AIX-MARSEILLE UNIVERSITÉ
ECOLE DOCTORALE 184
LIF/UMR 7279

Thèse présentée pour obtenir le grade universitaire de docteur

Discipline: Informatique
Spécialité: Algorithmique Distribuée

Christina KAROUSATOU

Algorithmes Distribués pour les Agents Mobiles sous Contraintes d'Énergie

Distributed Algorithms for Energy Constrained Mobile Agents

Soutenue le 08/12/2017 devant le jury composé de:

Pierre FRAIGNIAUD	IRIF, Université Paris-Diderot	Rapporteur
Leszek Antoni GASIENIEC	University of Liverpool	Rapporteur
Linda PAGLI	University of Pisa	Examinateur
Victor CHEPOI	LIF, Aix-Marseille Université	Examinateur
Shantanu DAS	LIF, Aix-Marseille Université	co-Directeur de thèse
Peter NIEBERT	LIF, Aix-Marseille Université	Directeur de thèse

Numéro national de thèse/suffixe local: 2017AIXM0001/001ED62

Cette oeuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Résumé

Dans cette thèse, nous étudions et concevons des algorithmes pour des agents mobiles se déplaçant dans un graphe avec une énergie limité, restreignant leurs mouvements. Chaque agent mobile est une entité, équipée d'une batterie, qui peut parcourir les arêtes du graphe et visiter les noeuds du graphe. A chaque mouvement, l'agent consomme une partie de son énergie. Contrairement à divers modèles bien étudiés pour les agents mobiles, très peu de recherches ont été menées pour le modèle compte tenu des limites d'énergie. Nous étudions les problèmes fondamentaux de l'exploration d'un graphe, du gathering et du collaborative delivery dans ce modèle.

Le problème d'exploration des graphes consiste à visiter tous les noeuds d'un graphe donné. Nous étudions l'exploration des graphes dans trois modèles différents. Tout d'abord, nous considérons une exploration collaborative où le niveau d'énergie des agents est fixé et nous souhaitons minimiser le nombre d'agents utilisés pour explorer entièrement le graphe. Ensuite, nous considérons que les niveaux d'énergie et le nombre d'agents sont fixés. L'objectif est de maximiser le nombre de noeuds explorés. Enfin, nous considérons l'exploration d'arbres connus par un unique agent qui peut collecter de l'énergie à partir des nœuds qu'il visite pour poursuivre l'exploration. L'objectif est de maximiser la quantité d'énergie collectée que l'agent ramène à la racine à la fin de l'exploration. Pour chacun des problèmes mentionnés, nous présentons des algorithmes efficaces et prouvons plusieurs bornes inférieures.

Pour le problème du gathering, le but des agents est de se rencontrer à un seul noeud d'un graphe, commençant des différents endroits. Pour le cas des agents contraints en énergie, nous considérons le problème modifié du near-gathering où l'objectif d'optimisation est de déplacer les agents, de telle sorte que la distance maximale entre deux agents soit minimisée. Nous montrons que le problème est NP-difficile et nous présentons un algorithme d'approximation qui correspond à la borne inférieure que nous prouvons pour ce problème. Nous proposons également un algorithme d'approximation de facteur constant pour le problème de minimisation de la distance moyenne par paire d' agents.

Le dernier problème considéré est le collaborative delivery, où étant donné un graphe, l'objectif est de déplacer un paquet de sa source vers sa destination cible par une équipe d'agents mobiles. Pour ce problème, nous supposons que les agents peuvent partager leur énergie quand ils se rencontrent, sous deux scénarios. Dans le premier scénario, nous permettons aux agents de partager leur énergie sans aucune restriction, tandis que dans le deuxième scénario, nous supposons que les agents ont limité la capacité énergétique. Nous montrons que le problème est NP-difficile pour les deux cas. Nous présentons également un algorithme resource-augmented pour le premier modèle et pour le second, nous montrons que nous pouvons résoudre le problème efficacement si nous pouvons faire une hypothèse supplémentaire sur le placement initial des agents.

Abstract

In this thesis we study and design algorithms for solving various well-known problems for mobile agents moving on a graph, with the additional constraint of limited energy which restricts the movement of the agents. Each mobile agent is an entity, equipped with a battery, that can traverse the edges of the graph and visit the nodes of the graph, consuming a part of its energy for movement. In contrast to various well-studied models for mobile agents, very little research has been conducted for the model considering the energy limitations. We study the fundamental problems of graph exploration, gathering and collaborative delivery in this model.

For the graph exploration problem the goal of the agents is to visit all the nodes of a given graph. We study graph exploration in three different settings. First, we consider the energy level of the agents is fixed and we wish to minimize the number of agents used. Next, we consider that both the energy levels and the number of agents are fixed and the goal is to maximize the number of explored nodes. Last, we consider the exploration of known trees by a single agent which can collect energy from the nodes that it visits, enabling it to continue the exploration. The goal in this setting is to maximize the amount of collected energy the agent brings back to the root at the end of the exploration. For each one of the mentioned problems, we present efficient algorithms and prove several lower bounds.

For the gathering problem the goal of the agents is to meet at a single node of a graph, starting from different locations. For the case of energy constrained agents we consider the modified problem of near-gathering where the optimization goal is to relocate the agents, in such a way that the maximum distance between any two agents is minimized. We show hardness results and we present an approximation algorithm that matches the lower bound we prove for this problem. We also provide a constant factor approximation algorithm for the problem of minimizing the average pairwise distance between agents.

The last problem we consider is Collaborative Delivery where given a graph, the goal is to move a package from its source to its target destination by a team of mobile agents. For this problem, we suppose that the agents can share their energy upon meeting, under two scenarios. In the first scenario, we allow the agents to share their energy without any restrictions, while in the second scenario, we assume the agents have restricted energy capacity. We show hardness results for both cases. We also present a resource augmented algorithm for the first setting and for the second, we show that we can solve the problem efficiently if we can make an extra assumption about the initial placement of the agents.

Acknowledgements

First and foremost, I would like to thank my PhD advisors Shantanu Das and Peter Niebert for their support, patience and guidance, throughout the past three years.

During my PhD studies I had the pleasure to have numerous collaborations whose outcome has resulted in this thesis. Particularly, I would like to thank, in alphabetical order, Evangelos Bampas, Andreas Bärtschi, Jérémie Chalopin, Jurek Czyzowicz, Dariusz Dereniowski, Jan Hackfeld, Arnaud Labourel, Euripides Markou and Matúš Mihalák.

Working at LIF has been a great pleasure and I would like to give a special thanks to my fellow PhD students, with whom I have shared more than one offices, Antoine, Makki, Eloi, Didier, Mathieu, Elie and Worachet, for creating a very positive and friendly work environment.

I am particularly grateful to Pierre Fraigniaud and Leszek Gąsieniec for agreeing to referee this manuscript. I equally wish to thank Linda Pagli and Victor Chepoi for accepting to be jury members of my PhD thesis committee.

Finally, I want to thank my family for their constant support and encouragement over all these years.

Contents

1 Introduction

Distributed computing systems are networks of computing devices that communicate and coordinate with each-other to solve a common task. There are several advantages of these systems such as the robustness to failures and the parallelization of tasks to achieve speedup over sequential computing. In particular, when each computing unit performs autonomously (such units are referred to as agents) there is no single point of failure and such a system can tolerate an arbitrary number of failures. However it is often challenging to design correct algorithms for distributed systems due to several issues such as the problem of synchronization, problems in communication between the agents and the data dependencies for the task to be performed. In some cases, the computing entities or agents may be mobile and mobility is a factor which adds to the difficulty of designing correct and efficient algorithms for distributed systems. For example, consider a team of mobile robots which are deployed to collectively explore an area of interest, to search for resources (e.g. oil) or locate dangers (hidden mines) or to collect environmental data. When the robots move autonomously, the coordination among the robots is a challenging task which requires the design of sophisticated algorithms.

The model of mobile agent computing has been initially suggested in the context of software engineering by Fukuda et al. [Fuk+99]. As a computational paradigm it offers a simple and natural setting for describing distributed systems, [LO99], and it has been studied a lot in the past few years [Kos13; Das13; FPS12; KKM10].

The mobile agents represent software entities that have the ability to move in a network, or they can represent mobile robots that navigate in an environment. The agents are usually modeled as automata and the network or the environment in which they exist is modeled as a graph. The mobile agents have the ability to move from node to node along the edges of the graph, can perform computations and can interact with other agents.

There are various properties of the agents and the networks that host them that have been considered in the literature, depending on the nature of the problem concerned. We give a brief summary of the main features of the agents and the hosts that have been considered.

- **Identifiers**. The agents can have unique identities or not. Agents without distinct identities are called *anonymous* agents. Anonymous agents are limited to execute the same algorithm, as opposed to agents with distinct identities, whose identity can be part of the input of the executed algorithm. Furthermore, the nodes of the graph can have unique labels or not. The model in which the nodes do not have unique labels, is called *anonymous* network.

- **Memory**. The size of the memory of the agents can vary. It can be considered to be unbounded, bounded or even equal to zero (*oblivious* agents).

- **Communication**. The communication between the agents can be either direct (*global* or *local*) or indirect. In the global communication scenario the agents can exchange information while being located at arbitrary nodes. To the contrary, in the local communication scenario the agents can exchange information only when they are concurrently located at the same node. In the indirect case, the agents communicate through the nodes of the graph. They can leave messages at a shared memory space (*whiteboard*) that is present at the nodes, or they can leave a mark (*token* or *pebble*) at a visited node.

- **Visibility**. The agents can have full knowledge of the graph a priori (*global* vision), or they may have no information about the graph (*local* vision). When the agents have local visibility, it is usually assumed that the agents can distinguish between the incident edges of the node hosting them. This is achieved by using a predefined local ordering of the edges, most commonly referred to as *local port numbering.* Moreover, under the visibility feature we can consider the ability of the agents to see the positions of other agents in the graph as well as the number of agents that are occupying the same node (*multiplicity detection*).

- **Time**. There exist two main models to measure time in a distributed system. The *synchronous* and the *asynchronous* settings. In the synchronous case, there is a global clock available to all nodes that is being followed by the agents as well. This global clock defines the rounds during which the agents synchronously perform their tasks. In the asynchronous setting, there is no central clock and each agent acts at its own pace. Even though the delays may be arbitrarily long, it is guaranteed to be finite.

Several algorithms and techniques have been proposed to solve basic coordination tasks for mobile agents such as making the agents meet in a single location (*rendezvous*), or spreading them evenly (*uniform dispersal*), or making them form a specific pattern, e.g. a circle or a line (*pattern formation* problem). The problems of searching, monitoring or patrolling an area have also been studied for teams of collaborating agents or robots. However, in all these basic studies, the fact that agents may have limited energy resources is not considered. This assumption is problematic since for a mobile robot, motion tends to consume energy, and if the algorithms do not optimize the amount of movement of each robot and balance these amounts well among the robots, it may so happen that several of the robots run out of energy and become dysfunctional. Even though the problems studied in this thesis are motivated by physical robots, we will be referring to them as mobile agents. In this work we are interested in designing distributed algorithms for mobile agents that are fully aware of the energy levels of the agents and plan accordingly.

1.1 Our Model

Throughout this thesis we consider discrete environments. We will be using the following setting to describe and analyze the problems visited in this work.

The network is modeled as an undirected graph $G(V, E)$, containing $n = |V|$ nodes and $m = |E|$ edges. The set of nodes describes the locations of the graph that are capable of hosting an agent, whereas the set of edges describes the communication links on which the agents can travel in any direction. We will also be considering edge-weighted graphs, defined as a pair (G, w), where G is a graph and $w : E \to \mathbb{N}$ is the weight function. The weight function assigns to each edge a non-negative value representing the length of the edge. Given any two nodes u and v in G we denote by $d(u, v)$ the length of the shortest path between u and v.

A group of k distinct and autonomous mobile agents is deployed in G. Each agent has memory proportional to the size of the graph. Any node in V that corresponds to the initial position of an agent is called a *homebase* node and we denote by H the set containing all such nodes.

In this model, we consider that the nodes of the graph do not have any sort of memory. Therefore, the agents cannot leave any messages on the nodes, nor use any mechanism for marking the nodes. The communication of the agents is only direct and we will be considering both the global and the local communication scenarios.

Regarding the visibility of the agents, we study both the cases of global and local visibility. Independently of their vision range, we always consider that the nodes have local port numbering so that an agent arriving at a node can distinguish among the incident edges. For every node $v \in V$, the local port numbering is defined as the bijective function $\lambda_v : \{(v, w) \in E\} \to \{0, \ldots, deg(v) - 1\}$, where $deg(v)$ denotes the degree of node v. We also consider that the agents can detect the presence of other agents at the nodes, as well as their multiplicity.

The agents move at synchronous rounds, respecting the global clock, at the same speed. At each round an agent may choose to move to an adjacent node or remain at its current position.

Last, we consider that the agents have an amount of initial energy denoted by B. They use this energy to move in the graph. More precisely, the agents consume energy when traversing an edge proportional to the distance covered, that is equal to the weight of the edge, for the case of weighted graphs, and it is equal to 1 for the case of unweighted graphs. The agents once they have depleted their energy, they cannot perform any other task.

In the cases where the agents have only local visibility (*online* case), we measure the efficiency of our solutions in terms of the *competitive ratio*. It is defined as the worst case ratio of the cost of the online algorithm for some graph G over the cost of the optimal *offline* algorithm for the same graph, where the offline case is when the agents have global visibility.

This work is one of the first studies concerned with energy constrained agents.

In this work, we consider both the agents and the network to be fault free.

1.2 Problems Studied

In this manuscript we study and design algorithms for solving various well-known problems in the field of mobile agent computing, with a view to optimizing the energy consumption of the agents, instead of optimizing the time agents need for solving such problems. Moreover, we consider agents with a hard constraint on their available energy resources, in the sense that they are no longer functional once they deplete their battery. A plethora of interesting problems arises when we consider mobile agents with limited energy. The focus of this research is on the following problems.

Graph Exploration

The graph exploration problem is a well studied problem in computer science with a wide range of applications from searching the internet to navigation of robots in unknown environments. Given a graph $G(V, E)$ the objective is to visit all the nodes of the graph starting from a given node. We will be considering the following types of exploration.

- **Online exploration**. The agents have no initial knowledge about the topology of the graph, apart from their homebase node.
- **Offline exploration**. The agents have full knowledge of the graph topology.
- **Exploration with stop**. Once every node of the graph has been visited by at least one agent, we require that the agents consequently do not make any further moves.
- **Exploration with return**. After every node of the graph has been visited by at least one agent, we require that the agents return to their homebase node.

In graph exploration the most common goal is to minimize the time needed to perform the exploration. The energy-awareness of the agents considered in this work, as we remarked previously, gives rise to other optimization goals. If we consider that we are given a fixed number of available agents, the natural optimization goal is to find the minimum amount of energy needed by the agents to perform the exploration. On the other hand, if we consider that the agents have a fixed amount of available energy, then the goal is to minimize the total number of agents needed to fully explore the graph. Finally, if we consider that both the number of the agents and the amount of their available energy are fixed then full exploration of the graph may not be possible. The optimization goal in such a case, is to maximize the total number of nodes explored.

Last, in this work we consider a variant of the exploration with return with a single agent. In this setting, some nodes of the graph contain resources (prizes) that the agent can collect and use as fuel to continue with the exploration. Hence the agent can continue the exploration for a longer time without running out of energy/fuel. This problem has possible applications for scavenger robots that look for fuel sources while performing interplanetary exploration. On a more practical level, the problem models variants of travelling salesperson problems where the salesperson makes profits by visiting some towns and uses this profit to buy fuel for his vehicle and continue the travel. We call this problem the **sustainable exploration** problem and we define it formally using a weighted graph G where weights on the nodes (called gain) represent the energy gained by the agent and the weights on the edges (called cost) represents the energy consumed while traversing the edge. The agent starts at a designated node r and its initial energy is the value of the gain at r. At any point the agent may traverse an edge if its remaining energy is more than the cost of the edge and during this traversal the value of its energy is reduced by the cost of the edge traversed. On visiting a node v for the first time, the agent collects energy equal to the gain at that node. The objective is to maximize the remaining energy at the end of the traversal. Note that in sustainable exploration it is not asked for all the nodes to be visited.

