exploration.

Theorem 4.4. *The collision-free exploration time of any n-node tree network T is precisely equal to $n\Delta(T)$.*

Proof. The upper bound follows from Proposition 4.3. Now, we prove the lower bound, i.e., that the collision-free exploration time of T is at least $n\Delta(T)$. Let u be a fixed node of degree $\Delta(T)$ in T. First assume that there are n agents in T. We say that an agent λ is *active* in round r if λ goes from v to u in round r for some $v \in \Gamma_T(u)$. In each round at most one agent is active. For each agent λ there exist at least $\Delta(T)$ rounds in which λ is active, because the route of λ needs to be closed and T is a tree. Since there are n agents in total, we obtain that there are at least $n\Delta(T)$ rounds in which an agent is active. This proves that there exists an agent λ that is active in round $n\Delta(T)$, and hence its exploration strategy is of length at least $n\Delta(T)$. Finally, observe that λ constructs the same route regardless of the number of agents present in the network. This is due to the fact that T and $id(home(\lambda))$ is the entire input to the algorithm that λ executes. □

We finish this section by remarking on the complexity of Algorithm Tree-Exploration. For any tree network T on n nodes, there exists a $\Delta(T)$-edge-coloring of T and it can be computed in $O(n)$-time. Consequently, the total time of an agent's local computations when running Algorithm Tree-Exploration is $O(n\Delta(T))$.

We say that T is a *spanning tree* of G if T is a tree such that $V(T) = V(G)$ and $E(T) \subseteq E(G)$. Then, T is a *minimum degree spanning tree* of G if T is a spanning tree of G and the degree of T is minimum over the degrees of all spanning trees of G. Define $\Delta^*(G) = \Delta(T)$, where T is a minimum degree spanning tree of G. We propose the following solution to the collision-free exploration problem.

Algorithm Network-Exploration(G)
Input: A node-labeled network G.
begin
Compute the lexicographically first minimum-degree spanning tree T^* of G.
Call Algorithm Tree-Exploration(T^*).
end Network-Exploration

The next proposition follows from the formulation of Algorithm Network-Exploration and from Proposition 4.3.

Proposition 4.5. *Let G be a network and let $\lambda_1, \ldots, \lambda_k$, $1 \leq k \leq n$, be the agents initially located at pairwise different nodes of G. Suppose that the agent λ_i uses Algorithm* Network-Exploration *to compute its route $\mathcal{R}(\lambda)$, $i = 1, \ldots, k$. Then, $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_k)$ are collision-free exploration strategies of length $n\Delta^*(G)$.*

Now, the following theorem implies that our result is asymptotically tight, i.e., it implies that Algorithm Network-Exploration constructs exploration strategies whose length is within a constant factor from the optimum.

Theorem 4.6. *The collision-free exploration time of any network G is $\Theta(n\Delta^*(G))$.*

Proof. The fact that the collision-free exploration time of G is $O(n\Delta^*(G))$ follows from Proposition 4.5.

Now, we prove the lower bound of $\Omega(n\Delta^*(G))$. Observe that if $\Delta^*(G) \leq 3$, then the theorem follows, because each exploration strategy must be of length $\Omega(n)$. To finish the proof, suppose that there exist exploration strategies for the agents, such that the length of each exploration strategy is at most $n(\Delta^*(G) - 3)/2$.

For each node v of G let $E_v = \{\{v, u\} \colon u \in \Gamma_G(v)\}$. Consider the following linear program (LP) with variables $(x_e \colon e \in E(G))$, which satisfies the following set of constraints [69, 97]:

$$\sum_{e \in E(G)} x_e = n - 1 \tag{4.6}$$

$$\sum_{e \in E(G[S])} x_e \leq |S| - 1, \quad \text{for each } S \subseteq V(G) \tag{4.7}$$

$$\sum_{e \in E_v} x_e \leq t, \quad \text{for each } v \in V(G) \tag{4.8}$$

$$0 \leq x_e \leq 1, \tag{4.9}$$

where t is an integer and n is the number of nodes of G. Any solution to the above problem is called a *fractional spanning tree* of *degree* t of G. Informally speaking, if $(x_e \colon e \in E(G))$, is a solution to (4.6)-(4.9), then x_e is the 'fraction' of the edge e that is included in the resulting fractional spanning tree. Note that any integer solution, i.e. the one in which $x_e \in \{0, 1\}$ for each $e \in E(G)$, is a spanning tree of degree at most t of G.

Suppose that n agents $\lambda_1, \ldots, \lambda_n$ are present in the network G. Let $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_n)$ be some collision-free exploration strategies for the agents. Suppose that the length of each exploration strategy is at most $nt/2$. Based on these exploration strategies, we now construct a solution to the LP in (4.6)-(4.9). For each $i = 1, \ldots, n$, let T_i be any spanning tree of G such that if $e \in E(T_i)$, then there exists a round r such that $e = \{\mathcal{R}_{r-1}(\lambda_i), \mathcal{R}_r(\lambda_i)\}$ (in other words, λ_i traverses e in some round). Such a T_i exists, because $\mathcal{R}(\lambda_i)$ covers G, $i = 1, \ldots, n$. Define:

$$f_i(e) = \begin{cases} 1/n, & \text{if } e \in E(T_i) \\ 0, & \text{if } e \notin E(T_i) \end{cases} \qquad \text{and} \qquad x_e = \sum_{i=1}^{n} f_i(e) \text{ for each } e \in E(G). \tag{4.10}$$

Now, we prove that x_e's defined in (4.10) form a solution to the LP in (4.6)-(4.9).

First note that $\sum_{e \in E(G)} f_i(e) = (n-1)/n$ for each $i = 1, \ldots, n$, because f_i assigns $1/n$ to exactly $n - 1$ edges of G, which follows from the fact that T_i is a spanning tree of G, $i = 1, \ldots, n$. Thus, (4.6) holds.

Now, let $S \subseteq V(G)$ be selected arbitrarily. For each $i = 1, \ldots, n$, $|E(T_i) \cap E(G[S])| = |E(T_i[S])| \leq |S| - 1$, because $T_i[S]$ is, by definition, a collection of node-disjoint trees on set S. Hence, (4.7) follows.