Gathering

The problem of Gathering (or rendezvous) is another important problem in mobile agent computing, as it can be a basic subtask useful for solving other problems, such as exchanging information, locating a fault, capturing an intruder etc. In this problem, we are given a graph $G(V, E)$ and a set of distinct homebase nodes, the goal for the agents is to eventually meet at a single unspecified node.

Gathering has been studied under various settings concerning the network as well as the agents (ring topologies, anonymous networks, faulty agents, etc.). For our case, the gathering of energy constrained agents at a single node might not always be possible. For this reason, we study the problem of **near-gathering**, where we ask for the agents to relocate in such a way that their maximum or average pairwise distance is minimized.

Collaborative Delivery

In collaborative delivery, given a graph $G(V, E)$ and a set of distinct homebase nodes, the problem consists of moving a single package from its source node to its target node by one or more mobile agents. The package could be a sample collected by a robotic sensor that needs to be delivered to a base station for analysis. One can also imagine an automated postal delivery system where packages need to be delivered between sources and destinations using teams of robots or drones.

The main issue with collaborative delivery is again the restricted supply of energy the agents have and thus, an agent can move only a limited distance before running out of power. However, if we allow the agents to share their energy, multiple agents can cooperate to deliver a package from source to destination. An agent can gather the unused energy of any other agent that it meets. The optimization goal for collaborative delivery is to minimize the total amount of initial energy the agents need to deliver the package.

1.3 Related Work

Graph Exploration

The first algorithm designed for the exploration of a graph by a finite automaton dates back in 1951 and is credited to Shannon [Sha51]. Ever since graph exploration has been extensively studied under a vast number of models. We will try to present an overview of the related work.

Single Agent For the case of a single agent on unknown graphs a very natural optimization goal is the aim of minimizing the exploration time or equivalently the number of edges traversed. Under this setting, one of the most simple and quite efficient algorithms is the Depth First Search (DFS). In the worst case a DFS traversal of a graph needs $2m$ steps, whereas the optimal offline algorithm would need m steps. Therefore, DFS admits a competitive ratio of 2. In [PP99], Panaite and Pelc proposed an algorithm that requires $m + 3n$ steps, which is better than the DFS, given that $3n < m$. In [MMS11] the authors studied the case of weighted graphs and presented a $2w$-competitive algorithm for general graphs with at most w distinct weights. Dessmark and Pelc in [DP04], studied how different types of initial knowledge of the graph impact the efficiency of exploration. More in particular, they showed that if the agent is given an unlabelled isomorphic copy of the graph (*unanchored* map) the optimal competitive ratio for line topologies is $\sqrt{3}$ and for tree topologies is less than 2. On the other hand, if the agent is given an unlabelled isomorphic copy of the graph with the starting node marked (*anchored* map) the optimal competitive ratio for lines is $\frac{7}{5}$ and for trees is $\frac{3}{2}$. In the same work, it was shown that the DFS is optimal for arbitrary graphs even if the agent is given an anchored map of the graph. Fraigniaud et al. in [FIP08] considered a more abstract concept of a map and they showed that $O(\log \log D)$ bits of information are sufficient to achieve competitive ratio smaller than 2 for trees, where D is the diameter of the graph.

If we consider exploration of all edges of directed graphs the problem becomes much harder, as the direction property of the edges allows the adversary to force the agent in traversing multiple times parts of the graph only to explore a single node each time. For this case, Deng and Papadimitriou in [DP99], presented an algorithm with $d^{O(d)}m$ time complexity, where d denotes the deficiency of the graph (the minimum number of edges that have to be added to make the graph Eulerian), they also conjectured the existence of a $poly(d)$-competitive algorithm. Later, Albers and Henzinger presented an improved algorithm with $d^{O(\log d)}m$ time complexity [AH00]. Finally, Fleischer and Trippen proved the conjecture of Deng and Papadimitriou by presenting a $O(d^8)$-competitive algorithm in [FT05]. For the exploration of all nodes of a directed graph, matching lower and upper bounds for the competitive ratio were obtained in [FW16].

From the perspective of minimizing the memory needed by the agent for exploration of anonymous graphs, in a work of Fraigniaud et al. [Fra+05] it was shown that $\Omega(\log n)$ bits of memory are necessary for exploring a graph of size n. In the same work they showed that $\Theta(D \log \delta)$ memory bits are sufficient and necessary for exploring graphs of diameter D and maximum degree δ. In [Dik+04], the authors studied tree topologies and they showed that $\Omega(\log \log \log n)$ memory bits are required for exploration with stop of trees of size n and that $\Omega(\log n)$ bits are needed for exploration with return. In [Amb+11] Ambühl et al. proposed an algorithm for exploration with return that matches the lower bound of $\Omega(\log n)$.

For the exploration of anonymous networks, another direction is to consider that the agent has pebbles to mark nodes in order to recognize them. The goal in that case is to minimize the number of pebbles used. It is easy to show that with a single pebble an agent can traverse an anonymous undirected graph in $O(nm)$ steps. Bender et al., in [Ben+02], showed that if the agent knows an upper bound on the number of nodes, a single pebble is sufficient for exploring in polynomial time $O(n^8\delta^2)$ directed graphs as well. In the same work they also showed that if the agent does not know any upper bound on the size of the graph, then $\Theta(\log \log n)$ pebbles are necessary and sufficient. In [BS94] the authors considered two agents cooperating for the exploration of an unknown graph, where one of the agents acts as a pebble by standing at a node while the other agent explores new nodes.

Multiple Agents For the exploration of graphs using multiple agents, the optimization problem was proved to be NP-hard even for tree topologies in a work of Fraigniaud et al. [Fra+06]. In the same work the authors presented an algorithm for collaborative exploration with return for a team of k mobile agents communicating through whiteboards. The running time of their algorithm is $O(D + n/\log k)$, where D is the height of the tree and its competitive ratio is $O(k/\log k)$. They also showed a lower bound of $\Omega(2-1/k)$ on the competitive ratio for any collaborative exploration algorithm using k agents. In [Hig+14], the authors showed that if we consider only greedy exploration algorithms, the lower bound on the competitive ratio becomes $\Omega(k/\log k)$, that matches the competitiveness of the greedy algorithm presented in [Fra+06]. Ortolf and Schindelhauer in [OS14] adopted a recursive approach and improved the upper bound on the competitive ratio to $O(2^{O(\sqrt{(\log D)(\log \log k)})}(\log k)(\log k + \log n))$ for $k \geq 2^{\omega(\sqrt{\log D \log \log D})}$ and

$n \geq 2^{O(2^{\sqrt{\log D}})}$.

The lower bound on the competitive ratio of any deterministic algorithm using global communication was improved by Dynia et al. in [DŁS07] to $\Omega(\log k / \log\log k)$ for $k < \sqrt{n}$. Disser et al. in [Dis+17] showed that the same lower bound holds for the extended range of $k < n \log^c n$, for every constant $c \in \mathbb{N}$. In the same work they showed that any collaborative exploration algorithm with $k = Dn^{1+o(1)}$ agents has a competitive ratio of $\omega(1)$. This lower bound almost matches the upper bound of Dereniowski et al. in [Der+15]. In that work the authors studied the case for large values of k, i.e., $k = Dn^{1+\epsilon}$, for any $\epsilon > 0$ and they presented a $O(1)$-competitive algorithm. Their solution also works for general graphs where all nodes are within distance D from the starting node.

For a more restricted class of graphs called *sparse* trees, Dynia et al. in [Dyn+06] presented an algorithm with a competitive ratio of $O(D^{1-1/p})$ for trees restricted by parameter p, where p is the density of the tree as defined in that work.

Ortolf and Schindelhauer in [OS12] studied the case of grid graphs with obstacles. They showed that the lower bound of $\Omega(\log k / \log\log k)$ of trees holds for the grid graphs as well. They also showed that for randomized strategies the lower bound for grid graphs changes to $\Omega(\sqrt{\log k} / \log\log k)$.

Energy Aware Agents Betke et al. [BRS95] considered for the first time energy constrained agents in the context of exploration of grid graphs by an agent who can return to its starting node s for refueling (*piecemeal* exploration). The agent is given an upper bound $l = (2+\alpha)r$ on the number of edge traversals it can make before returning to its starting node, where α is some nonnegative constant and r is the distance to the farthest node from s (*radius*). In their work they showed that an agent can perform a piecemeal exploration of a grid graph with rectangular obstacles in linear time. Awerbuch et al. [Awe+99] studied the same problem for general graphs and they presented a nearly linear algorithm where the agent traverses at most $O(|E| + |V|^{1+o(1)})$ edges. In [DKK06], Duncan et al. presented an optimal time algorithm $\Theta(|E|)$ for exploration of weighted graphs under the constraint where the agent is tied to the starting node with a string of fixed length of $l = (1+\alpha)r$ (*tethered*). In the same work the authors showed that the two models, piecemeal and tethered exploration are equivalent within constant factors.

In [DDU17], the authors considered the problem of decomposing a DFS traversal of a weighted tree into a sequence of closed routes (starting and ending at the designated root of the tree) of length at most B, with the goal of minimizing the number of such routes. To this end, they proposed a strategy that achieves asymptotically the minimum number of routes and minimizes the cost up to a small constant, where the cost of a solution is defined as the sum of the lengths of all routes.

When refuelling is not allowed, multiple agents may be needed to explore even graphs of restricted diameter. Given a graph G, determining whether a team of k

agents, each having an energy constraint of B can explore G is known to be an NP-hard problem, even when the graph G is a tree [Fra+06]. Dynia et al. in [DKS06] studied collaborative exploration with return for the case where the number of agents is fixed and the goal is to optimize the amount of energy B required by each agent. They presented an 8-competitive algorithm for trees under the local communication scenario and showed a lower bound of 1.5 on the competitive ratio for any deterministic algorithm. For the same model, the authors in [DŁS07] improved the upper bound by presenting a $(4 - 2/k)$-competitive algorithm.

Prize Collecting Agents In this type of offline exploration, there is a prize or gain associated to each node and the goal is to maximize what the agent collects from visiting the nodes of a graph (not necessarily all of them). Standard examples of problems belonging in this variant of exploration are the Prize Collecting Travelling Salesman Problem (TSP) and the Prize Collecting Steiner Tree (PCST).

In the prize collecting travelling salesman problem, as it was introduced by Balas in [Bal89], we are given an edge-weighted graph G in which each node has an associated prize and penalty, the goal is to collect a given quota of the prizes of the nodes while minimizing the length of the tour plus the sum of penalties of nodes not included in the tour. Bienstock et al. in [Bie+93] considered a version of the problem in which there is no quota of the prizes to be collected. Inspired by the metric TSP which can be approximated within a constant factor, they designed a 2.5-approximation algorithm for the case where the edge costs satisfy the triangle inequality, using linear programming. This result was improved by Goemans and Williamson in [GW95] by presenting a 2-approximation algorithm. Archer et al. in [Arc+11], further improved the result by presenting a $(2 - \epsilon)$-approximation algorithm. The current best known result is by Goemans who presented a 1.91457-approximation algorithm in [Goe09], by combining techniques used in the previous works [Bie+93] and [GW95].

In the work of Bienstock et al. [Bie+93], was also introduced the problem of PCST. In this problem the setting is similar to the prize collecting TSP with the objective, however, of finding a Steiner tree that spans a subset of the nodes such that its cost plus the sum of penalties of nodes not included in the tree is minimized. They presented a 3-approximation algorithm, as an extension of their algorithm for prize collecting TSP. Goemans and Williamson in [GW95], gave a $(2 - \frac{1}{n-1})$-approximation algorithm using a primal-dual scheme with $O(n^3 \log n)$ time complexity, improving the previous result. Johnson et al. in [JMP00] proposed an algorithm with the same approximation ratio of $(2 - \frac{1}{n-1})$, but with better time complexity $O(n^2 \log n)$. However, Feofiloff et al. in [Feo+07] gave a counterexample for which the algorithm of Johnson et al. derives an approximation ratio of 2. In addition, Feofiloff et al. presented a $(2 - \frac{2}{n})$-approximation algorithm with $O(n^2 \log n)$ running time, which is the current best known approximation ratio for PCST.

Johnson et al. in [JMP00] also introduced three variants of PCST, namely, the

Net Worth Maximization problem, the Quota problem and the Budget problem. In the net worth maximization problem the goal is to find a subtree that maximizes the profit, where the profit is defined as the sum of the prizes of nodes included in the solution minus the cost of the subtree. In [FPS01] it was proved that net worth maximization is NP-hard to approximate within any constant factor. In the quota problem the objective is to find a subtree that collects at least a given quota of the prizes while minimizing its cost. The authors showed that any α-approximation algorithm for k-MST can be extended to a pseudopolynomial-time α-approximation algorithm for the quota problem (Garg in [Gar05] gave a 2-approximation algorithm for k-MST, which is the best current known). Last, for the budget problem the goal is to find a subtree that maximizes the prizes of nodes included in the solution with a given bound cost. They showed that the budget problem can be solved by the quota problem using binary search achieving a $(5+\epsilon)$ approximation ratio.

Movement Problems

In movement problems, we are given a graph $G(V, E)$ and the initial placement of a group of k mobile agents on some nodes of G, the goal is to move the agents so as to obtain a desired final configuration while minimizing the maximum or the total movement of the agents. Movement problems have been widely studied in mobile agent computing mainly for the distributed setting. In survey [PS06], the authors describe the algorithmic capabilities and limitations of autonomous mobile agents with respect to movement problems. For the centralized setting, the first work was presented by Demaine et al. in [Dem+09]. They considered the problems of *connectivity*, where the induced subgraph by the agents' final locations is connected, s-t connectivity, where the induced subgraph by the agents' final locations contains nodes s and t in one connected component, *independence*, where the final locations of the agents should form an independent set and *matching*, where the pairwise distance of the agents in the final configuration, is at most one. They proved several approximation and inapproximability results for the considered problems, apart from the matching problem, for which they showed that it admits a polynomial time algorithm for both optimization goals. For the maximum movement version of the connectivity and s-t connectivity problems they showed lower bounds of 2 and upper bounds of $O(1+\sqrt{k/OPT})$, where OPT is the cost of an optimal solution. Berman et al. in [BDZ11], improved these upper bounds by presenting constant factor approximation algorithms for both problems.

In [Dem+09], the authors also suggested the *facility-location* movement problem where there are two types of agents, namely *clients* and *servers*, and the target property is that every client is located at a node that contains a server. They presented a 2-approximation algorithm for the maximum movement version, but they left open the total movement version. Friggstad and Salavatipour in [FS11] proved a lower bound of 2 for the maximum movement facility location, proving the algorithm of Demaine et al. tight, and they presented an 8-approximation algorithm, based on LP-rounding, for the total movement objective of facility location. This upper bound was improved by Ahmadian et al. in [AFS13] by presenting a $(3+\epsilon)$-approximate algorithm, where $\epsilon > 0$.

Bilò et al. in [Bil+13] studied the gathering problem for the case of agents moving in a vertex-to-vertex manner in the visibility graphs of simple polygons. They showed that minimizing either the maximum or the total movement is polynomial time solvable.

In [Bil+16], the authors studied the *clique* problem, in which we ask that the induced graph by the agents' final locations forms a clique in G. They showed that minimizing the maximum or the total movement is NP-hard. Moreover, they presented an approximation algorithm for the maximum movement version which is optimal except for an additive term equal to 1 and a 2-approximation algorithm for the total movement version.

In [DHM14] for the case of small number of agents (compared to the size of the graph) the authors considered the fixed-parameter tractability. They showed that the complexity of movement problems depends on the treewidth of their minimal configurations (the configuration of a feasible solution where the deletion of an edge destroys the solution).

Collaborative Delivery and Related Problems

Collaborative delivery by energy-aware mobile agents, is a problem that has been considered in some recent works under various assumptions. In [Cha+13], the authors proved that the problem of deciding whether delivery is feasible is NP-hard even if the agents are initially collocated, and they provided a 2-approximation algorithm for the optimization version of finding the minimum initial energy that can be given to all agents so that delivery becomes feasible, as well as exact, approximation, and resource-augmented algorithms for variants of the problem. Chalopin et al. in [Cha+14], showed that the problem is weakly NP-hard even on the line (with initially dispersed agents), and provided a quasi-, pseudo-polynomial algorithm under the assumption of integer numerical values in the problem instance.

In [Bär+16], the authors considered the variant of the delivery problem in which the agents have to return to their respective starting positions and they proved that this problem is NP-hard for planar graphs but can be solved efficiently on trees and lines, in contrast to the non-returning version which is NP-hard on lines. They also gave resource-augmented algorithms for returning delivery in general graphs and proved tight lower bounds on the resource augmentation for both the returning and the non-returning variant.

Bärtschi et al. in [Bär+17], considered agents that do not have a limited energy source, but instead they have different rates of energy consumption and the goal is to find a delivery schedule that minimizes the total energy spent. Moreover, there are several messages that need to be delivered from their respective source to their respective target. The authors studied separately three subtasks that need to be solved in order to compute the optimal solution (collaboration of different agents on the same message, planning for an agent that works on multiple messages, and assignment of messages to agents) and they provided a polynomial-time (non-constant) approximation algorithm for the problem. In this setting, the authors in [BGP17] studied the design of truthful mechanisms in a game-theoretic model where the rate of energy consumption is information private to each agent. Bärtschi and Tschager in [BT17], considered the case where the agents have different rates of energy consumption as well as different rates of travelling and they were interested in finding a schedule that simultaneously minimizes the total energy consumption $\mathcal{E}$ and the delivery time $\mathcal{T}$. They showed that minimizing lexicographically the tuple $(\mathcal{E}, \mathcal{T})$ is polynomial time solvable.

Broadcast and convergecast are two closely related problems. In the broadcast problem a specified agent has some piece of information that has to become available to all other agents. In the convergecast problem each agent initially has some piece of information and every piece of information needs to be collected by some agent. Anaya et al. in [Ana+16] studied for both centralized and distributed settings the broadcast and convergecast problems. They showed hardness results for tree topologies and presented approximation algorithms for arbitrary graphs. Czyzowicz et al. in [Czy+17] studied the variant of broadcast in which a piece of information is initially stored at a node of a graph and it needs to be broadcasted to all other nodes by mobile agents while minimizing the total energy used by all agents. In their work they considered tree topologies and they presented an algorithm with $O(n \log n)$ time complexity for solving the problem optimally.

The only previous work that considers energy sharing by mobile agents is [Czy+16], where this feature is introduced in the context of the delivery and convergecast problems. The authors showed that both problems can be solved efficiently in trees, whereas they are NP-complete in general undirected and directed graphs. It is important to note that, in the model of [Czy+16], an agent may store an *unlimited* amount of energy as a result of receiving energy from other agents it encounters. In other words, there is no battery capacity constraint for the agents.

1.4 Overview of the Thesis

This thesis is organized as follows.

In Chapter 2, we consider the exploration with no return of an unknown tree by a team of energy constrained mobile agents. Each agent has a fixed amount of initial energy and the optimization goal is to minimize the total number of agents used. We study this problem under two communication models. First, we consider that the agents have global communication, meaning that the agents communicate with each other instantaneously and we provide an efficient algorithm for the problem. Next, we consider the local communication model, in which the agents have to be collocated in order to share information. We modify the previous algorithm in order to work under this model, achieving the same asymptotic cost with a small multiplicative overhead. Finally, we give a lower bound for the local communication model that matches the cost of our algorithm, proving that our result is tight.

In Chapter 3, we consider the exploration with no return of an unknown tree by a fixed number of agents, where each agent has a fixed amount of initial energy. Since the number of agents and their initial energy are fixed there is no guarantee that we can explore the entire tree, for this reason, the optimization goal in this case is to maximize the number of explored nodes. Here, we only consider the global communication model. We present an efficient algorithm that is constant competitive with respect to the offline optimal algorithm. We also prove a lower bound on the competitive ratio for the problem.

In Chapter 4 we consider the exploration with return of known trees by a single agent which can collect energy from the nodes that it visits, enabling it to continue the exploration. We study the problem in weighted trees where the weight of an edge represents the amount of energy the agent has to spend in order to traverse it and the weight (gain) of each node represents the amount of energy that the agent can collect upon visiting the node for the first time. The goal in this problem is to maximize the amount of collected energy the agent brings back to the root at the end of the exploration. The agent is not required to visit all the nodes of the tree and indeed an optimal solution might not visit every node of the tree, in order to maximize the collected gain. We prove that this problem is NP-hard even in trees. Next, we show that if we provide the agent with sufficient initial energy we can solve the problem efficiently and we present an algorithm for collecting the maximum gain. Furthermore, we present an algorithm that computes the minimum initial energy the agent needs in order to achieve the maximum collected gain.

In Chapter 5, we consider the problem of near-gathering where the optimization goal is to relocate the agents, in such a way that the maximum or the average distance between any two agents is minimized. We study the problem in general graphs and prove that computing a strategy for minimizing their maximum pairwise distance is NP-hard to approximate within a factor of $2 - o(1)$ and we provide a matching approximation algorithm. We also provide a constant factor approximation algorithm for the problem of minimizing the average pairwise distance between agents.

In Chapter 6, we consider the collaborative delivery problem assuming the agents can share their energy upon meeting, under two scenarios. In the first scenario, we allow the agents to share their energy without any restrictions, while in the second scenario, we assume each agent has restricted energy capacity, such that at any moment, the amount of available energy an agent has, cannot be greater than the amount of energy it started with. For the first scenario we prove that the problem is NP-hard and we show that if we augment the initial energy of the agents by a constant factor, then we can solve the problem optimally. For the second scenario, we show that the problem is NP-hard even when the agents have initial energy equal to 2 if the initial positions of the agents is given as part of the input of the problem. On the other hand, if the algorithm can choose the initial placement of the agents from a given subset of starting nodes, then we show that we can solve the problem efficiently and we provide a solution strategy using the minimum number of agents.

Last, we finish this document with a chapter of conclusions, where we summarize the results presented in this thesis and we also propose some perspectives for further research in the topic of energy constrained mobile agents.

2 Online Exploration of Trees

Introduction

We consider the problem of exploration of an unknown tree T by a team of mobile agents initially located at the root of the tree. Each agent is equipped with a battery of size B which bounds the total number of edges the agent can traverse during its lifetime. The agents do not have to return to the root. We assume the height of the tree, i.e., the longest path between the root and a leaf, to be at most $B-1$, and our objective is to find an exploration strategy where every node of the tree is visited by at least one agent, and we wish to minimize the total number of agents used. If the height of T is greater than B, then T cannot be fully explored, even by an unbounded number of agents. Moreover, if the tree has height greater than B, this fact cannot be detected at the root of the tree. On the other hand, if the height of the tree is exactly B then each leaf at depth B must be visited by a separate agent. Once the tree is completely explored and known up to depth $B-1$ then we can send one additional agent to explore each leaf at depth B. Observe that any optimal offline algorithm would also need to send one agent per leaf at depth B. Thus, we can safely ignore these agents, as this would not increase the cost of the algorithm by more than a factor of 2. So, we will be considering algorithms for exploring trees of height at most $B-1$.

We study this problem first assuming a global communication model (where agents communicate to each other instantaneously) and provide an algorithm for online exploration, that has a competitive ratio of $O(\log B)$. We then show how to remove the assumption of global communication and achieve the same result in the local communication model, with a constant overhead. Finally we provide a lower bound of $\Omega(\log B)$ on the competitive ratio of any online exploration algorithm for energy constrained agents in the local communication model, showing that our result is tight.

The results presented in this chapter are accepted for publication in [DDKar]. A preliminary version of these results has appeared in the conference publication [DDK15].

The Model

The environment to be explored is a rooted tree T. The root r_0 contains an infinite supply of mobile agents, each of which has a limited energy B, allowing it to traverse at most B edges during its lifetime. There is a total order among the agents (i.e., they have distinct identities). The nodes of the tree may be assumed to be anonymous (i.e., we do not require unique identifiers for the nodes of T). Each agent has unlimited memory. When two agents are at the same node, they can freely exchange information. However the agents may not write any information on the nodes of the tree [1]. We call this the *local communication* model. In contrast, in a *global communication* model an agent can communicate instantaneously with any other agent irrespective of their location in the tree.

All agents start at the same time, in the same state. The agents move synchronously. At each time unit, any agent can move to an adjacent node or stay at its current node. Each move costs one unit of time and one unit of energy, while computation and communication between agents are instantaneous and do not consume any energy. The agents cannot share their energy resources or recharge their batteries.

The height of the tree (i.e., the distance to the furthest leaf from the designated root r_0) is at most $B-1$ and this information is known to the agents. However, the size (# nodes in the tree) and the structure of the tree is initially unknown to the agents. The edges incident at each node are locally ordered with port numbers, allowing the agents to choose edges to visit in a deterministic manner. An exploration strategy for the team of agents is successful if each node of the tree is visited by at least one agent. The cost of the exploration strategy is the number of agents which made any non-null moves during the exploration. We denote by OPT the cost of the optimal offline strategy that has complete knowledge of the tree.

For any node $r \in T$ we denote by T_r the subtree of T, rooted at r. Further for any node $v \in T_r$, we define the depth of node v as the length of the path from r to v. We denote by T_r^δ the subtree rooted at r truncated to depth δ from r. We denote by $|T|$, the number of edges in T.

2.1 Exploration with Global Communication

In this section we describe and analyze a recursive algorithm for tree exploration under the global communication model. The algorithm is called *Global Communication Tree Exploration* ($\mathtt{GCTE}_\varepsilon$). The main idea of the algorithm is to explore the tree up until a certain depth and afterwards take advantage of the already known part of the tree to continue the exploration. More specifically, this algorithm proceeds by *levels*. Each level of the algorithm is a set of nodes which are located at a certain depth of the tree. The first level consists of the root r_0. At each level i, agents having energy b_i, expand the explored part of the tree further by increasing its depth by $\lceil \varepsilon \cdot b_i \rceil$ where ε is a parameter of the algorithm such that $0 < \varepsilon < \frac{1}{4}$. The new frontier of the explored part defines the next level of the algorithm. The algorithm $\mathtt{GCTE}_\varepsilon$ is then recursively called at each node r of the newly created level i. Whenever an agent is needed at a node r, one agent from the global root r_0 arrives at r to execute $\mathtt{GCTE}_\varepsilon$

[1]Note that the only way to store information at a node is by having an agent stay at the node carrying this information

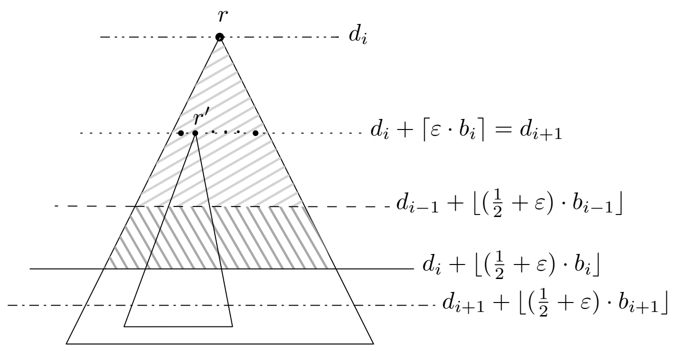
r
d_i
r'
$d_i + \lceil \varepsilon \cdot b_i \rceil = d_{i+1}$
$d_{i-1} + \lfloor (\frac{1}{2} + \varepsilon) \cdot b_{i-1} \rfloor$
$d_i + \lfloor (\frac{1}{2} + \varepsilon) \cdot b_i \rfloor$
$d_{i+1} + \lfloor (\frac{1}{2} + \varepsilon) \cdot b_{i+1} \rfloor$

Figure 2.1: The explored subtrees for two consecutive calls to $\texttt{GCTE}_\varepsilon$: the first for node r at level i and the next recursive call for r' at level $i+1$. The first shaded region is the explored part of T_r and the second shaded region is the part explored during level i.

Definition 1. For $i = 1$, *level* i of algorithm $\texttt{GCTE}_\varepsilon$ consists of the root node r_0; the depth d_i of the level i is $d_1 = 0$, the energy b_i at this level is $b_1 = B$. For $i > 1$, *level* i of $\texttt{GCTE}_\varepsilon$ consists of all nodes at depth $d_i = d_{i-1} + \lceil \varepsilon \cdot b_{i-1} \rceil$, and $b_i = B - d_i$.

For any two nodes u and v at the same level, we would like the exploration of the trees T_u and T_v to proceed independently, using disjoint sets of agents. To this end, we allow some overlap between successive levels of the algorithm. More precisely, at each level i, the exploration is extended to depth of $\left\lfloor (\frac{1}{2} + \varepsilon) \cdot b_i \right\rfloor$, although the next level still starts at depth $\lceil \varepsilon \cdot b_i \rceil$ from the current level. This additional extension at each level i allows the algorithm to *look ahead* at the start of the next level $(i+1)$. Thus, at the start of a recursive call to Algorithm $\texttt{GCTE}_\varepsilon$ at a node r at level $i+1$, the subtree T_r has been already partially explored to some depth. We show below (c.f. Lemma 3) that the exploration of this partially explored subtree T_r can be done independently to any other subtree at the same level.

Definition 2. Two partially explored subtrees T_u and T_v, rooted at nodes u and v located at the same depth from r_0, are said to be *independent* if no single agent can visit nodes in the unexplored part of both subtrees.

Informally, this independence means that disjoint teams of agents can be used for exploring such subtrees during the algorithm. See Figure 2.1, which illustrates the 'overlaps' of explored parts of the subtrees for two consecutive calls to $\texttt{GCTE}_\varepsilon$.

We now formally describe our algorithm $\texttt{GCTE}_\varepsilon$.

For an execution of $\texttt{GCTE}_\varepsilon$ with parameters r and b, we define $x(\varepsilon) = \frac{1}{2}(\frac{1}{2} - \varepsilon)b$. Each agent saves $\lceil x(\varepsilon) \rceil$ units of energy which is never used during $\texttt{GCTE}_\varepsilon$. This energy will be used later as explained in the next section.

Procedure $\texttt{Uncover}(r, \delta)$ with input node r and an integer δ works as follows. During this procedure, the agents explore the unexplored part of subtree T_r rooted

Algorithm 1 GCTE$_\varepsilon$: An algorithm for tree exploration, $0 < \varepsilon < \frac{1}{4}$

```
Input: The root $r$ of the tree and an integer $b$ that equals the size of the available energy the agents have.
1: Uncover$(r, \lfloor (\frac{1}{2} + \varepsilon) b \rfloor)$
2: Let $r_1, r_2, \dots$ be nodes at depth $\lceil \varepsilon \cdot b \rceil$ from $r$, such that $T_{r_i}$ has some unexplored edges.
3: For each $r_i$, call Algorithm GCTE$_\varepsilon(r_i, (b - \lceil \varepsilon \cdot b \rceil))$.
```

at r, using a Depth First Search (DFS) traversal restricted to a depth of δ from r. This *DFS traversal* is defined as the walk followed by an agent of infinite energy starting at node r and performing DFS on the tree T' that is obtained from T_r^δ by pruning all subtrees that have been already explored (before the call to `Uncover`). An agent initially located at the root r_0 arrives at the current root r, having b units of energy and begins to explore the subtree T_r^δ. First, this agent goes to the unexplored node that is supposed to be visited next in the DFS traversal. From this node, the agent follows the DFS traversal. Finally, when the agent has $\lceil x(\varepsilon) \rceil$ units of energy left, it interrupts the exploration. Note that in the global communication model, at any point of the exploration, each agent possesses the full knowledge of the part of the tree explored to date and the current locations of all agents. Hence, another agent will arrive at r and continue the exploration by visiting the unexplored node that is supposed to be visited next according to the DFS traversal. This procedure ends when all nodes at depth δ or less have been visited.