Let v be any node of G and let $X = E_v \cap (E(T_1) \cup \cdots \cup E(T_n))$. For each r there exist at most two edges in X traversed by an agent in round r. Hence,

$$\sum_{e \in X} \sum_{i=1}^{n} f_i(e) \leq \frac{nt}{2} \cdot \frac{2}{n} = t.$$

Note that if $e \in E_v \setminus X$, then $\sum_{i=1}^{n} x_e = 0$. This proves that (4.8) holds.

Finally, (4.9) follows directly from (4.10).

We have proved that the existence of exploration strategies of length $nt/2$ implies the existence of a solution to (4.6)-(4.9). Moreover, we have the following.

Claim ([97]). *If there exists a solution to* (4.6)-(4.9)*, then there exists an integer solution to (4.6),(4.7),(4.9) with the additional constraint*

$$\sum_{e \in E_v} x_e \leq t + 2 \text{ for each } v \in V(G)$$

which replaces (4.8).

We remark that such an integer solution defines a spanning tree of G, given by the set of edges $\{e \in E(G) \colon x_e = 1\}$.

In view of the definition of x_e's in (4.10), it follows that if there exist exploration strategies of length at most $nt/2$ for the n agents, then there exists a spanning tree T^* of G, and the degree of T^* is at most $t+2$. By assumption, there exist in G exploration strategies of length at most $n(\Delta^*(G)-3)/2$, hence, putting $t = \Delta^*(G)-3$, it follows that G has a spanning tree of degree at most $\Delta^*(G) - 1$, a contradiction with the definition of $\Delta^*(G)$. □

We finish this section with a complexity remark. Finding a minimum-degree spanning tree is in general an NP-hard problem. We can, however, modify the approach to obtain an exploration strategy of length $n(\Delta^*(G) + 1)$ that can be computed efficiently. We make use of a $O(mn\alpha(m,n)\log n)$-time algorithm that for a given G finds its spanning tree T of degree $\Delta(T) \leq \Delta^*(G)+1$, where m and n are, respectively, the number of edges and nodes of G, and α is the inverse Ackermann function [92]. By using the tree T in Algorithm Network-Exploration instead of T^* we obtain an exploration strategy of length $n(\Delta^*(G) + 1)$ for agent λ, and this strategy is computed in time $O(mn\alpha(m,n)\log n)$. On the other hand, computing the precise value of collision-free exploration time is a hard problem.

Proposition 4.7. *The problem of deciding, for a given network G and integer l, whether the collision-free exploration time of G is at most l, is* NP*-complete.*

Proof. We prove that the problem is NP-complete already for the special case of $l = n$, where n is the number of nodes of G. The proof is by reduction from the Hamiltonian cycle problem [93]. We argue that there exists a Hamiltonian cycle of G if and only if the collision free exploration time of G is n.

If such a Hamiltonian cycle $C = v_1\text{-}v_2\text{-}\cdots\text{-}v_n$ exists, then one can construct an exploration strategy $\mathcal{R}(\lambda)$ for an agent λ with $home(\lambda) = v_i$ by taking $\mathcal{R}(\lambda) =$

$(v_i, v_{i+1}, \ldots, v_n, v_1, \ldots, v_{i-1}, v_i)$. Clearly, $\mathcal{R}(\lambda)$ is a route of length n in G, because C is a cycle. Also, $\mathcal{R}(\lambda)$ is closed and covers G. Moreover, if for another agent λ' we have $home(\lambda') \neq v_i$, then $\mathcal{R}(\lambda)$ and $\mathcal{R}(\lambda')$ are collision-free.

Now suppose that the collision-free exploration time of G equals n. Let λ be any agent initially occupying any node of G and take an exploration strategy $\mathcal{R}(\lambda)$ of length n for λ. Since $\mathcal{R}(\lambda)$ covers G, $\mathcal{R}_i(\lambda) \neq \mathcal{R}_{i-1}(\lambda)$ for each $i = 1, \ldots, n$. By the fact that $\mathcal{R}(\lambda)$ is closed, $\mathcal{R}_0(\lambda) = \mathcal{R}_n(\lambda)$. Hence, $\mathcal{R}(\lambda)$ is a cycle of length n in G as required. □

4.3 Local network exploration

In this section we consider the problem of collision-free exploration in the setting when the agents do not receive any information about the network in which they operate. Recall that we assume, that each node $v \in V$ is equipped with a unique identifier $id(v) \in \{1, 2, \ldots, n\}$, and each agent located at v is only aware of the identifier $id(v)$ and the identifiers of the neighbors of v at the endpoints of respective edges incident to v. In Section 4.3.1 we consider tree networks, and in Section 4.3.2 we show how any network can be explored.

Let G be any network. For the purposes of this section we introduce an edge-labeling function id' defined as

$$id'(\{u, v\}) = id(u) + id(v) \text{ for each } \{u, v\} \in E(G). \tag{4.11}$$

We recall without proof the following essential property of function id'.

Lemma 4.8. *Let G be any n-node network. Then, id' is a $2n$-edge-coloring of G.*

4.3.1 Local exploration of tree networks

In this section we provide an algorithm which defines collision-free routes of agents, and is guaranteed to perform exploration if the explored network is a tree. For any integer $b \geq 2$ define the following sequence of integers $U(b) = (1, \ldots, 2b, \quad \ldots, \quad 1, \ldots, 2b)$, where $1, \ldots, 2b$ is repeated b times, and let $U_i(b)$, $i \in \{1, \ldots, 2b^2\}$, be its i-th element.