Lemma 3. *The subtrees that are created in step 2 of* GCTE$_\varepsilon$ *are pairwise independent. Moreover, for any such subtree T_r rooted at a node r any agent that explores any edge in the unexplored part of T_r cannot return to node r.*

Proof. Consider the subtree T_r rooted at r at the start of procedure `Uncover`(r, δ) at some level $i > 1$ of the algorithm. Let E'_r denote the set of unexplored edges of T_r. By construction, we have that subtree T_r is already explored until depth $d \geq \lfloor \frac{1}{2} \cdot b_{i-1} \rfloor$ from r and thus any edge in E'_r is at distance at least d from r. Any agent located at r has at most b_i units of energy and $d \geq \lfloor \frac{1}{2} \cdot b_{i-1} \rfloor \geq \lfloor \frac{1}{2} \cdot b_i \rfloor$. As a result, the agent will use at least $d + 1$ units of its available energy to explore any edge in E'_r, where $d + 1 > \frac{1}{2} \cdot b_i$, thus, such an agent would not have sufficient energy to return to the root r and subsequently to reach any other subtree rooted at the same level.

□

Theorem 4. *For any ε, $0 < \varepsilon < \frac{1}{4}$, Algorithm* GCTE$_\varepsilon$ *called with parameters r_0 and B correctly explores the tree.*

Proof. To prove the correctness of 1, we will first show that procedure `Uncover`(r, δ) called for a subtree T_r rooted at r at some level $i \geq 1$ with $\delta = \lfloor (\frac{1}{2} + \varepsilon) b_i \rfloor$ correctly explores the subtree T_r up to depth δ. In order to prove this, we will show that each agent has enough energy to reach an unexplored edge. Note that by a simple induction on the distance of r from r_0, any agent that arrives at node r, to execute `Uncover`(r, δ), has exactly b_i units of energy. Further any such agent a has complete knowledge of the part of subtree T_r already explored by previous agents and thus agent a knows the path from r to the next unexplored node v in the DFS traversal of T_r^δ. This node v is at distance at most δ from r. According to the algorithm, agent a has

$$l := b_i - \lceil x(\varepsilon) \rceil = \lfloor b_i - x(\varepsilon) \rfloor \geq \delta + \lfloor x(\varepsilon) \rfloor$$

units of energy available for the DFS traversal. Since $l \geq \delta$ the agent does succeed in reaching the node v. Hence, each agent used in `Uncover` visits at least one previously unexplored node in T_r^δ. As long as there are unexplored edges we keep sending agents, where this implies that eventually all nodes within depth δ in T_r are visited during the DFS exploration. This proves the correctness of procedure `Uncover`.

In order to complete the proof of the correctness of `GCTE`$_\varepsilon$, we note that the algorithm makes progress at each level i, that is, level $i+1$ is at strictly greater depth than level i. Indeed, this follows from $\varepsilon b_i > 0$ for $\varepsilon > 0$, which gives $\lceil \varepsilon b_i \rceil \geq 1$. □

Lemma 5. *The number of levels in Algorithm* `GCTE`$_\varepsilon$ *is at most* $\log_{(\frac{1}{1-\varepsilon})} B$.

Proof. For the purpose of this proof introduce two variables d'_i and b'_i as follows: $d'_1 = 0$ and $d'_i = d'_{i-1} + \varepsilon \cdot b'_{i-1}$ for $i > 1$, where $b'_i = B - d'_i$ for $i \geq 1$.

We first argue by induction on i that $d'_i = B - (1-\varepsilon)^{i-1}B$ for $i \geq 1$. This claim trivially holds for $i = 1$ so take now some $i \geq 1$ and assume that $d'_i = B - (1-\varepsilon)^{i-1}B$. We obtain

$$\begin{aligned} d'_{i+1} = d'_i + \varepsilon b'_i = d'_i + \varepsilon(B - d'_i) = (1-\varepsilon)d'_i + \varepsilon B \\ = (1-\varepsilon)\Big(B - (1-\varepsilon)^{i-1}B\Big) + \varepsilon B = B - (1-\varepsilon)^i B \end{aligned}$$

as required.

Since $d_i \geq d'_i$ for each $i \geq 1$, we have proved that $d_{i+1} \geq B - (1-\varepsilon)^i B$ for each $i \geq 1$. Observe now, that if $(1-\varepsilon)^i \cdot B \leq 1$, then the i-th call is the last recursive call to 1, i.e., the subtrees rooted at level $i+1$ have no unexplored edges. This holds due to the fact that the height of the tree is no more than $B - 1$. Therefore, the inequality $(1-\varepsilon)^i \cdot B \leq 1$, gives us an upper bound on the maximum level number i, which gives at most $\log_{(\frac{1}{1-\varepsilon})} B$ levels. □

Before proceeding to calculating the cost of Algorithm `GCTE`$_\varepsilon$, let us make the following useful remark. During the procedure `Uncover`, each participating agent uses at most $\delta - 1$ energy to reach the starting node for its DFS exploration and uses at least $b - (\delta - 1) - \lceil x(\varepsilon) \rceil \geq \left\lceil \frac{1}{2}(\frac{1}{2} - \varepsilon)b \right\rceil$ units of energy to contribute to the DFS exploration of unexplored nodes.

Lemma 6. *Procedure* `Uncover`*(r,δ) for* $r = r_0$ *and* $\delta = \lfloor (1/2 + \varepsilon)B \rfloor$ *uses* SOL_r *agents where* $SOL_r \leq \frac{4}{(\frac{1}{2}-\varepsilon)} \cdot OPT$.

Proof. In this case, T_r^δ is completely unexplored at the start of the procedure. By the properties of DFS exploration, the DFS walk of a tree T_r^δ is of length $2 \cdot |T_r^\delta|$. Each agent (except possibly the last one) uses at least $\left\lceil \frac{1}{2}(\frac{1}{2} - \varepsilon) \cdot B \right\rceil$ of its available energy to traverse a part of this walk. Thus, we get the following inequality:

$$\left\lceil \frac{1}{2}\left(\frac{1}{2} - \varepsilon\right) \cdot B \right\rceil \cdot SOL_r \leq 2 \cdot |T_r^\delta| \leq 2 \cdot |T|$$

In addition, we have that $|T| \leq OPT \cdot B$, since the optimal algorithm must visit all the edges of the tree T. Hence, we get

$$\left\lceil \frac{1}{2}\left(\frac{1}{2} - \varepsilon\right) \cdot B \right\rceil \cdot SOL_r \leq 2 \cdot OPT \cdot B$$

Finally, we have that $\frac{1}{2}\left(\frac{1}{2} - \varepsilon\right) \cdot B \leq \left\lceil \frac{1}{2}\left(\frac{1}{2} - \varepsilon\right) \cdot B \right\rceil$, therefore

$$SOL_r \leq \frac{4}{\frac{1}{2} - \varepsilon} \cdot OPT$$

□

Theorem 7. *Algorithm* $\texttt{GCTE}_\varepsilon$ *has a competitive ratio of at most* $\frac{4}{(\frac{1}{2}-\varepsilon)} \cdot \log_{(\frac{1}{1-\varepsilon})} B$.

Proof. Consider a call to $\texttt{GCTE}_\varepsilon(r, b_i)$ at some level $i > 1$, where r is at depth $d_i > 0$ from the global root. Let SOL_r denote the number of agents used by the algorithm to explore edges of the subtree T_r during level i. A DFS exploration walk of T_r that starts and ends at r has length $2 \cdot |T_r|$. As explained before each of the SOL_r agents (except the last one) use at least $\left\lceil \frac{1}{2}(\frac{1}{2} - \varepsilon)b_i \right\rceil$ of their available energy to contribute to the DFS exploration. The last agent may have some available energy after visiting the last unexplored edge in T_r but it does not have enough energy to return to node r (by Lemma 3). Therefore, if we assume that the last agent attempts to reach the root r with its remaining energy, we can say that the path traversed in total by the agents is less than $2 \cdot |T_r|$. Thus,

$$\frac{1}{2}(\frac{1}{2} - \varepsilon)b_i \cdot SOL_r \leq \left\lceil \frac{1}{2}(\frac{1}{2} - \varepsilon)b_i \cdot SOL_r \right\rceil \leq 2 \cdot |T_r|$$

$$\implies SOL_r \leq \frac{4}{b_i(\frac{1}{2} - \varepsilon)}|T_r|$$

Furthermore, due to Lemma 3, we know the subtrees at the same level are independent so we can sum up over all subtrees at level i:

$$\sum_{r \in r_1, r_2, \ldots} SOL_r \leq \frac{4}{b_i(\frac{1}{2} - \varepsilon)} \sum_{r \in r_1, r_2, \ldots} |T_r|$$

$$SOL(i) \leq \frac{4}{b_i(\frac{1}{2}-\varepsilon)}|T \setminus T_{r_0}^{d_i}|$$

where $SOL(i)$ denotes the number of agents used by the algorithm at level i. The optimal algorithm uses OPT agents to explore the tree. Any agent that reaches to depth d_i of T has b_i units of energy remaining. Thus, each agent can traverse at most b_i edges below this depth. Hence

$$b_i \cdot OPT \geq |T \setminus T_{r_0}^{d_i}|$$

Combining the above two equations, we have

$$SOL(i) \leq \frac{4}{\frac{1}{2}-\varepsilon}OPT$$

The above bound holds for any level $i > 1$. Moreover, due to Lemma 6, we have exactly the same bound for level $i = 1$ of the algorithm. Since there are at most $\log_{(\frac{1}{1-\varepsilon})} B$ levels in the algorithm (due to Lemma 5), we obtain the total cost SOL of the algorithm,

$$SOL \leq \frac{4}{\frac{1}{2}-\varepsilon} \cdot \log_{(\frac{1}{1-\varepsilon})} B \cdot OPT$$

□

Note that on the termination of algorithm $\texttt{GCTE}_\varepsilon$, each agent that participated in the exploration at level i has at least $x_i(\varepsilon) = \frac{1}{2}(\frac{1}{2}-\varepsilon) \cdot b_i$ units of unused energy. This remaining energy would be used by the algorithm presented in the next section.

2.2 Exploration with Local Communication

This section is devoted to adaptation of $\texttt{GCTE}_\varepsilon$ for the model with local communication between agents. This is done in two steps. In the first step we introduce an intermediate stage between two models of global and local communication. We call this a *semi-local communication model* and we define it as follows: two agents performing the DFS exploration in Step 1 of an instance of $\texttt{GCTE}_\varepsilon$ can communicate only locally, that is, they can communicate only when present at the same node; on the other hand, whenever an agent is needed at a node r, one agent arrives from the global root at r to execute the instance of $\texttt{GCTE}_\varepsilon$. Thus, in our semi-local communication model this mechanism of calling for agents uses the global communication model. In Section 2.2.1 we adopt $\texttt{GCTE}_\varepsilon$ so that it operates in the semi-local communication model and we calculate the cost of this modification in terms of the number of agents used. In particular, we prove that with respect to the original algorithm, the total number of agents increases by a constant factor (depending only on ε). Then, in Section 2.2.2, we add to our algorithm a mechanism for calling for new agents at local roots so that this part is also done via local communication.

2.2.1 Semi-local communication model

We start this section by providing some intuition. We consider an arbitrary execution of $\texttt{GCTE}_\varepsilon(r, b)$ for an input node r and energy level b. Recall Step 1 of $\texttt{GCTE}_\varepsilon$, where the agents, one by one, perform the DFS traversal up to a certain depth of the subtree T_r. Suppose that the agents that perform this traversal are $a_1, \ldots, a_k$ and that they are ordered according to the precedence of their movements, i.e., a_i traverses its path prior to a_{i+1} for each $i \in \{1, \ldots, k-1\}$. For each agent a_i we will add 5 additional agents denoted $a_i^1, \ldots, a_i^5$. To simplify some statements we sometimes write a_i^0 in place of a_i. The agents $a_i, a_i^1, \ldots, a_i^5$ are called the *i-th team* for each $i \in \{1, \ldots, k\}$. For the purposes of the analysis we introduce some additional notation that allows us to describe the behavior of agents during this DFS traversal in more details. As mentioned earlier, we denote

$$x(\varepsilon) = \frac{1}{2}\left(\frac{1}{2} - \varepsilon\right) b \tag{2.1}$$

We also say that an agent *heads towards* a node v if in each of the following consecutive time units the agent makes a move that gets it closer to v until either v is reached or the agent runs out of energy. It is said that the i-th team is *successful* if: (i) the agent a_i visited a superset of nodes with respect to its original behavior in Step 1 of $\texttt{GCTE}_\varepsilon$, and (ii) the agent a_i^1 reaches the root r and possesses the information about all moves performed by agents $a_1, \ldots, a_i$.

We now describe the modification of the DFS traversal from Step 1 of $\texttt{GCTE}_\varepsilon$ by describing how a_i and $a_i^1, \ldots, a_i^5$ operate for each $i \in \{1, \ldots, k\}$.

Behavior of a_i. Recall that in Step 1 of $\texttt{GCTE}_\varepsilon$, each agent a_i, $i \in \{1, \ldots, k\}$, finishes its part of DFS traversal having at least $\lceil x(\varepsilon) \rceil$ energy left. We now use this energy as follows: the agent heads towards the root r in the next $\lceil x(\varepsilon) \rceil$ time units.

Behavior of a_i^j's. For each $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, 5\}$, the agent a_i^j follows the movements of a_i up to depth

$$d_j(\varepsilon) = \lfloor j \cdot x(\varepsilon) \rfloor$$

until the completion of the movement of a_i. More precisely, the agent a_i^j mimics each move of a_i from node u to node v if both u and v are within depth (from r) at most $d_j(\varepsilon)$. If, on the other hand, either u or v is at depth greater than $d_j(\varepsilon)$, then a_i^j stays idle during any such move of agent a_i. Finally, the agent a_i^j heads towards the root r; we will describe below in which time step this action is triggered.

Order of movements. Having described the movements of a_i and a_i^j for each $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, 5\}$, we specify the order of their actions. The agent a_1

starts its movement once all agents of the 1-st team are at r. For each $i \in \{2, \dots, k\}$, the agent a_i starts its movement once a^1_{i-1} completed its movement by arriving at r and once all agents of the i-th team are at r. (We will argue later that a^1_{i-1} indeed returns to the root r.) In other words, once a^1_{i-1} completes its movement, all agents of the i-th team are called to appear at r. For each $j \in \{1, \dots, 5\}$, we only need to describe how they operate once a_i runs out of energy, as their preceding movements are specified above. The agent a^j_i heads towards the root r in time unit in which he occupies the same node as a^{j+1}_i and the latter agent is heading towards r. Thus, it may happen that for some units of time both agents will head towards r together.

In the following we prove that the above actions of agents are valid under the assumption that they have to communicate locally. Considering the order of movements of agents it suffices to argue that each team is successful. We refer to all movements of the agents a^j_i, $i \in \{1, \dots, k\}$, $j \in \{1, \dots, 5\}$, as the *extended DFS traversal* of T.