Algorithm Local-Tree-Exploration
begin
Let v be the initial position of the executing agent.
$b \leftarrow 2$
$r \leftarrow 0$
while not all nodes have been visited so far **do** {start a new phase}
for $s \leftarrow 1$ **to** $|U(b)|$ in round $r + s$ **do**
if there exists an edge $\{v, u\}$ such that $id(u) \leq b$
and $id(v) \leq b$ and $id'(\{v, u\}) = U_s(b)$
then move from v to u {in round $r + s$}; set $v \leftarrow u$.
else stay at v. {in round $r + s$}
end for
$r \leftarrow r + |U(b)|$
$b \leftarrow 2b$
end while
Backtrack all previous moves, i.e., λ moves from v to u in round $r + i$ if and only if λ moved from u to v in round $r - i + 1$ for each $i = 1, \ldots, r$.
end Local-Tree-Exploration

Define *phase* p, $p \geq 1$, as the sequence of rounds $(1 + \sum_{j=1}^{p-1} |U(2^j)|, \ldots, \sum_{j=1}^{p} |U(2^j)|)$ and denote by $\ell(p) = |U(2^p)|$ the number of rounds of phase p. Note that

$$\ell(p) = 2^{2p+1} \text{ for each } p \geq 1, \tag{4.12}$$

and that phase p consists of the rounds in which the behavior of any agent λ is determined in the p-th iteration of the 'while' loop of its execution of Algorithm Local-Tree-Exploration, whenever p does not exceed the total number of iterations executed. Denote by $\mathcal{R}(\lambda, p)$ the route of an agent λ restricted to its moves in phase p, $p \geq 1$.

We denote by T_p the subgraph of T induced by all edges e whose endpoints have identifiers at most 2^p, $T_p = T[\{\{u, v\} \in E(T) \colon id(u) \leq 2^p \wedge id(v) \leq 2^p\}]$.

Finally, define $\ell = 2 \sum_{p=1}^{\lceil \log_2 n \rceil} \ell(p)$.

We now prove that each agent λ moves in phase p 'inside' the connected component T' of T_p that contains the vertex occupied by λ at the beginning of phase p.

Lemma 4.9. *Let $p \geq 1$ be an integer, let T be a tree network and let λ be an agent. Let v be the vertex occupied by λ at the beginning of phase p. Then, $\mathcal{R}(\lambda, p)$ is a route in the connected component of T_p that contains v, and $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(v, 2^{p+1}, id')$.*

Proof. First we argue that $\mathcal{R}(\lambda, p)$ is a route in the connected component T' of T_p that contains the node v. The agent λ performs its moves in phase p as a result of the execution of the p-th iteration of the 'while' loop of Algorithm Local-Tree-Exploration. The value of the variable b in this p-th iteration equals 2^p. Hence, if λ decides to move from a node v to a node u in some round of phase p, then $id(u) \leq 2^p$ and $id(v) \leq 2^p$. Thus, $\{u, v\}$ is an edge of T_p, and therefore $\{u, v\} \in E(T')$.

To conclude that $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(v, 2^{p+1}, id')$, note that, by Lemma 4.8, id' restricted to T' is a 2^{p+1}-edge-coloring of T'. Thus, the lemma follows from the definition of $\mathcal{T}$ and from the formulation of Algorithm Local-Tree-Exploration. □

Note that the length of the route $\mathcal{R}(\lambda, p)$ of λ in phase p is bounded by $\ell(p)$, hence is, in general, 'unrelated' to the number of nodes of T'. For this reason, T' need not be completely explored. However, by the definition of T_p, we have that $T_p = T$ (and $T' = T$) if and only if $p \geq \lceil \log_2 n \rceil$. We use this observation to show that all agents perform backtracking and stop after exactly the same phase $p = \lceil \log_2 n \rceil$, and that in this phase each of them visits all nodes of T.

Lemma 4.10. *Let T be a n-node tree network. For each agent λ the number of iterations of the 'while' loop of Algorithm* Local-Tree-Exploration *executed by λ equals $\lceil \log_2 n \rceil$. Moreover, $\mathcal{R}(\lambda, \lceil \log_2 n \rceil)$ covers T.*

Proof. If $p \in \{1, \ldots, \lceil \log_2 n \rceil - 1\}$, then $T_p \neq T$. Hence, T_p is not connected and, by Lemma 4.9, $\mathcal{R}(\lambda, p)$ does not cover T. The agent λ determines this fact, e.g., by recording, in a set X, the identifiers of all nodes adjacent to the nodes of its route in phase p, $\mathcal{R}(\lambda, p)$. Then, due to the connectedness of T, X contains an identifier such that the corresponding node is not in $\mathcal{R}(\lambda, p)$ and consequently λ starts executing the $(p+1)$-st iteration of the 'while' loop of Algorithm Local-Tree-Exploration.

Now, let $p = \lceil \log_2 n \rceil$, and so $T_p = T$. Due to Lemma 4.9, $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(u, 2^{p+1}, id')$, where u is the node occupied by λ at the beginning of phase p. By the formulation of Algorithm Local-Tree-Exploration and by (4.12), $\ell(p) = 2^{2p+1} \geq 2n^2 \geq 2n\Delta(T)$. By Lemma 4.2, $\mathcal{R}(\lambda, p)$ covers T, because, due to Lemma 4.8, id' is a $2n$-edge-coloring of T. □

We now argue that the agents will never meet while moving during any given phase p.

Lemma 4.11. *Let λ and λ' be any two agents, let T be a tree network, and let $p \geq 1$ be an integer. The routes $\mathcal{R}(\lambda, p)$ and $\mathcal{R}(\lambda', p)$ are collision-free.*

Proof. Let u and u' be the nodes occupied by λ and λ', respectively, at the beginning of phase p. By the formulation of Algorithm Local-Tree-Exploration, the moves of both agents in phase p are determined in the p-th iteration of the 'while' loop of their executions of Algorithm Local-Tree-Exploration. If u and u' belong to different connected components of T_p, then due to Lemma 4.9 the routes $\mathcal{R}(\lambda, p)$ and $\mathcal{R}(\lambda', p)$ are collision-free. Hence, assume that u and v are in the same connected component T' of T_p. By Lemma 4.9, the routes in T' are given as $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(u, 2^{p+1}, id')$ and $\mathcal{R}(\lambda', p) = \mathcal{T}^{\ell(p)}(u', 2^{p+1}, id')$. Thus, the proof is complete in view of Lemma 4.1. □

Taking into account the above lemmas, we obtain our main result for local exploration of trees.