Lemma 8. *For each $i \in \{1, \dots, k\}$, the i-th team is successful.*

Proof. We prove the lemma by induction on i. The argument is identical for $i = 1$ and $i > 1$ so take an arbitrary $i \in \{1, \dots, k\}$ and inductively assume that the $(i-1)$-th team is successful whenever $i > 1$. If $i = 1$, then clearly the 1-st team needs no additional information about the subtree. By assumption, when $i > 1$, a^1_{i-1} completes its movement reaching the root r and this agent knows the entire subtree explored by $a_1, \dots, a_{i-1}$. Thus, once a_i is about to start its actions at r, it learns the subtree explored to date: for $i > 1$ this information comes from a^1_{i-1} and for $i = 1$ it is known that the DFS traversal will be initiated by a_i. Thus, a_i can indeed correctly resume in Step 1 of $\texttt{GCTE}_\varepsilon$ the DFS traversal and can correctly complete all its following actions as they do not depend on any extra knowledge. Hence, it remains to analyze the behavior of the remaining agents of the i-th team. Note that, by algorithm definition, all agents of the i-th team are present at the root r once a_i is about to start its movements. Thus, each agent a^j_i can correctly mimic the movements of a_i till the depth predefined for a^j_i.

Suppose that a_i reaches a node v at depth l with its last move; the move that ends the stage when it is heading towards r. According to the algorithm of the extended DFS traversal, a_i was heading towards r and the duration of this stage was $\lceil x(\varepsilon) \rceil$. Thus, these movements of a_i of heading towards r started at a node v' at depth $l + \lceil x(\varepsilon) \rceil$, at time unit t', ended at depth l and node v, and there exists a node u at depth $d_j(\varepsilon)$ on the path traversed during these movements of heading towards r. By construction, agent a^j_i is at node u at time unit t'.

At time step t', each agent $a^{j'}_i$, $j' \in \{1, \dots, 5\}$, has at least $\lceil x(\varepsilon) \rceil$ units of energy available. Indeed, this follows from two observations: first, the available energy of $a_i = a^0_i$ is at least $\lceil x(\varepsilon) \rceil$ at time step t'; second: the length of the path traversed by $a^{j'-1}_i$ till time step t' is not greater than the length of the path traversed by $a^{j'}_i$

till time step t', for each $j' \in \{1, \ldots, 5\}$. Furthermore, at time step t', the distance between any two consecutive agents $a_i^{j'}, a_i^{j'+1}$ for $j \in \{1, \ldots, 4\}$ is not greater than $\lceil x(\varepsilon) \rceil$ since $\lfloor (j+1) \cdot x(\varepsilon) \rfloor - \lfloor j \cdot x(\varepsilon) \rfloor \leq \lfloor (j+1) \cdot x(\varepsilon) - j \cdot x(\varepsilon) \rfloor + \alpha = \lfloor x(\varepsilon) \rfloor + \alpha$, where $\alpha < 1$. Thus, in particular, each agent $a_i^{j'}$ has enough energy to reach the node occupied by $a_i^{j'-1}$ for each $j' \in \{2, \ldots, 5\}$. Hence, by a simple induction on $j' = j-1, \ldots, 1$ we obtain that $a_i^{j'+1}$ meets $a_i^{j'}$ at some time step and in this step $a_i^{j'}$ has at least $\lceil x(\varepsilon) \rceil$ units of energy and is present at depth $\lfloor j' \cdot x(\varepsilon) \rfloor$ in T. Thus, we obtain that a_i^1 is able to reach r (at depth 0 of T_r) carrying the required information.

□

As a consequence of the above, we obtain the following.

Lemma 9. *The extended DFS traversal correctly explores T_r to a depth of $(\frac{1}{2} + \varepsilon)B$ using $6k$ agents that communicate locally, where k is the number of agents used in the DFS traversal performed in Step 1 of Algorithm* `GCTE`$_\varepsilon$.

Proof. The fact that the k teams, each of size 6, explore the tree to the required depth follows directly from Lemma 8.

□

2.2.2 Local communication between levels

We start this section with an informal description, also pointing out the obstacles we need to overcome. The mechanism of communication between two consecutive levels will be handled by special agents that we call *managing agents* (see below for a formal definition). A managing agent arrives at a root r for which a call to `GCTE`$_\varepsilon$ is performed. This agent is not used for the extended DFS traversal of T_r but will play a crucial role while conducting recursive calls for descendants $r_1, r_2, \ldots$. More precisely, this agent will keep track of which subtrees have been already explored and for which one the recursive call is 'in progress'. By a recursive call, made say for r_i, being in progress we mean that the exploration of T_{r_i} is in progress. Thus, until the exploration of that subtree is completed, the managing agent for T_r is responsible for redirecting all agents arriving at r to this subtree, T_{r_i}. Once the exploration of T_{r_i} is completed, the managing agent for T_{r_i} will report this fact to the managing agent for T_r and the latter one may initiate the process of exploration of the next subtree $T_{r_{i+1}}$. Once all subtrees $T_{r_1}, T_{r_2}, \ldots$ are explored the managing agent for T_r returns 'one level up' to report this event to appropriate managing agent.

Observe that the above scheme should be performed in such a way that each subtree T_r 'receives' just enough agents needed for its exploration and not more. This includes one managing agent for the subtree itself, the agents performing the extended DFS traversal of T_r and the agents needed for recursive calls, if any. This is regulated by introducing the agents slowly at the global root so that, within predefined time intervals new agents appear at the global root and are directed gradually by managing agents precisely to the subtree for which the current extended DFS traversal is performed. The time intervals are set up in such a way that if an exploration of a particular subtree is completed then this information has enough time to be carried by the managing agent to the one residing one level up. In this way the flow of agents to a particular subtree is stopped and redirected to the next one supplying the exact amount of agents needed for each of the subtrees. Intuitively, the measurement of time is used indirectly as a communication tool: if a managing agent does not receive for a given amount of time a signal that a recursive call to a subtree is completed, then this means that the exploration of that subtree is not completed and more agents are needed to finish it — hence another agent will be sent to that subtree.

Now we give a detailed description of the modifications to the exploration strategy described in Section 2.2.1 so that it is valid for agents communicating locally. At the beginning of exploration (i.e., when $\texttt{GCTE}_\varepsilon$ is called for a tree T), one distinguished agent is selected to be constantly present at the root r_0 of the entire tree T. This agent is called the *managing agent for T*. Similarly, whenever a recursive call of $\texttt{GCTE}_\varepsilon$ is made for any input node r, the first agent that arrives at r is the managing agent for T_r and it stays at r until the entire subtree T_r is explored.

Extension of Step 1 of $\texttt{GCTE}_\varepsilon$. Once all 6 members of the i-th team are present at the root r of a subtree for which the extended DFS traversal is performed, the i-th team operates exactly as described in Section 2.2.1. Recall that the i-th team finishes its work with one of its agents being at the root. The beginning of the operation of the $(i+1)$-th team is postponed until exactly 6 new agents, each with energy b, appear at r. Then, the $(i+1)$-th team resumes the extended DFS traversal. We note that the agents forming each team will arrive at r directly from the global root of the tree and this will become clear after description of the extension of Step 3 of $\texttt{GCTE}_\varepsilon$.

Extension of Step 3 of $\texttt{GCTE}_\varepsilon$. For this part we need to describe how a recursive call is performed by an instance of $\texttt{GCTE}_\varepsilon$. This includes two actions: initiating the call and receiving information that a recursive call is completed, i.e., that the exploration of the subtree for which the call was conducted is finished. Suppose that an instance of $\texttt{GCTE}_\varepsilon$ with input r and b performs a call for a subtree rooted at a node r_i (see Figure 2.1). Recall that the managing agent for T_r, denoted by $a(r)$ is present at r during exploration of T_r. First, $a(r)$ waits until a new agent, denoted by $a(r_i)$, appears at r and after it arrives, agent $a(r_i)$ is sent to r_i and it becomes the managing agent for T_{r_i}. Then, the algorithm sends each agent arriving at r to the node r_i until the agent $a(r_i)$ returns to r. This completes the recursive call for r_i and $a(r_i)$ stays idle at r indefinitely (and will not play any role in the remaining part of the exploration). Then, the next recursive call, if any, that needs to be done is performed. The information about the current status of each recursive call made by the instance of $\texttt{GCTE}_\varepsilon(r, b)$, is maintained by $a(r)$, the managing agent for

T_r, and once all recursive calls are completed this managing agent returns to the node that is the ancestor of r from which the instance of $\texttt{GCTE}_\varepsilon(r, b)$ was called.

Distribution of agents at the global root. Note that the above description defines the operation of agents for each instance of $\texttt{GCTE}_\varepsilon$ except for the managing agent at the global root r_0 for the first call to $\texttt{GCTE}_\varepsilon$. The managing agent at the global root has all agents at its disposal from the first step and does not need to wait for the arrival of an agent. Therefore we introduce an artificial delay denoted by $d(\varepsilon)$ as defined below. The $d(\varepsilon)$ is an integer and it will be understood that the agents will appear at the global root r in time intervals of length $d(\varepsilon)$. This time interval is defined as

$$d(\varepsilon) = 8B \tag{2.2}$$

We will prove later that this delay is sufficient for our strategy.

The exploration strategy modified as above is called $\texttt{LCTE}_\varepsilon$ (*Local Communication Tree Exploration*). We now prove that $\texttt{LCTE}_\varepsilon$ works correctly in the local communication model.

Lemma 10. *For $0 < \varepsilon < 1/4$, Algorithm $\texttt{LCTE}_\varepsilon$ correctly explores any tree T using local communication between agents.*

Proof. The correctness of the extension of Step 1 follows from Lemma 9. To ensure that the extension of Step 3 is correct we need to argue that if an instance of $\texttt{GCTE}_\varepsilon$ called for a node r performs a recursive call for a descendant v, then the information that the subtree T_v is explored (and hence that the corresponding call for v is completed) eventually arrives at r. Let b be the available energy that is part of the input for the call to $\texttt{LCTE}_\varepsilon$ at r. By definition, $B - b$ is the distance of r from the global root and

$$b = b_i = \left\lfloor (1-\varepsilon)^{i-1} B \right\rfloor$$

for some $i \geq 1$. We calculate the total distance that the managing agent for T_v, denoted by $a(v)$, needs to traverse. The agent $a(v)$ traverses the path from the global root to v at distance $B - \lfloor (1-\varepsilon)^i B \rfloor$ and the path from v to the root r. The latter path is of length

$$\left(B - \left\lfloor (1-\varepsilon)^i B \right\rfloor\right) - \left(B - \left\lfloor (1-\varepsilon)^{i-1} B \right\rfloor\right) = \left\lfloor (1-\varepsilon)^{i-1} B \right\rfloor - \left\lfloor (1-\varepsilon)^i B \right\rfloor$$

Thus, the total distance traversed by $a(v)$ is

$$B - \left\lfloor (1-\varepsilon)^{i-1} B \right\rfloor + \left\lfloor (1-\varepsilon)^{i-1} B \right\rfloor - \left\lfloor (1-\varepsilon)^i B \right\rfloor \leq B$$

□

Theorem 11. *Algorithm $\texttt{LCTE}_\varepsilon$ explores T using at most $O(\log B) \cdot OPT$ agents.*

Proof. By Lemma 10, the tree is correctly explored by $\texttt{LCTE}_\varepsilon$ and hence it is enough to count the number of agents used in this exploration strategy. Denote by $m(\varepsilon)$

the overall number of managing agents used by the procedure and denote by $e(\varepsilon)$ the overall number of agents used to perform all extended DFS traversals. Also, denote by $f(\varepsilon)$ the number of remaining agents used by the algorithm.

As shown in Lemma 9, the algorithm uses 6 times the number of agents used by algorithm $\texttt{GCTE}_\varepsilon$ (see Theorem 7) for all extended DFS traversals. Therefore we have

$$e(\varepsilon) \leq \frac{24}{\left(\frac{1}{2} - \varepsilon\right)} \cdot \log_{\left(\frac{1}{1-\varepsilon}\right)} B \cdot OPT \tag{2.3}$$

We now bound $m(\varepsilon)$. Note that there exists exactly one managing agent for each node for which $\texttt{LCTE}_\varepsilon$ is called. Whenever two recursive calls to $\texttt{LCTE}_\varepsilon$ are performed for two nodes r_i and r_j at the same level, the subtrees T_{r_i} and T_{r_j} are independent due to Lemma 3. Thus, any exploration strategy (offline or not) would use at least two distinct agents for exploring T_{r_i} and T_{r_j}. This proves, that if an instance of $\texttt{LCTE}_\varepsilon$ performs recursive calls for nodes $r_1, \ldots, r_p$, then at least p agents need to be used in any exploration strategy for the subtrees $T_{r_1}, \ldots, T_{r_p}$. Thus, the overall number of managing agents for subtrees rooted at the same level is at most OPT. Therefore, by Lemma 5,

$$m(\varepsilon) \leq \log_{\left(\frac{1}{1-\varepsilon}\right)} B \cdot OPT \tag{2.4}$$

We finally prove that

$$f(\varepsilon) = 0 \tag{2.5}$$

To that end take two agents a and a' that arrive at the global root of the tree separated by time interval $d(\varepsilon)$, i.e, a and a' are two consecutive agents 'deployed' at the global root. We will show that $a \notin f(\varepsilon) \implies a' \notin f(\varepsilon)$. Suppose that a becomes the managing agent for some subtree T_r or $a = a_i^j$, where $j < 5$ or the i-th team is not the last one used for the extended DFS traversal of T_r. In any of these cases, each managing agent visited by a' is present at the same node as when visited by a, and hence a' arrives at r and becomes a member of some team performing the extended DFS traversal of T_r. Now let us consider the remaining case when $a = a_k^5$ and k-th team is the last one used for the extended DFS traversal of T_r. We need to consider two sub-cases depending on whether the subtree T_r is big (requiring more than one level of recursion) or small (only a single level).

In the first sub-case, when T_r is big, a recursive call is performed for some node $r_1 \in T_r$ during the execution of $\texttt{LCTE}_\varepsilon(r, b)$ at the node r that we consider. Then, a' arrives at r and is directed to r_1 and becomes the managing agent for T_{r_1}.

In the second sub-case, no recursive call is performed by the instance of $\texttt{LCTE}_\varepsilon$ called for r. Thus, there exists a sequence of nodes $v_0 = r, v_1, \ldots, v_p$ such that an instance of $\texttt{LCTE}_\varepsilon$ was called for v_i by an instance of $\texttt{LCTE}_\varepsilon$ called for v_{i+1}, $i \in \{0, \ldots, p-1\}$, and the subtree $T_{v_{p-1}}$ is explored, or equivalently, none of the instances of $\texttt{LCTE}_\varepsilon$ called for $v_0, \ldots, v_{p-1}$ is going to perform another recursive call. By the formulation of the algorithm, the managing agent for v_i returns to v_{i+1} once

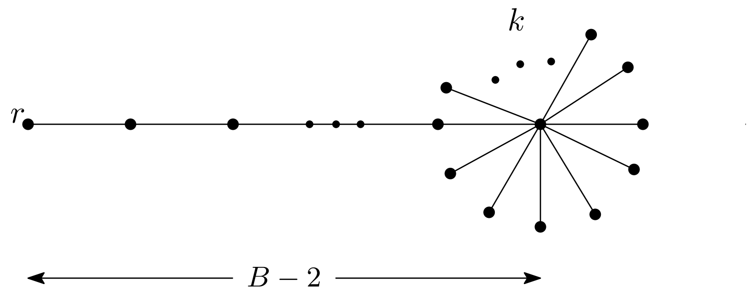

Figure 2.2: An example of a family of graphs for which any online exploration algorithm has a competitive ratio of at least $\Omega(\log B)$.

the managing agent for v_{i-1} returned to v_i for each $i \in \{1, \ldots, p-1\}$. Moreover, the exploration of $T_r = T_{v_0}$ is completed and the managing agent for T_r recognizes this fact and goes to v_1. Thus, the information about completion of the extended DFS traversal of $T_{v_{p-1}}$ arrives at v_p after a number of time units at most the depth of T_{v_p}, which is bounded by B. We bound the number of time units that elapsed since a has been deployed at the global root till the time it reported completion of the extended DFS traversal of T_r. This is at most B, which accounts the time needed to reach r from the global root, plus the time of operation of the k-th team of the extended DFS traversal, to which a belongs, which is at most

$$6b \leq 6B$$

Therefore, by (2.2), the managing agent for T_{v_p} learns about the completion of the exploration of $T_{v_{p-1}}$ prior to the event when a' is deployed at the global root and hence prior to the arrival of a' at v_p. In particular, if v_p is the global root and the exploration of the entire tree is completed, then this fact is known before a' is deployed at the global root. This proves (2.5).

Since the overall number of agents used by $\texttt{LCTE}_\varepsilon$ is $e(\varepsilon) + m(\varepsilon) + f(\varepsilon)$, the theorem follows from (2.3), (2.4) and (2.5).

□

2.3 Competitive Ratio of Online Exploration

We now show a lower bound on the competitive ratio of any online exploration algorithm in the local communication model. The following result implies that the competitive ratio of algorithm $\texttt{LCTE}_\varepsilon$ is asymptotically optimal.

Theorem 12. *Any online exploration algorithm for exploring a tree of depth $D = B - 1$ has a worst case competitive ratio of at least $\Omega(\log B)$.*

Proof. We consider the family of trees which consist of a line of length $D-1$, where one endpoint of the line is the root and the other endpoint is connected to the center of a star with p leaves. Thus all the p leaves of the tree are at distance $D = B - 1$ from the root and there is only one node at distance $D - 1$ (cf. Figure 2.2). An offline algorithm would use exactly p agents for exploring this tree. An online algorithm for exploring this tree can be of two types: We say an algorithm is type-1 if during the algorithm there is no transfer of information from the node at depth $D - 1$ to the root; All other algorithms are of type-2. First notice that if an algorithm of type-1, uses k agents for exploration then k is independent of p, since p remains unknown to the root. Thus, by taking $p > k$, we can make the algorithm fail. So we need to consider only type-2 algorithms where information from the node at depth $D - 1$ is transferred to the root. Any agent visiting this node has at most $B - (D - 1) = 2$ units of energy remaining, so it can return back to depth $D - 3 = B - 4$. Similarly, any agent visiting the node at depth $B - 4$ can return back to depth $B - 8$, and so on. Thus, at least $\Omega(\log B)$ agents are needed to carry the information from the node at depth $D - 1$ back to the root. So any type-2 algorithm would use at least $\Omega(\log B)$ agents. By taking $p = 1$, we get a competitive ratio of $\Omega(\log B)$ for any such algorithm.

$\square$

3 Partial Exploration of Trees

Introduction

In this chapter we consider the problem of tree exploration with a limited number of mobile agents that start at the root of the tree. Each agent has limited energy equal to B and cannot, as a result, traverse more than B edges. The agents do not have to come back to the root. We consider the online version of the problem where the tree is unknown to the agents. The goal is to maximize the number of nodes collectively visited by all agents during the execution. We present an intuitive algorithm based on depth-first search and we study its performance compared to the optimal solution that we could obtain if we knew in advance the map of the tree. We prove that this algorithm has a constant competitive ratio. We also provide a lower bound on the competitive ratio of any algorithm.

The results included in this chapter were presented in [Bam+17a].

The Model

The agents operate in an undirected tree T. The edges at every vertex v in T have locally distinct edge labels $0, \dots, deg(v) - 1$, where $deg(v)$ is the degree of v. These edge labels are referred to as the *local port numbers* at v. Initially, a group of k agents (numbered 1 to k) are placed at a node r of T. Each agent has limited energy B and it consumes one unit of energy for every edge that it traverses.

The tree is initially unknown to the agents, but they learn the map of the tree as they traverse new edges. Each time an agent arrives at a new node, it learns the local port number of the edge through which it arrived, as well as the degree of the node. We assume that agents can communicate at arbitrary distances, so the updated map of the tree, including all agent positions, is instantaneously available to all agents (global communication). Moreover, we will assume, without loss of generality, that the local port number of the edge leading back to the root r is $deg(v) - 1$ for any vertex $v \neq r$ in T.

The goal is to design an algorithm A that maximizes the number of edges collectively discovered by the agents. If $I = \langle T, r, k, B \rangle$ is an instance of the problem, let $A(I)$ denote the number of edges explored using algorithm A on I. We measure the performance of an algorithm A by the competitive ratio $\rho_A = \sup_I \frac{OPT(I)}{A(I)}$, where $OPT(I)$ is the maximum number of edges that can be explored on instance I with full initial knowledge of the instance.

3.1 Exploration Algorithm

Given an instance $I = \langle T, r, k, B \rangle$ of the online tree exploration problem, for $d \geq 0$ we let T_d denote the induced subtree of T containing all vertices at distance at most d from r. We propose an algorithm that works in phases. In each phase i, the agents attempt to completely explore T_{d_i}, for some $d_i \geq 0$. The increasing sequence $(d_i)_{i \geq 1}$ depends only on B and is chosen appropriately, so as to optimize the obtained competitive ratio.

In each phase i, the agents essentially perform a collaborative depth-first search within T_{d_i} (they ignore unexplored edges that lead to nodes at distance greater than d_i from r). Agents are sent off sequentially, i.e., the next agent waits until the current one has depleted its energy. Each agent always chooses the smallest local port number that leads to an unexplored edge within T_{d_i}. Formally, the algorithm is as follows: (the computation of $(d_i)_{i \geq 1}$ is explained in the proof of Theorem 13)

Algorithm 2 Tree exploration algorithm for energy constrained agents

```
Input: Tree $T$, root vertex $r$, number of agents $k$, energy budget $B$
 1: for $i \leftarrow 1, 2, \dots$ do
 2:     while there is an agent $a$ at $r$ and $T_{d_i}$ is not fully explored do
 3:         $a \leftarrow$ the agent with the least positive remaining energy at $r$
 4:         while agent $a$ has energy and $T_{d_i}$ is not fully explored do
 5:             agent $a$ follows the smallest local port number that leads to an
    unexplored edge in $T_{d_i}$
 6:         end while
 7:         while agent $a$ has energy and it is not at $r$ do
 8:             agent $a$ moves towards $r$
 9:         end while
10:     end while
11: end for
```

Note that, if the current agent stops and leaves some unexplored edges in the subtree rooted at its last position, then the next agent will move to the last position of the current agent and will continue the depth-first search. On the other hand, if the current agent has completely explored the subtree rooted at its last position, the next agent will take a shortcut to the next node along the depth-first search path of T_{d_i} that has unexplored edges.

Theorem 13. *The sequence $(d_i)_{i \geq 1}$ can be chosen as a function of B so that the proposed algorithm achieves a competitive ratio of $1 + 4\varphi < 7.473$, where $\varphi \approx 1.618$ is the golden ratio.*

Proof. Consider the sequence $(f_i)_{i \geq 0}$ given by $f_0 = 0$, $f_1 = 1$, $f_2 = 2$, and $f_i = f_{i-1} + f_{i-2}$ for $i \geq 3$. For $i \geq 0$, let $S_i = \sum_{j=0}^{i} f_j$ and let $\gamma \geq 1$ be the smallest index such that $S_{\gamma+1} > B$. Note that $S_i = f_{i+2} - 2$ for all $i \geq 0$ and, furthermore, $\gamma \geq 2$ if and only if $B \geq 3$. Now, for $i = 1, \ldots, \gamma$, we define $d_i = B - S_{\gamma-i}$. The sequence $(d_1, \ldots, d_\gamma)$ is strictly increasing and, setting for convenience $d_0 = 0$, it satisfies $d_{i-1} - d_{i-2} > B - d_i$ for $i \geq 2$.

We will assume that the instance is such that the algorithm fails to explore the whole tree T. We further assume for the moment that $B \geq 3$, therefore $\gamma \geq 2$, and that the last phase to be executed by the algorithm is phase $\sigma \geq 2$, i.e., $T_{d_{\sigma-1}}$ is completely explored and T_{d_σ} only partially.

We first give an upper bound on the number of edges explored by an optimal offline algorithm OPT. Any agent in the offline algorithm can explore at most $B - d_{\sigma-1}$ new edges that are not contained in $T_{d_{\sigma-1}}$, because it uses up at least $d_{\sigma-1}$ of its energy to reach the bottommost node of $T_{d_{\sigma-1}}$. We can therefore bound the number of edges explored by OPT as

$$|OPT| \leq |T_{d_{\sigma-1}}| + k(B - d_{\sigma-1}) \tag{3.1}$$

Next, we give a lower bound on the number of edges $|SOL|$ explored by our algorithm. For $i = 1, \ldots, \sigma$, let n_i be the number of newly explored edges in phase i of the algorithm and k_i be the number of fresh agents (not carried over from the previous phase) used in this phase. For convenience, let also $n_0 = 0$. Note that we have $\sum_{i=1}^{\sigma-1} n_i = |T_{d_{\sigma-1}}|$.

Consider the first phase of the algorithm. Let DFS_1 be the closed walk visiting all edges in T_{d_1} in the same order as a depth-first search, which always follows the local port with the smallest number first. The length of DFS_1 satisfies $|\mathrm{DFS}_1| \leq 2n_1$. The first agent used by the algorithm will spend all of its energy making a progress of B steps on DFS_1. The second agent either moves to where the first agent stopped or shortcuts to a point in the path from r to where the first agent stopped, and then makes progress on the closed walk DFS_1. Note that if the previous agent stopped at distance d_1, then the next agent will always shortcut. In any case, the second agent uses at most $d_1 - 1$ of its energy before it starts making progress on DFS_1, so the progress is at least $B - d_1 + 1$. Similarly, all other agents in the first phase, except for the last agent, make a progress of at least $B - d_1 + 1$ on DFS_1, yielding a total progress of at least $B + (k_1 - 2)(B - d_1 + 1)$ for the first $k_1 - 1$ agents. If p is the progress contributed by the k_1-th agent, we have $B + (k_1 - 2)(B - d_1 + 1) + p \leq 2n_1$.

The k_1-th agent used at most $d_1 - 1$ of its energy before it made progress p, therefore at the end of phase 1 either its energy is exhausted (which implies $p \geq B - d_1 + 1$, and the previous progress inequality gives $k_1(B - d_1 + 1) \leq 2n_1$), or it is located at the root with at least $B - d_1 + 1 - p$ available energy. If its remaining energy is positive, it will be the first agent to be activated in the second phase.

Now assume that the algorithm has completed phase $i-1$ for $i \geq 2$ and therefore completely explored $T_{d_{i-1}}$. Let $v_1^{(i)}, v_2^{(i)}, \ldots, v_{t_i}^{(i)}$ be all vertices of depth d_{i-2} in T,

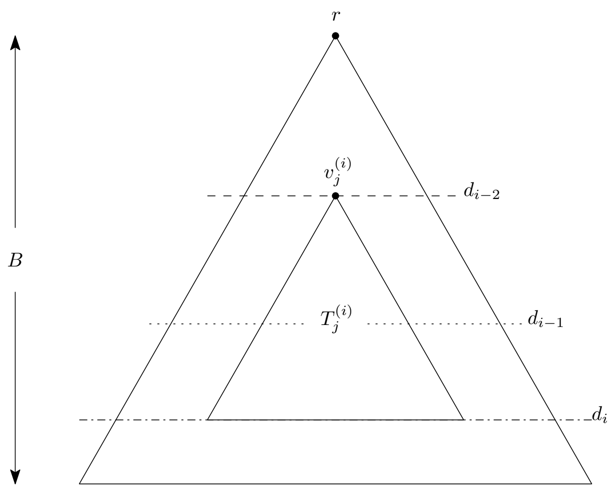

Figure 3.1: Subtrees $T_j^{(i)}$ in tree T.

whose subtrees contain unexplored edges. Moreover, let $T_j^{(i)}$ be the induced subtree of T_{d_i} with root $v_j^{(i)}$ (Fig. 3.1) and $n_j^{(i)}$ be the number of edges of $T_j^{(i)}$. The subtrees $T_j^{(i)}$ are completely explored up to the vertices at level d_{i-1}, but unexplored below. As all vertices of the subtrees $T_j^{(i)}$ lie at a distance between d_{i-2} and d_i from the root, we have

$$\sum_{j=1}^{t_i} n_j^{(i)} \leq n_{i-1} + n_i \tag{3.2}$$

Let $k_j^{(i)}$ be the number of agents that start in the i-th phase and reach the vertex $v_j^{(i)}$ with energy at least $B - d_{i-2}$. As the agents in every phase only move to subtrees with unexplored edges, every agent used in phase i, will move to one of the subtrees $T_j^{(i)}$ and therefore arrive at $v_j^{(i)}$ with energy $B - d_{i-2}$. We therefore have $\sum_{j=1}^{t_i} k_j^{(i)} = k_i$.

We now want to bound the number of agents $k_j^{(i)}$ that we need to explore a subtree $T_j^{(i)}$ in terms of $n_j^{(i)}$ as above. Let $\mathrm{DFS}_j^{(i)}$ be a depth-first search tour of all vertices in $T_j^{(i)}$. Then $|\mathrm{DFS}_j^{(i)}| \leq 2n_j^{(i)}$ and $|\mathrm{DFS}_j^{(i)}| \geq 2(d_{i-1} - d_{i-2})$ because $T_j^{(i)}$ contains an unexplored leaf at distance at least $d_{i-1} - d_{i-2}$ from $v_j^{(i)}$. Every agent that enters $T_j^{(i)}$ with energy at least $B - d_{i-2}$, will either move to the previous agent's stopping position or shortcut, thus making at least $B - d_i + 1$ progress on $\mathrm{DFS}_j^{(i)}$. This also holds for the last agent because at the time the last agent enters $T_j^{(i)}$ there is still an unexplored leaf. Thus, the last agent, also, can make a progress of at least $B - d_i + 1$ on $\mathrm{DFS}_j^{(i)}$, as the part of $\mathrm{DFS}_j^{(i)}$ returning from the last unexplored leaf to $v_j^{(i)}$ is at least $d_{i-1} - d_{i-2} > B - d_i$. We get

$$k_j^{(i)} \cdot (B - d_i + 1) \leq 2n_j^{(i)}$$

and hence by using Inequality 3.2 and assuming $i \geq 3$:

$$k_i \cdot (B - d_i + 1) = \sum_{j=1}^{t_i} k_j^{(i)} \cdot (B - d_i + 1) \leq \sum_{j=1}^{t_i} 2n_j^{(i)} \leq 2n_{i-1} + 2n_i$$

By considering the subtree T'_{d_σ} of T_{d_σ} containing all vertices explored by our algorithm, we obtain the above inequality also for the last phase σ of the algorithm. In particular for $i = 2$, we also take into account the contribution of the last agent of phase 1 (assuming it reached r), which starts at $v_1^{(2)} = r$ with at least $B - d_1 + 1 - p$ available energy and, since it is the first agent to contribute to $DFS_1^{(2)}$, it contributes at least $B - d_1 + 1 - p$ progress. The total progress during the second phase is, therefore: $B - d_1 + 1 - p + k_2 \cdot (B - d_2 + 1) \leq 2n_1 + 2n_2$. If, however, the last agent of the first phase didn't reach r, then for phase 2 we have simply $k_2 \cdot (B - d_2 + 1) \leq 2n_1 + 2n_2$.