Theorem 4.12. *Let T be a tree network and let $\lambda_1, \ldots, \lambda_k$, $1 \leq k \leq n$, be the agents initially located at pairwise different nodes of T. Suppose that the agent λ_i uses Algorithm* Local-Tree-Exploration *to compute its route $\mathcal{R}^\ell(\lambda_i)$, for each $i = 1, \ldots, k$. Then, $\mathcal{R}^\ell(\lambda_1), \ldots, \mathcal{R}^\ell(\lambda_k)$ are collision-free exploration strategies of length $O(n^2)$.*

Proof. By Lemma 4.10, each route $\mathcal{R}^\ell(\lambda_i)$, $i = 1, \ldots, l$ covers T in at least one phase. Since the route performed by each agent is closed due to the backtracking steps included in Algorithm Local-Tree-Exploration, $\mathcal{R}^\ell(\lambda_i)$ is an exploration strategy for λ_i. Taking into account that the phases of all agents are perfectly synchronized in each phase, and that the agents perform backtracking and stop after exactly the same phase $\lceil \log_2 n \rceil$, it follows from Lemma 4.11 that their exploration strategies are collision-free. Finally, from the definition of ℓ we have $\ell = O(n^2)$. □

4.3.2 Local exploration of general networks

For the purposes of analysis, we introduce some auxiliary notation concerning the so-called *anonymous graph model*. In this model nodes are anonymous, and each edge has two port numbers assigned, each to one of its endpoints, in such a way that the ports at edges incident to any node form a set of consecutive integers, starting from 1. An agent located at a node v can only perform its next move based on the local port numbers.

Before continuing, we provide several comments and informal intuitions concerning this model. First note that a collision-free exploration is, in general, impossible in arbitrary anonymous port-labeled networks. (This is the case, for example, for two agents located initially in symmetric, and thus indistinguishable, positions at the endpoints of a 3-node path.) However, we will overcome this difficulty by designing an auxiliary port-labeled network $A(G)$ based on the node-labeled network G, that has the property that each edge has identical port numbers at both of its endpoints, and in such a case the collision-free exploration will be guaranteed to exist. The behavior of an agent can be seen as navigating in our node-labeled network G by navigating in the underlying 'virtual' port-labeled network $A(G)$. In particular, the function id' defined in (4.11) provides both port numbers for each edge. Hence, each agent, while present at any node v can compute the port number of the edges incident to v. Then, the agent 'simulates' its next move in the port-labeled network and, based on that, performs the move in the node-labeled network.

As a tool for our analysis we use the theory of *universal sequences* (formal definitions are provided below) that has been developed for regular port-labeled networks. Such a universal sequence, once computed by all agents, is then used to find a collision-free exploration strategy in the port-labeled network. In view of our earlier comment, the latter results in the collision-free exploration strategy in the node-labeled network.

We say that a network is *d-regular* if all nodes of the network have degrees equal to d. Given a port-labeled network A and a node v of A, we say that an agent λ initially located at v *follows* a sequence of integers $U = (x_1, \ldots, x_l)$, with $1 \leq x_i \leq d$ for $i = 1, \ldots, l$, if for each $i = 1, \ldots, l$, in round i the agent λ performs a move along the

edge with port number x_i at its current node. By a slight extension of notation, we allow a port-labeled network to have self-loops (with exactly one port number assigned to the loop); a traversal of the self-loop is assumed not to change the location of the agent.

We recall the concept of universal sequences defined in Chapter 1. We say that a sequence U of integers is (n,d)-*universal* if for each node v of each regular n-node network A of degree d, an agent initially placed at v visits each node of A by following U. Aleliunas *et al.* [6] have shown non-constructively that for each $n > 0$ and $d > 0$, there exists a (n,d)-universal sequence of length $O(d^2n^3 \log n)$ for networks with self-loops. Note that a (n,d)-universal sequence can be computed (rather inefficiently) by examining all sequences of the considered length and for each such candidate sequence one can generate all n-node port-labeled regular networks of degree d. Once a sequence U and a network A are selected, it can be tested if following U from each node of A results in visiting all nodes of A.

Given a node-labeled network G, we define the corresponding port-labeled network $A(G)$ so that there exists a bijection $\varphi\colon V(G) \to V(A(G))$ such that $\{u,v\} \in E(G)$ if and only if $\{\varphi(u),\varphi(v)\} \in E(A(G))$, and for each $\{u,v\} \in E(G)$ the port numbers at both endpoints of edge $\{\varphi(u),\varphi(v)\} \in E(A(G))$ are equal to $id'(\{u,v\})$. Since, according to Lemma 4.8, id' is an edge-coloring of G, no two edges of $A(G)$ sharing a node have the same port number at this node. Then, for each node $u \in V(G)$ we add $2n - |N_G(u)|$ loops at $\varphi(u)$ in $A(G)$. As a result, the degree of each node of $A(G)$ is $2n$, and the length of the universal sequences constructed following [6], which we will use when exploring $A(G)$, will not exceed $O(n^5 \log n)$. In what follows, we will identify exploration of G with exploration of $A(G)$.

Theorem 4.13. *There exists an algorithm that allows any set of agents located initially at distinct nodes of any network G, and having no information about G, to compute collision-free exploration strategies of length $O(n^5 \log n)$.*

Proof. Consider an execution of Algorithm Local-Tree-Exploration such that the sequence $U(b)$ defined in Section 4.3.1 is replaced by a $(b,2b)$-universal sequence. By [6], such a sequence exists and is of length $O(n^5 \log n)$. We argue that for this modified algorithm, the route $\mathcal{R}(\lambda)$ of each agent λ is a collision-free exploration strategy of G.

First, observe that $\mathcal{R}(\lambda)$ is a well-defined route, because, due to the formulation of Algorithm Local-Tree-Exploration, the agent λ does check if an edge $\{v,u\}$ exists before moving from v to u in any round. Now we argue that $\mathcal{R}(\lambda)$ covers G. Consider phase $p = \lceil \log_2 n \rceil$ that consists of the rounds in which the moves of λ are determined in the p-th iteration of the 'while' loop of Algorithm Local-Tree-Exploration. We have that $id(v) \leq b$ for each node v of G, because $b \geq n$ in this particular iteration. Let λ' be an agent that follows $U(b)$ in $A(G)$, starting at the node $\varphi(v')$ such that v' is the node occupied by λ at the beginning of phase p. By the formulation of Algorithm Local-Tree-Exploration, the agent λ moves from v to u in round s of phase p if and only if $id'(\{u,v\}) = U_s(b)$. By the definition of $A(G)$, the port number of $\{\varphi(u),\varphi(v)\}$ at $\varphi(v)$ is $U_s(b)$. Hence, λ goes from v to u in round s of phase p if and only if λ' goes from $\varphi(v)$ to $\varphi(u)$ in $A(G)$ in round s.

Since $U(b)$ is $(b, 2b)$-universal and $b \geq n$ in phase p, the route of λ in phase p covers G, regardless of the position of λ at the beginning of phase p. Finally, the fact that $\mathcal{R}(\lambda)$ is closed is due to the formulation of Algorithm Local-Tree-Exploration (λ backtracks its moves performed during the execution of the 'while' loop).

Let λ and λ' be two agents initially placed at distinct nodes of G. We prove that their routes $\mathcal{R}(\lambda)$ and $\mathcal{R}(\lambda')$ are collision-free. Similarly as in the proof of Lemma 4.10 one can argue that the number of phases for each agent equals $\lceil \log_2 n \rceil$. Hence, it is enough to analyze the moves of λ and λ' in an arbitrarily selected phase $p \in \{1, \ldots, \lceil \log_2 n \rceil\}$. Suppose that λ moves from v to u in round s of phase p. This implies that $id'(\{u, v\}) = U_s(b)$. If λ' is located at u at the beginning of this round, then λ' moves from u to v in round s of phase p, because it also verifies that $id'(\{u, v\}) = U_s(b)$. Also, two agents cannot simultaneously move from v to u and from v' to u for two different nodes v and v', because $id(v) \neq id(v')$ and therefore $id'(\{v, u\}) = id(v) + id(u) \neq id(v') + id(u) = id'(\{v', u\})$.

To complete the proof, observe that for each agent λ the length of its route is at most

$$2 \sum_{i=1}^{\lceil \log_2 n \rceil} |U(2^i)| = \sum_{i=1}^{\lceil \log_2 n \rceil} O(2^{5i} \log 2^i) = O(n^5 \log n).$$

□

4.4 Conclusions

We have shown that, in our model, a solution to the collision-free exploration problem is always feasible, even when the agents only have local knowledge. This should be sharply contrasted with asynchronous variants of the problem (when agents do not have synchronized clocks or may perform an asynchronous meeting in the middle of an edge), in which a solution is not always feasible. This is the case even for the fully symmetric scenario on the two-node line, where two agents starting from the two nodes cannot complete exploration without swapping, thus implying the possibility of asynchronous meeting.

Chapter 5

Conclusions

In this thesis we have studied algorithms for graph exploration with mobile agents. Most of the results present in the literature concerned the case of a single agent. In this thesis, we focused on studying time-efficient algorithms for multiple mobile entities in several different models. In the considered scenarios, we showed in what way and to what extent exploration with multiple agents is more efficient than single-agent exploration. In particular, we considered tradeoffs between team size and exploration time. In Chapter 2, devoted to team exploration, we were considering agents equipped with memory whereas in Chapter 3, devoted to the rotor-router model, some memory was available at the nodes. Finally, in the collision-free model studied in Chapter 4 we showed that it is possible to perform efficient exploration with every agent not meeting any other agent operating in the graph.

When comparing time-efficiency of exploration in a team we can observe that making good use of the communication capabilities of agents is essential when designing an efficient exploration strategy. In Chapter 2 we showed that it is possible to explore any graph in time proportional to the diameter of the graph using a team of polynomial number of communicating agents. On the other hand, for the rotor-router considered in Chapter 3, for some graph classes we need exponentially many agents to achieve the same exploration time. Interestingly, the case where the rotor-router turned out to be more efficient is the case of expanders. Our analysis shows that efficient exploration of expanders can be performed using the multi-agent rotor-router with a smaller number of agents than that needed by the algorithm from Chapter 2. We note that the ability of agents to exchange information is a very powerful feature of the model allowing for sophisticated algorithms, whereas the rotor-router rule is rather simple.

Our analysis of the multi-agent rotor-router concerned both the general case and a number of important graph classes, and was tight (or almost tight) for any number of agents. On the other hand, our results from Chapter 2 worked for any graph but only for a large number of agents.

An interesting research perspective is to consider scenarios in which agents have internal memory and at each node there is a whiteboard with some number of bits of space. Two optimization criteria of interest in this context are the size of the internal memory

of each agent and the size of the whiteboard at each node. In future work it would be interesting to establish the weakest possible model for which it is possible to explore graphs in time $O(D)$ using a team of polynomial size. The multi-agent rotor-router often needs an exponential number of agents but perhaps it is possible to decrease the cover time by adding some number of bits of memory to the agents and adapting the rules of the agents accordingly. In such a model, agents would no longer be propagated by the environment (as was the case in the rotor-router model) but could make "conscious" decisions. An interesting direction of study is to consider tradeoffs between the number of bits given to the agents and sizes of whiteboards on each node that allow for the polynomial team of agents to explore graphs in optimal time. Such tradeoffs may also depend on graph topology, since we showed in Chapter 3 that exploration of expanders with agents with no internal memory is possible in time $O(D)$ using polynomial number of agents.

Another important question concerns the values of speedup of exploration in the communicating agent model (Chapter 2). If we look at the speedup as a function of k (team size) then the question would be whether in this model the optimal speedup admits threshold behaviour i.e., if its dependence on k can be split into several ranges, with different asymptotic behaviour. In Chapter 3, we observed such thresholds for the multi-agent rotor-router for several graph classes.

Bibliography

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, 2006.

[2] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[3] H. Akbari and P. Berenbrink. Parallel rotor walks on finite graphs and applications in discrete load balancing. In *SPAA*, pages 186–195. ACM, 2013.

[4] D. Aldous. On the time taken by random walks on finite groups to visit every state. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 62(3):361–374, 1983.

[5] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. http://stat-www.berkeley.edu/users/aldous/RWG/book.html, 2001.

[6] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *FOCS*, pages 218–223. IEEE Computer Society, 1979.

[7] N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. *Combinatorics, Probability & Computing*, 20(4):481–502, 2011.

[8] N. Alon, F. R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. In *STOC*, pages 583–591, 1993; also SIAM J. Discrete Math. 7(3): 513-530, 1994.

[9] S. Alpern. Rendezvous search: A personal perspective. *Operations Research*, 50(5):772–795, 2002.