Summing the progress inequalities from each phase and using the monotonicity of the d_i and the fact that $\sum_{i=1}^{\sigma} k_i = k$, we obtain

$$\begin{aligned} k(B - d_\sigma + 1) &\leq \sum_{i=1}^{\sigma} k_i(B - d_\sigma + 1) \leq \sum_{i=1}^{\sigma} k_i(B - d_i + 1) \\ &\leq \sum_{i=1}^{\sigma} 2n_{i-1} + 2n_i \leq 4|SOL| \end{aligned}$$

Combining the upper bound on OPT (inequality 3.1), the lower bound on SOL above, and the inequality $|T_{d_{\sigma-1}}| \leq |SOL|$, we obtain

$$\begin{aligned} \frac{|OPT|}{|SOL|} &\leq \frac{|T_{d_{\sigma-1}}| + k(B - d_{\sigma-1})}{|SOL|} \\ &\leq 1 + 4\frac{B - d_{\sigma-1}}{B - d_\sigma + 1} = 1 + 4\frac{S_{\gamma-\sigma+1}}{1 + S_{\gamma-\sigma}} = 1 + 4\frac{-2 + f_{\gamma-\sigma+3}}{-1 + f_{\gamma-\sigma+2}} \end{aligned}$$

It can be verified that $\frac{-2+f_{i+3}}{-1+f_{i+2}}$ is strictly increasing in i for $i \geq 0$ and it converges to φ, therefore $\rho_A \leq 1 + 4\varphi$.

Finally, if $B \leq 2$, which implies $\gamma = \sigma = 1$, or if $\sigma = 1 < \gamma$, then the algorithm finishes in the first phase and it follows from the above arguments that $|SOL| \geq \frac{k(B-d_1+1)}{2}$, while $|OPT| \leq kB$. Therefore, $\rho_A \leq \frac{2B}{B-d_1+1} = \frac{2B}{1+S_{\gamma-1}}$. However, recall that $S_{\gamma+1} > B$, so we have

$$\rho_A \leq 2\frac{-1 + S_{\gamma+1}}{1 + S_{\gamma-1}} = 2\frac{-3 + f_{\gamma+3}}{-1 + f_{\gamma+1}}$$

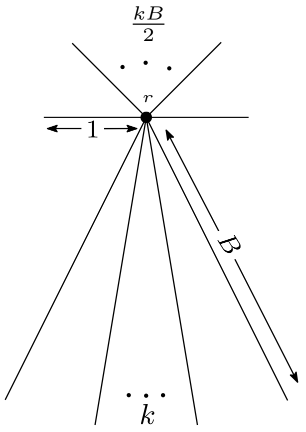

Figure 3.2: An example of a family of graphs for which no algorithm can achieve a competitive ratio of $2 - o(1)$.

It can be verified that $\frac{-3+f_{i+3}}{-1+f_{i+1}}$ is strictly increasing in i for $i \geq 2$ and it converges to $\varphi + 1$, therefore $\rho_A \leq 2 + 2\varphi < 1 + 4\varphi$. □

3.2 Lower Bound on the Competitive Ratio

Proposition 14. *No algorithm achieves a competitive ratio of* $2 - o(1)$.

Proof. Given positive integers k and B, where B is even, consider the star consisting of k rays of length B and $\frac{kB}{2}$ rays of length 1 (cf. Figure 3.2). A group of k agents, each with energy B, start on r. For every algorithm, the adversary can ensure that no agent ever enters a long ray: whenever an agent is at the center and decides to follow an unexplored edge, the adversary directs it to a short ray. For every edge that an agent explores, it needs to go back to the center in order to explore other edges. Therefore, every agent can explore at most $\frac{B}{2}$ edges for a total of at most $\frac{kB}{2}$ edges. On the other hand, the optimal solution is to send all agents in the long rays and explore kB edges. □

4 Sustainable Exploration of Trees

Introduction

We consider the problem of sustainable exploration on any weighted tree T. We believe this is the first study on these types of exploration problems where the profits and costs are interchangeable, i.e., the profits can be used for paying the costs. The main difficulty in the problem comes from the constraint that the agent may not run out of energy at any point during the traversal, which puts constraints on the order in which various parts of the tree are traversed.

We first show that the general version of the problem called CollectMax is NP-hard on weighted trees. We then consider some easier versions of the problem. We define CollectMaxBigBudget as the same problem with the assumption that the agent initially starts with a sufficiently large budget B of energy (which equals twice the sum of the weights of T); in such case, the agent can perform a traversal of the complete tree irrespective of the values of gains at the nodes. We provide an algorithm for finding the optimal subtree of T that maximizes the final gain. Finally, we show how the agent can obtain the maximum gain while using the smallest starting budget B.

The Model

We consider an agent starting at the root r of a tree $T = (V, E)$ endowed with an edge weight function $w : E \to \mathbb{N}$ and a node gain function $g : V \to \mathbb{N}$. The agent has full knowledge of the graph. Whenever the agent traverses an edge e, it spends energy equal to the weight of the edge $w(e)$. Additionally, whenever the agent visits a node v for the first time, it collects energy equal to the gain of the node $g(v)$. The agent stores the collected energy and it can also use it for traversing edges. We are interested in computing a closed walk $p = v_0, e_1, v_1, \ldots, e_\ell, v_\ell$, where $v_0 = v_l = r$, with the property that the available energy of the agent cannot become negative at any moment of the execution.

We will be using the following notation. Let $p_j = v_0, e_1, v_1, \ldots e_j, v_j,$ where $0 \leq j \leq \ell$, denote the walk that is included in walk p between its starting node v_0 and node v_j, and $V_p = \{v \in V : v = v_i, i \leq \ell\}$ denote the set of nodes that are included in walk p.

Given a walk p on instance $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$, let

$$\mathsf{n\text{-}surplus}_i(p) = \sum_{v \in V_{p_i}} g(v) - \sum_{j=1}^{i} w(e_j),\ 0 \leq i \leq \ell$$

be the function which calculates the amount of energy collected by the mobile agent by traversing the walk p up until node v_i. Moreover, let

$$\begin{aligned}\mathsf{e\text{-}surplus}_i(p) &= \mathsf{n\text{-}surplus}_{i-1}(p) - w(e_i),\ 1 \leq i \leq \ell\\ \mathsf{e\text{-}surplus}_0(p) &= g(v_0)\end{aligned}$$

be the function which calculates the amount of available energy when the agent has traversed the walk p up until the edge e_i, i.e., before collecting any energy available in node v_i. Note that if p is a closed walk, it holds that $\mathsf{n\text{-}surplus}_\ell(p) = \mathsf{e\text{-}surplus}_\ell(p)$.

We define *sustainable* walk, any walk $p = v_0, e_1, v_1, \ldots, e_\ell, v_\ell$, where $v_0 = v_l = r$, such that $\mathsf{e\text{-}surplus}_i(p) \geq 0$, $\forall i \in \{1, \ldots, \ell\}$. Furthermore, we consider that a walk p is of *DFS-type*, if every distinct edge in p is being traversed exactly twice. Given an instance $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$, let $I|_{T_v} = \langle T_v, v, g|_{T_v}, w|_{T_v} \rangle$, denote the instance I restricted to the subtree T_v rooted at v, where $v \in V$. Finally, given an edge-weighted tree $T(V, E)$, we denote by $W = \sum_{\forall e \in E} w(e)$ the sum of weights of the edges of the tree, by $deg(v)$ the number of neighbours of node v, by $parent(v)$ the parent node of v and by $\mathsf{ch}(v)$ the set of child nodes of v.

4.1 Hardness of Sustainable Exploration

We can now proceed to the formal definition of CollectMax.

Definition 15 (CollectMax)**.**
Instance: $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$, where $T(V, E)$ is an undirected, weighted tree rooted at r, $g(v) \in \mathbb{N}^+$ denotes the profit of energy of node v and $w(e) \in \mathbb{N}^+$ denotes the weight of the edge e.
Feasible solution: Any sustainable walk $p = v_0, e_1, v_1 \ldots, e_\ell, v_\ell$, where $v_0 = v_\ell = r$.
Goal: Maximize $\mathsf{n\text{-}surplus}_\ell(p)$.

Theorem 16. CollectMax *is NP-hard.*

Proof. We consider the CollectMax decision problem:

CollectMax decision problem
Instance: $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E}, G \rangle$, where $T(V, E)$ is an undirected, weighted tree rooted at r, $g(v) \in \mathbb{N}^+$ denotes the profit of energy of node v, $w(e) \in \mathbb{N}^+$ denotes the weight of the edge e and G is an integer.
Question: Is there a sustainable walk $p = (v_0, e_1, v_1 \ldots, e_\ell, v_\ell)$, with $v_0 = v_\ell = r$, such that $\mathsf{n\text{-}surplus}_\ell(p) \geq G$?

We will reduce the following problem called *Partition* to CollectMax. Partition is one of Karp's 21 NP-complete problems, [Kar72].

Partition problem
Instance: A set $W = \{s_1, s_2, \ldots, s_m\}$ of positive integers, where $S = \sum_{i=1}^m s_i$.
Question: Can $\{1, 2, \ldots, m\}$ be partitioned into 2 disjoint subsets $A = \{\alpha_1, \alpha_2, \ldots, \alpha_{m_1}\}$ and $B = \{\beta_1, \beta_2, \ldots, \beta_{m_2}\}$, such that $\sum_{\alpha_i \in A} s_{\alpha_i} = \sum_{\beta_i \in B} s_{\beta_i} = S/2$?

We construct an instance $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E}, G \rangle$ of the COLLECT-MAX decision problem from an instance of Partition as follows. The tree T has a root r, a set $U = \{u_i \mid 1 \leq i \leq m\}$ of internal nodes adjacent to r, a set $V = \{v_i \mid 1 \leq i \leq m\}$ of leaves, where $\forall i \in \{1, \ldots, m\}$, v_i is adjacent to u_i and finally a leaf z adjacent to the root.
The gain function g is defined as follows:

- $g(r) = S/2$
- $\forall 1 \leq i \leq m$, $g(u_i) = 3s_i$
- $\forall 1 \leq i \leq m$, $g(v_i) = 6S + 4s_i$
- $g(z) = 4S$

The weight function w is defined as follows:

- $w(r, z) = S$
- $\forall 1 \leq i \leq m$, $w(r, u_i) = s_i$
- $\forall 1 \leq i \leq m$, $w(u_i, v_i) = 3S + s_i$

Optimal gain $G = 9S/2$.

First, assume that there exists a solution A, B for the instance of the Partition problem. In such a case, notice that $\forall 1 \leq i \leq m$, $s_i \leq S/2$. We show that there exists a sustainable walk p of a certain length ℓ such that $\mathsf{n\text{-}surplus}_\ell(p) = G$. In order to define the walk p, we define the following walks which we will then concatenate. Let p_1 be the closed walk starting from r in which the agent visits the nodes $u_{\alpha_1}, \ldots, u_{\alpha_{m_1}}, z$ in this order and let p_2 be the closed walk starting from r in which the agent visits the leaves $v_{\beta_1}, \ldots, v_{\beta_{m_2}}$ in this order. In both walks the agent goes from r to the first node, from the first node to the next and from the last node to r via the shortest path in the tree. Finally, the walk p is the concatenation $p_1; p_2$.

Claim 1. *p_1 is sustainable and at the end the agent has $3S$ energy units.*

Proof. First, we prove by a simple induction that the agent returns to r, after visiting u_{α_i}, having collected $S/2 + s_{\alpha_1} + \ldots + s_{\alpha_i}$ units of energy, for $1 \leq i \leq m_1$. For $i = 1$, the agent starts p_1 with $S/2$ units of energy, which is sufficient to traverse any edge incident to r, except for (r, z). After traversing (r, u_{α_1}) and collecting the energy from u_{α_1}, it has $S/2 + 2s_{\alpha_1}$ units of energy. This is sufficient to return to r, having $S/2 + s_{\alpha_1}$ energy units.

Assume now that the agent is at r, having collected $S/2 + s_{\alpha_1} + \ldots + s_{\alpha_i}$ units of energy from the previously visited nodes and is supposed to visit next $u_{\alpha_{i+1}}$, where $i < m$. The agent's energy is sufficient to visit $u_{\alpha_{i+1}}$, collect $3s_{\alpha_{i+1}}$ energy units and then return to r having $S/2 + s_{\alpha_1} + \ldots + s_{\alpha_{i+1}}$ units of energy.

Therefore, after visiting node $u_{\alpha_{m_1}}$ and returning to r, the agent has collected $S/2 + s_{\alpha_1} + \ldots + s_{\alpha_{m_1}} = S/2 + S/2 = S$ energy units. This amount of energy is sufficient to the agent to traverse (r, z), collect $4S$ and return to r with $3S$ energy units.

□

Claim 2. *p_2 is sustainable and at the end the agent has $9S/2$ energy units.*

Proof. We prove by induction that the agent returns to r after visiting v_{β_i}, having collected $3S + 3s_{\beta_1} + \ldots + 3s_{\beta_i}$ units of energy, for $1 \leq i \leq m_2$. For $i = 1$, by Claim 1 the agent starts p_2 with $3S$ units of energy, which is sufficient to reach node u_{β_1}. After traversing (r, u_{β_1}) and collecting the energy from node u_{β_1}, the agent has $3S + 2s_{\beta_1}$ units of energy. This is sufficient for the agent to reach the leaf node v_{β_1}. After traversing $(u_{\beta_1}, v_{\beta_1})$ and collecting the energy from v_{β_1}, the agent has $6S + 5s_{\beta_1}$ units of energy. Therefore, the agent returns to r, having $3S + 3s_{\beta_1}$ energy units left.

Assume now that the agent is at r, having collected $3S + 3s_{\beta_1} + \ldots + 3s_{\beta_i}$ units of energy from the previously visited nodes and is supposed to visit next $v_{\beta_{i+1}}$, where $i < m$. The agent's energy is sufficient to visit $u_{\beta_{i+1}}$. After traversing $(r, u_{\beta_{i+1}})$ and collecting the node's energy, the agent has at its disposal $3S + 3s_{\beta_1} + \ldots + 3s_{\beta_i} + 2s_{\beta_{i+1}}$ units of energy. This is sufficient for the agent to reach leaf $v_{\beta_{i+1}}$. After traversing $(u_{\beta_{i+1}}, v_{\beta_{i+1}})$ and collecting the energy from $v_{\beta_{i+1}}$ the agent has $6S + 3s_{\beta_1} + \ldots + 3s_{\beta_i} + 5s_{\beta_{i+1}}$ energy units. This energy is sufficient for the agent to return to r, having $3S + 3s_{\beta_1} + \ldots + 3s_{\beta_{i+1}}$ energy units.

Therefore, after visiting leaf $v_{\beta_{m_2}}$ and returning to r, the agent has collected $3S + 3s_{\beta_1} + \ldots + 3s_{\beta_{m_2}} = 3S + 3S/2 = 9S/2$ energy units. □

By Claim 2, the walk p is sustainable and at the end the agent has collected $9S/2 = G$ units of energy.

Now assume that there is a sustainable walk p which the agent concludes with at least G energy units. We will show that there exists a partition for the original Partition instance.

Let us start by noticing that the agent cannot visit any leaf v_i during its walk before visiting leaf node z, as the agent can collect at most $S/2 + s_1 + \ldots + s_m = S/2 + S = 3S/2$ energy units from visiting the internal nodes u_i. On the other hand, the agent must visit at least a subset of the leaf nodes, otherwise it can collect at most $3S/2$ energy units from the internal nodes and $2S$ energy units from z, thus a total of $7S/2$ energy units. Therefore, the agent visits z at some point of its walk p, before visiting any leaf node v_i.

The agent in order to visit z, it must collect at least $S/2$ additional energy units by visiting some subset of the internal nodes u_i, we will denote this subset by $A' \subseteq W$. Suppose now, that the agent has collected $S + x$ units of energy at r right before visiting z, where $x \geq 0$. This implies that the sum of integers in A' is equal to $S/2 + x$ and the sum of integers in $W \setminus A'$ is equal to $S/2 - x$. When the agent returns to r, after visiting z, it has $3S + x$ energy units.