[10] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. International Series in Operations Research & Management Science. Springer, 2003.

[11] C. Ambühl, L. Gąsieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. *ACM Transactions on Algorithms*, 7(2):17, 2011.

[12] H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. In *Intelligent Control Proceedings of the IEEE International Symposium on*, pages 453–460, Aug 1995.

[13] R. Armoni, A. Ta-Shma, A. Wigderson, and S. Zhou. An $O(\log(n)^{4/3})$ space algorithm for $(s,\ t)$ connectivity in undirected graphs. *J. ACM*, 47(2):294–311, 2000.

[14] R. Baldoni, F. Bonnet, A. Milani, and M. Raynal. Anonymous graph exploration without collision by mobile robots. *Information Processing Letters*, 109(2):98–103, 2008.

[15] E. Bampas, L. Gąsieniec, N. Hanusse, D. Ilcinkas, R. Klasing, and A. Kosowski. Euler tour lock-in problem in the rotor-router model. In *DISC*, volume 5805 of *Lecture Notes in Computer Science*, pages 423–435. Springer, 2009.

[16] E. Bampas, L. Gąsieniec, R. Klasing, A. Kosowski, and T. Radzik. Robustness of the rotor-router mechanism. In *OPODIS*, volume 5923 of *Lecture Notes in Computer Science*, pages 345–358. Springer, 2009.

[17] Y. Baudoin and M. Habib. *Using Robots in Hazardous Environments: Landmine Detection, De-Mining And Other Applications*. Woodhead Publishing in Mechanical Engineering. Elsevier, 2010.

[18] P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrich, and T. Sauerwald. Randomized diffusion for indivisible loads. In *SODA*, pages 429–439. SIAM, 2011.

[19] S. N. Bhatt, S. Even, D. S. Greenberg, and R. Tayar. Traversing directed eulerian mazes. *J. Graph Algorithms Appl.*, 6(2):157–173, 2002.

[20] L. Blin, J. Burman, and N. Nisse. Perpetual Graph Searching. Technical Report RR-7897, INRIA, 2012-02.

[21] L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2010.

[22] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[23] A. Z. Broder, P. Raghavan, R. W. Taylor, A. R. Karlin, A. R. Karlin, E. Upfal, and E. Upfal. Trading space for time in undirected s-t connectivity. In *STOC*, pages 543–549, 1991.

[24] J. Chalopin, S. Das, Y. Disser, M. Mihalák, and P. Widmayer. Telling convex from reflex allows to map a polygon. In *STACS*, volume 9 of *LIPIcs*, pages 153–164. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[25] J. Chalopin, S. Das, and A. Kosowski. Constructing a map of an anonymous graph: Applications of universal sequences. In *OPODIS*, volume 6490 of *Lecture Notes in Computer Science*, pages 119–134. Springer, 2010.

[26] J. Chalopin, S. Das, A. Labourel, and E. Markou. Black hole search with finite automata scattered in a synchronous torus. In *DISC*, volume 6950 of *Lecture Notes in Computer Science*, pages 432–446. Springer, 2011.

[27] J. Chalopin, S. Das, A. Labourel, and E. Markou. Tight bounds for black hole search with scattered agents in synchronous rings. *Theoretical Computer Science*, 509:70–85, 2013.

[28] J. Chalopin, S. Das, and P. Widmayer. Rendezvous of mobile agents in directed graphs. In *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2010.

[29] J. Chalopin, P. Flocchini, B. Mans, and N. Santoro. Network exploration by silent and oblivious robots. In *WG*, volume 6410 of *Lecture Notes in Computer Science*, pages 208–219, 2010.

[30] A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1997.

[31] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 1181–1196. Springer, 2003.

[32] S. A. Cook and C. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM J. Comput.*, 9(3):636–652, 1980.

[33] C. Cooper and A. M. Frieze. The cover time of random regular graphs. *SIAM J. Discrete Math.*, 18(4):728–740, 2005.

[34] C. Cooper, D. Ilcinkas, R. Klasing, and A. Kosowski. Derandomizing random walks in undirected graphs using locally fair exploration strategies. *Distributed Computing*, 24(2):91–99, 2011.

[35] C. Cooper, R. Klasing, and T. Radzik. Searching for black-hole faults in a network using multiple agents. In *OPODIS*, volume 4305 of *Lecture Notes in Computer Science*, pages 320–332. Springer, 2006.

[36] C. Cooper, R. Klasing, and T. Radzik. A randomized algorithm for the joining protocol in dynamic distributed networks. *Theoretical Computer Science*, 406(3):248–262, 2008.

[37] C. Cooper, R. Klasing, and T. Radzik. Locating and repairing faults in a network with mobile agents. *Theor. Comput. Sci.*, 411(14-15):1638–1647, 2010.

[38] J. N. Cooper, B. Doerr, J. H. Spencer, and G. Tardos. Deterministic random walks on the integers. *Eur. J. Comb.*, 28(8):2072–2090, 2007.

[39] J. N. Cooper and J. Spencer. Simulating a random walk with constant error. *Combinatorics, Probability & Computing*, 15(6):815–822, 2006.

[40] J. Czyzowicz, S. Dobrev, L. Gąsieniec, D. Ilcinkas, J. Jansson, R. Klasing, I. Lignos, R. Martin, K. Sadakane, and W.-K. Sung. More efficient periodic traversal in anonymous undirected graphs. *Theoretical Computer Science*, 444:60–76, 2012.

[41] J. Czyzowicz, S. Dobrev, R. Královic, S. Miklík, and D. Pardubská. Black hole search in directed graphs. In *SIROCCO*, volume 5869 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2009.

[42] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *ESA*, volume 6942 of *Lecture Notes in Computer Science*, pages 701–712. Springer, 2011.

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

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

[45] J. Czyzowicz, D. R. Kowalski, E. Markou, and A. Pelc. Searching for a black hole in synchronous tree networks. *Combinatorics, Probability & Computing*, 16(4):595–619, 2007.

[46] J. Czyzowicz, A. Pelc, and A. Labourel. How to meet asynchronously (almost) everywhere. *ACM Transactions on Algorithms*, 8(4):37, 2012.

[47] X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1-3):97–112, 2008.

[48] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *SPAA*, pages 139–148. ACM, 2011.

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

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

[51] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1-3):343–362, 2004.

[52] S. Devismes, F. Petit, and S. Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. *Theoretical Computer Science*, 498:10–27, 2013.

[53] Y. Dieudonné and F. Petit. Swing words to make circle formation quiescent. In *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 166–179. Springer, 2007.

[54] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *J. Algorithms*, 51(1):38–63, 2004.

[55] Y. Disser. *Mapping Polygons.* PhD thesis, 2011.

[56] Y. Disser, S. Ghosh, M. Mihalák, and P. Widmayer. Mapping a polygon with holes using a compass. In *Algorithms for Sensor Systems*, volume 7718 of *Lecture Notes in Computer Science*, pages 78–89. Springer Berlin Heidelberg, 2013.

[57] Y. Disser, M. Mihalák, and P. Widmayer. A polygon is determined by its angles. *Comput. Geom.*, 44(8):418–426, 2011.

[58] Y. Disser, M. Mihalák, and P. Widmayer. Mapping polygons with agents that measure angles. In *WAFR*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 415–425. Springer, 2012.

[59] S. Dobrev, P. Flocchini, R. Kralovic, P. Ruzicka, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Networks*, 47(2):61–71, 2006.

[60] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Computing*, 19(1):1–18, 2006.

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

[62] B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Combinatorics, Probability & Computing*, 18(1-2):123–144, 2009.

[63] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom rumor spreading. In *SODA*, pages 773–781. SIAM, 2008.

[64] C. A. Duncan, S. G. Kobourov, and V. S. A. Kumar. Optimal constrained graph exploration. *ACM Transactions on Algorithms*, 2(3):380–402, 2006.

[65] M. Dynia, M. Korzeniowski, and C. Schindelhauer. Power-aware collective tree exploration. In *ARCS*, volume 3894 of *Lecture Notes in Computer Science*, pages 341–351. Springer, 2006.

[66] M. Dynia, J. Kutylowski, F. Meyer auf der Heide, and C. Schindelhauer. Smart robot teams exploring sparse trees. In *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 2006.

[67] M. Dynia, J. Lopuszanski, and C. Schindelhauer. Why robots need maps. In *SIROCCO*, volume 4474 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2007.

[68] K. Easton and J. Burdick. A coverage algorithm for multi-robot boundary inspection. In *ICRA 2005. Proceedings of the 2005 IEEE International Conference on Robotics and Automation.*, pages 727–734, April 2005.

[69] J. Edmonds. Matroids and the greedy algorithm. *Math. Programming*, 1:127–136, 1971.

[70] K. Efremenko and O. Reingold. How well do random walks parallelize? In *APPROX-RANDOM*, pages 476–489, 2009.

[71] R. Elsässer and T. Sauerwald. Tight bounds for the cover time of multiple random walks. *Theoretical Computer Science*, 412(24):2623–2641, 2011.

[72] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Ants: Mobile finite state machines. *CoRR*, abs/1311.3062, 2013.

[73] S. Even. *Graph Algorithms*. Cambridge University Press, 2011.

[74] U. Feige. A tight lower bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(4):433–438, 1995.

[75] U. Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(1):51–54, 1995.

[76] U. Feige. Collecting coupons on trees, and the cover time of random walks. *Computational Complexity*, 6(4):341–356, 1997.

[77] O. Feinerman, A. Korman, Z. Lotker, and J.-S. Sereni. Collaborative search on the plane without communication. In *PODC*, pages 77–86. ACM, 2012.

[78] R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2005.

[79] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theoretical Computer Science*, 411(14-15):1583–1598, 2010.

[80] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013.

[81] P. Flocchini, D. Ilcinkas, and N. Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033, 2012.

[82] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous oblivious robots with limited visibility. In *STACS*, volume 2010 of *Lecture Notes in Computer Science*, pages 247–258. Springer, 2001.

[83] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1-3):412–447, 2008.

[84] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, June 2008.

[85] P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[86] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.

[87] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Impact of memory size on graph exploration capability. *Discrete Applied Mathematics*, 156(12):2310–2319, 2008.

[88] P. Fraigniaud, D. Ilcinkas, S. Rajsbaum, and S. Tixeuil. The reduced automata technique for graph exploration space lower bounds. In *Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2006.

[89] P. Fraigniaud and A. Pelc. Delays induce an exponential memory gap for rendezvous in trees. *ACM Transactions on Algorithms*, 9(2):17, 2013.

[90] T. Friedrich, M. Gairing, and T. Sauerwald. Quasirandom load balancing. *SIAM J. Comput.*, 41(4):747–771, 2012.

[91] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. *Electr. J. Comb.*, 17(1), 2010.

[92] M. Fürer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, 1994.

[93] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[94] L. Gasieniec, R. Klasing, R. A. Martin, A. Navarra, and X. Zhang. Fast periodic graph exploration with constant memory. *J. Comput. Syst. Sci.*, 74(5):808–822, 2008.

[95] L. Gasieniec and T. Radzik. Memory efficient anonymous graph exploration. In *WG*, volume 5344 of *Lecture Notes in Computer Science*, pages 14–29, 2008.

[96] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Practical recommendations on crawling online social networks. *Selected Areas in Communications, IEEE Journal on*, 29(9):1872–1892, October 2011.

[97] M. X. Goemans. Minimum bounded degree spanning trees. In *FOCS*, pages 273–282. IEEE Computer Society, 2006.

[98] T. Herman and T. Masuzawa. Self-stabilizing agent traversal. In *WSS*, volume 2194 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2001.

[99] Y. Higashikawa and N. Katoh. Online exploration of all vertices in a simple polygon. In *FAW-AAIM*, volume 7285 of *Lecture Notes in Computer Science*, pages 315–326. Springer, 2012.

[100] Y. Higashikawa, N. Katoh, S. Langerman, and S.-i. Tanigawa. Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization*, pages 1–16, 2012.

[101] A. Hsieh and S. Lacroix, editors. *Special Issue: Special Issue on Multi Autonomous Ground-robotic International Challenge (MAGIC)*, volume 29 of *Journal of Field Robotics*, 2012.

[102] S. Ikeda, I. Kubo, N. Okumoto, and M. Yamashita. Impact of local topological information on random walks on finite graphs. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 1054–1067. Springer, 2003.

[103] D. Ilcinkas. Setting port numbers for fast graph exploration. *Theor. Comput. Sci.*, 401(1-3):236–242, 2008.

[104] A. Kawamura and Y. Kobayashi. Fence patrolling by mobile agents with distinct speeds. In *ISAAC*, volume 7676 of *Lecture Notes in Computer Science*, pages 598–608. Springer, 2012.

[105] S. Kijima, K. Koga, and K. Makino. Deterministic random walks on finite graphs. In *ANALCO*, pages 16–25. SIAM, 2012.

[106] W. B. Kinnersley. Cops and Robbers is EXPTIME-complete. *arXiv:1309.5405v1*, Sept. 2013.

[107] R. Klasing. Efficient exploration of anonymous undirected graphs. In *IWOCA*, volume 8288 of *Lecture Notes in Computer Science*, pages 7–13. Springer, 2013.

[108] R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theoretical Computer Science*, 411(34-36):3235–3246, 2010.

[109] R. Klasing, E. Markou, and A. Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theoretical Computer Science*, 390(1):27–39, 2008.

[110] R. Klasing, E. Markou, T. Radzik, and F. Sarracco. Approximation bounds for black hole search problems. *Networks*, 52(4):216–226, 2008.

[111] A. Kosowski. *Time and Space-Efficient Algorithms for Mobile Agents in an Anonymous Network*. Hdr, Université Sciences et Technologies - Bordeaux I, Sept. 2013.

[112] A. Kosowski and A. Navarra. Graph decomposition for memoryless periodic exploration. *Algorithmica*, 63(1-2):26–38, 2012.

[113] A. Kosowski, A. Navarra, and M. C. Pinotti. Synchronous black hole search in directed graphs. *Theoretical Computer Science*, 412(41):5752–5759, 2011.

[114] D. R. Kowalski and A. Malinowski. How to meet in anonymous network. *Theoretical Computer Science*, 399(1-2):141–156, 2008.

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

[116] E. Kranakis, D. Krizanc, and S. Rajsbaum. Mobile agent rendezvous: A survey. In *SIROCCO*, volume 4056 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006.

[117] D. Krizanc and L. Zhang. Many-to-one packed routing via matchings. In *COCOON*, volume 1276 of *Lecture Notes in Computer Science*, pages 11–17. Springer, 1997.

[118] A. Lamani, M. G. Potop-Butucaru, and S. Tixeuil. Optimal deterministic ring exploration with oblivious asynchronous robots. In *SIROCCO*, volume 6058 of *Lecture Notes in Computer Science*, pages 183–196. Springer, 2010.

[119] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Commun. ACM*, 42(3):88–89, Mar. 1999.

[120] A. S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, Apr. 1993.

[121] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[122] J. Łopuszański. Tree exploration (in Polish). Tech-report, Institute of Computer Science, University of Wrocław, Poland. 2007.

[123] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[124] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou. The complexity of searching a graph. In *FOCS*, pages 376–385, Oct 1981.

[125] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012.

[126] N. Nisan. RL $\subseteq$ SC. *Computational Complexity*, 4:1–11, 1994.

[127] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log(n)^{3/2})$ space. In *FOCS*, pages 24–29, 1992.

[128] Y. Nonaka, H. Ono, K. Sadakane, and M. Yamashita. The hitting and cover times of metropolis walks. *Theoretical Computer Science*, 411(16-18):1889–1894, 2010.

[129] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2–3):235 – 239, 1983.

[130] C. Ortolf and C. Schindelhauer. Online multi-robot exploration of grid graphs with rectangular obstacles. In *SPAA*, pages 27–36. ACM, 2012.

[131] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281–295, 1999.

[132] G. E. Pantziou, A. Roberts, and A. Symvonis. Many-to-many routings on trees via matchings. *Theoretical Computer Science*, 185(2):347–377, 1997.

[133] T. Parsons. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer Berlin Heidelberg, 1978.

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

[135] G. Prencipe. On the feasibility of gathering by autonomous mobile robots. In *SIROCCO*, volume 3499 of *Lecture Notes in Computer Science*, pages 246–261. Springer, 2005.

[136] V. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy. Eulerian walkers as a model of self-organized criticality. *Phys. Rev. Lett.*, 77(25):5079–5082, Dec 1996.

[137] A. Quilliot. Problemes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes. *These d'Etat, Université de Paris VI*, 1983.

[138] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of Markov chains and the analysis of iterative load balancing schemes. In *FOCS*, pages 694–705. IEEE Computer Society, 1998.

[139] O. Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[140] H.-A. Rollik. Automaten in Planaren Graphen. *Acta Inf.*, 13:287–298, 1980.

[141] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177 – 192, 1970.

[142] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28:1347–1363, 1999.

[143] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15:918–933, 1999.

[144] V. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.

[145] L. Zhang. Optimal bounds for matching routing on trees. *SIAM J. Discrete Math.*, 12(1):64–77, 1999.

[146] D. Zuckerman. A technique for lower bounding the cover time. *SIAM J. Discrete Math.*, 5(1):81–87, 1992.

Publications included in this thesis

[T1] J. Czyzowicz, D. Dereniowski, L. Gąsieniec, R. Klasing, A. Kosowski, and **D. Pająk**. Collision-free network exploration. In *LATIN*, pages 342–354, 2014.

[T2] D. Dereniowski, Y. Disser, A. Kosowski, **D. Pająk**, and P. Uznański. Fast Collaborative Graph Exploration. In *ICALP (2)*, pages 520–532, 2013, journal version accepted to Information and Computation.

[T3] D. Dereniowski, A. Kosowski, **D. Pająk**, and P. Uznański. Bounds on the Cover Time of Parallel Rotor Walks. In *STACS*, pages 263–275, 2014.

[T4] R. Klasing, A. Kosowski, **D. Pająk**, and T. Sauerwald. The Multi-Agent Rotor-Router on the Ring: A Deterministic Alternative to Parallel Random Walks. In *PODC*, pages 365–374, 2013.

[T5] A. Kosowski and **D. Pająk**. Does Adding More Agents Make a Difference? A Case Study of Cover Time for the Rotor-Router. `http://hal.inria.fr/hal-00950743`, accepted to *ICALP*, 2014.