Let us now consider the remaining part of the agent's walk, i.e, after visiting z. If the agent visits a leaf node v_i that its neighbouring node u_i has been previously visited then we distinguish the following two cases, $x < 2s_i$ and $x \geq 2s_i$. If $x < 2s_i$ then the walk is no longer sustainable, as the agent does not have sufficient energy to reach v_i. On the other hand, if $x \geq 2s_i$ then the agent returns to r after visiting v_i having $3S+x$ units of energy, therefore, the agent gains nothing additional. Thus, we can consider that the agent, in the remaining part of p, only visits previously unvisited branches whose internal nodes u_i belong to $W \setminus A'$.

If the agent, now, visits an unvisited branch up to the internal node u_i the agent returns to r, having collected s_i additional energy units. If, on the other hand, the agent visits the branch up to the leaf v_i, then it returns to r, having collected $3s_i$ additional energy units. Therefore, if the agent visits all the unvisited branches up to the leaf nodes, the agent will have collected a total of $3S+x+3(S/2-x) = 9S/2-2x$ units of energy at r. Thus, the optimal gain is achieved only when $x = 0$, which implies the initial subset A' corresponds to a solution to the original Partition problem.

□

4.2 Sustainable Exploration with Energy Source

In this section, we modify the CollectMax problem by considering that we can provide an amount of initial energy B to the agent. Under this assumption, the definition of function e-surplus changes into:

$$\begin{aligned} \textsf{e-surplus}_i(p) &= \textsf{n-surplus}_{i-1}(p) - w(e_i),\ 1 \leq i \leq \ell \\ \textsf{e-surplus}_0(p) &= g(v_0) + B \end{aligned}$$

Note that, given a tree, if $B \geq 2W$, the agent can traverse the complete tree. We are now interested in studying two different optimization goals:

- Maximize the agent's remaining energy at the end of the walk, assuming B is sufficiently large.
- Find the minimum initial energy B, that is sufficient to collect the optimal amount of energy at the root.

We will analyze these two goals in the following subsections.

4.2.1 Maximize Gain

Definition 17 (CollectMaxBigBudget).
Instance: $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$, where $T(V, E)$ is an undirected, weighted

tree rooted at r, $g(v) \in \mathbb{N}^+$ denotes the profit of energy of node v and $w(e) \in \mathbb{N}^+$ denotes the weight of the edge e.
Feasible solution: Any sustainable walk $p = v_0, e_1, v_1 \ldots, e_\ell, v_\ell$, with $v_0 = v_\ell = r$, for an agent with initial energy $B = 2W$.
Goal: Maximize $g_{\mathsf{max}}(I) = \mathsf{n\text{-}surplus}_\ell(p)$.

We will be using the terms $g_{\mathsf{max}}(I)$ and $OPT_g(I)$ interchangeably, to denote the maximum amount of energy we can collect at the root for instance I.

A closed walk on a tree T may include the total set of nodes of T, hence, the total set of edges as well, or it may be limited to a subset of it. We call the *induced* tree $T_p(V', E')$ of p, the tree which consists of the set of nodes and the set of edges that belong to the closed walk p.

Theorem 18. *Any optimal solution for* CollectMaxBigBudget *is of DFS-type, where the order of the traversals is not important.*

Proof. We will prove it by contradiction. Assume that there exists an instance I for CollectMaxBigBudget, for which an optimal solution is a non DFS-type closed walk p_{OPT}. Without loss of generality, consider that e_i is an edge in p_{OPT} that is traversed more than 2 times, i.e. at least 4 times. Suppose now, that e_{i1}, e_{i2}, e_{i3} and e_{i4} are the last appearances of e_i in the sequence of edges of p_{OPT} and let w_1 denote the part of the walk that is included between e_{i1} and e_{i2}, and w_2 denote the part of p_{OPT} that is included between e_{i3} and e_{i4}, therefore, p_{OPT} is of the form $v_0, \ldots, e_{i1}, w_1, e_{i2}, \ldots, e_{i3}, w_2, e_{i4}, \ldots, v_\ell$. We can construct a new closed walk p_{new} which consists of the same sets of nodes and edges of p_{OPT}, V' and E' respectively, but differs in the sequence of vertices and edges as follows. We concatenate w_1 and w_2 between e_{i1} and e_{i2} and we remove the part e_{i3}, w_2, e_{i4}. Hence, p_{new} is of the form $v_0, \ldots, e_{i1}, w_1, w_2, e_{i2}, \ldots, v_\ell$. The total gain of p_{new} is $\mathsf{n\text{-}surplus}_\ell(p_{\mathsf{new}}) = \mathsf{n\text{-}surplus}_\ell(p_{OPT}) - (-2 \cdot w(e_i)) > \mathsf{n\text{-}surplus}_\ell(p_{OPT})$ (the last inequality holds since $\forall e \in E, w(e) > 0$), which contradicts the optimality of p_{OPT}.

Considering the order of the traversals, given the fact that any optimal solution for CollectMaxBigBudget is of DFS-type, we have that each edge included in an optimal walk p_{OPT} is traversed exactly 2 times. In the other direction, by definition, we have that the agent has sufficient energy to traverse each edge of the tree two times, since $B = 2W$. Therefore, these two observations, result in the fact that the order in which the agent will choose to perform the DFS-traversal is of no significance, since the agent will always have energy to continue the DFS-traversal. □

Lemma 19. *For every instance* $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$ *of* CollectMaxBigBudget *it holds*

$$OPT_g(I) = g(r) + \sum_{\substack{v \in \mathsf{ch}(r): \\ OPT_g(I|_{T_v}) > 2w(r,v)}} \Big(OPT_g(I|_{T_v}) - 2w(r,v) \Big) \qquad (4.1)$$

where $I|_{T_v} = \langle T_v, v, g|_{T_v}, w|_{T_v}\rangle$ *and* $\mathsf{ch}(r)$ *denotes the set of children of* r.

Proof. Consider an optimal solution p_{OPT} for an instance I. By Theorem 18, we have that p_{OPT} is a sustainable walk of DFS-type. Therefore, we know that during the walk each edge will be traversed exactly 2 times. Necessarily, the first traversed edge of the walk, will be an edge adjacent to the root, leading to a child of the root, say v. Consequently, this optimal walk will contain a DFS-traversal, not necessarily a complete one, of the subtree hanging from node v, T_v. Upon return to node v, the agent must have collected the optimal gain in T_v, $OPT_g(I \mid_{T_v})$, otherwise it would contradict the optimality of the solution, and subsequently, traverse edge (r, v) in order to return to the root and continue, possibly, the DFS-traversal on another branch of T. For each edge (r, v), where $v \in \mathsf{ch}(r)$, included in p_{OPT}, holds the property $OPT_g(I \mid_{T_v}) > 2w(r, v)$, otherwise the gain in r would be decreased and the optimality of p_{OPT} would be contradicted. As a result, $OPT_g(I)$ is equal to the initial gain of the root, $g(r)$, plus the gain of every subtree hanging from a child node v of the root for which holds the property $OPT_g(I \mid_{T_v}) > 2w(r, v)$, minus the cost of reaching this subtree and returning from it. This is equal to the righthand side of equation 4.1. □

We will call a node v *profitable* if it has the following property, $OPT_g(I \mid_{T_v}) > 2w(parent(v), v)$.

Before the formal definition of the algorithm for CollectMaxBigBudget, let us briefly describe the algorithm. The basic idea of the algorithm is to check, in a bottom-up manner, every node of the tree to see if it is a profitable node, i.e, if we can have gain by visiting this node. The bottom-up manner means that we start by checking the leaves of the given tree T and work our way towards the root. Algorithm 3 is an iterative algorithm and at each iteration a leaf is picked to be checked. Consider that in the i-th iteration, leaf l has been chosen. First, we check if it is a profitable node and afterwards, we remove it from the tree, as a result, we get a new tree $T_i = T_{i-1} \setminus \{l\}$. If leaf l was a profitable node, then we update the gain of its parent node, $parent(l)$, by adding to the gain value of $parent(l)$, the gain value of l and subtracting the cost of reaching l and returning from it. If l is a non profitable node, the gain of the parent node remains unchanged. Observe that, during the execution of the algorithm, every node of the initial tree will become a leaf at some moment of the execution. That is, once all of the children of a node v have been checked, say at the j-th iteration the last child of v was checked, then node v will be a leaf node in tree T_j.

We use 3 auxiliary sets of nodes V', V'' and X to help us distinguish the unvisited nodes, the leaves and the profitable nodes. More precisely, at any moment of the execution, V'' contains the set of nodes that remain to be checked, through which we get the induced tree T_i at every iterative step i. V' contains the profitable nodes and last, X contains the set of leaf nodes at each iterative step. Note that if a node v, checked at some moment of the execution, is found to be non profitable then we must remove v and all of its descendants from set V'.

Algorithm 3 returns the subtree of profitable nodes. Any DFS traversal of this subtree would give us the optimal walk.

Algorithm 3 An algorithm for CollectMaxBigBudget.

```
Input: $I = \langle T(V, E), r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$.
 1: for all $v \in V$ do
 2:     $gain(v) \leftarrow g(v)$;
 3: end for
 4: $V' \leftarrow V$; $V'' \leftarrow V$;
 5: $X \leftarrow \{v \in V : deg(v) = 1\}$;
 6: while $X \neq \emptyset$ do
 7:     choose $v \in X$ arbitrarily; $u \leftarrow parent(v)$;
 8:     if $gain(v) > 2 \cdot w(u, v)$ then
 9:         $gain(u) \leftarrow gain(u) + gain(v) - 2 \cdot w(u, v)$;
10:     else
11:         remove $v$ and all its children from $V'$;
12:     end if
13:     $V'' \leftarrow V'' \setminus \{v\}$;
14:     if $deg(u) = 1$ in the tree induced by $V''$ then
15:         $X \leftarrow X \cup \{u\}$;
16:     end if
17:     $X \leftarrow X \setminus \{v\}$;
18: end while
19: Return $V'$ and $gain(r)$;
```

Theorem 20. *Given an instance I for* CollectMaxBigBudget *problem, Algorithm 3 correctly computes the optimal closed walk for I.*

Proof. For any $v \in V'$, let $g^\star(v)$ denote the final value of $gain(v)$, i.e., when v becomes a leaf and its gain value cannot be further updated.

We will prove the correctness of the algorithm by induction. The induction hypothesis is that at the end of each iteration i of the while loop at line 18, for any node $v \in X$, where v is a leaf in the tree induced by V'', $g^\star(v)$ is the optimal gain in subtree T_v of T rooted at v.

For $i = 0$, it trivially holds, since X contains only the leaves of T and the optimal gain of a leaf l is its initial gain $g(l)$. Assume that the claim holds for $i = k$, we will prove that it holds for $i = k + 1$. Let v be the leaf chosen at line 7 of the algorithm in the $k + 1$ iteration and let u be its parent node. For v and for any leaf $l \in X \setminus \{u\}$, the claim holds from the induction hypothesis. It suffices to show that if u becomes a leaf in iteration $k + 1$, hence, is inserted in X (line 15), $g^\star(u)$ is the optimal gain of T_u. We will introduce the indices k and $k + 1$ in gain function to make the distinction of the values of the function between iteration k and $k+1$. We want to prove that the equation $g^\star(u) = g_{k+1}(u) = g(u) + \sum_{v \in A}(g^\star(v) - 2w(u, v))$, where A is the set of the children nodes of u that caused an update of $gain_j(u)$, for $j \in \{1, \dots, k+1\}$, when removed from X, is the optimal gain in subtree T_u. From the induction hypothesis we have that $\forall v \in A : g^\star(v) = OPT_g(I|_{T_v})$. We have

$$\begin{aligned}
g^\star(u) &= g(u) + \sum_{v \in A} (g^\star(v) - 2w(u,v)) \\
&= g(u) + \sum_{\substack{v \in \mathsf{ch}(u): \\ g^\star(v) > 2w(u,v)}} (g^\star(v) - 2w(u,v)) && \text{(by Algorithm 3)} \\
&= g(u) + \sum_{\substack{v \in \mathsf{ch}(u): \\ OPT_g(I|_{T_u}) > 2w(u,v)}} (OPT_g(I|_{T_v}) - 2w(u,v)) && \text{(by induction hypothesis)} \\
&= OPT_g(I|_{T_u}) && \text{(by Lemma 19)}
\end{aligned}$$

The last equality completes the induction.

□

4.2.2 Minimize Initial Energy

In the previous subsection we showed how to compute the maximum amount of energy we can collect at the root, under the assumption that the agent has sufficient energy ($B = 2W$). We are now interested in minimizing the amount of initial energy we need to provide to the agent in order to collect the maximum amount of energy at the root. Notice now that two closed walks on the same set of nodes but with different orders of edge traversals, may require different amounts of initial energy of the agent. Therefore, the order of edge traversals in this problem is important, as opposed to the previous case. For this reason, in this subsection, we are considering sorted sets of nodes, where the order of the nodes denotes the order in which the agent visits the nodes. The formal description of the problem follows.

Definition 21 (CollectMaxMinBudget).
Instance: $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$, where $T(V,E)$ is an undirected, weighted tree rooted at r, $g(v) \in \mathbb{N}^+$ denotes the profit of energy of node v and $w(e) \in \mathbb{N}^+$ denotes the weight of the edge e.
Feasible solution: (B, p) where $B \geq 0$ and p_B is any sustainable walk $p_B = v_0, e_1, v_1 \dots, e_\ell, v_\ell$, with $v_0 = v_\ell = r$, for an agent with initial energy B.
Goal: Minimize B such that $\mathsf{n\text{-}surplus}_\ell(p_B) = g_{\mathsf{max}}(I)$.

Given an instance I for CollectMaxMinBudget, let $OPT_d(I)$ denote the minimum amount of energy we need to provide initially to the agent in order to collect at the root $OPT_g(I)$ units of energy.

Lemma 22. *Consider an instance $I = \langle T, r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$ of* Collect-

MaxMinBudget *and an optimal walk p_{OPT} for I. For any $v \in p_{OPT}$, we have*

$$OPT_d(I|_{T_v}) = \max\Big\{w(u,v), \max_{1\leq j\leq |\mathsf{ch}(v)|}\{w(u,v)-g(v)+OPT_d(I|_{T_{v_j}})-\sum_{i=1}^{j-1} OPT_g(I|_{T_{v_i}})\}\Big\} \tag{4.2}$$

where $u = parent(v)$, $w(parent(r), r) = 0$ and $\mathsf{ch}(v) = \{v_1, v_2, \ldots, v_{|\mathsf{ch}(v)|}\}$ is sorted in non-decreasing order with respect to the values of $OPT_d(I|_{T_{v_i}})$.

Proof. First, we will prove that given any ordering of the children of a node v, equation 4.2 computes the minimum amount of initial energy needed by the agent to perform the traversal of the children nodes. Next, we will prove that, in order to minimize the amount of initial energy required, the optimal order of visiting the children of a node v is sorted in non-decreasing order with respect to the values $OPT_d(I|_{T_{v_i}})$ of the children of v.

Given an ordering of the children of a node v, equation 4.2 calculates a set, where the elements of this set correspond to the energy needed for the visit of v and of its children. More in particular, first, it computes the amount of energy needed to reach node v from its parent node. Next, for each visit of a child node v_i, it calculates the amount of energy that has already been collected before visiting v_i, and the amount of energy needed to visit v_i (before collecting the energy of v_i). The difference between these two amounts denotes the amount of extra energy needed by the agent to visit node v_i. It returns the maximum value over all, which is the minimum amount of energy needed to perform the given ordered traversal. Notice that the difference between the energy collected and the energy spent can be negative, such a case means that the agent can perform the traversal without requiring any extra energy. If every subtree returns a negative value, $OPT_d(I|_{T_r})$ will be equal to 0, since we have set the weight of the dummy edge connecting the root with its parent node to be equal to 0.

For the second part of this proof, for the sake of contradiction, consider that there exists an ordering $\mathsf{ch}(v) = \{v_1, v_2, \ldots, v_{|\mathsf{ch}(v)|}\}$ of the children of a node v that is not sorted in non-decreasing order with respect to the values of $OPT_d(I|_{T_{v_i}})$, which returns $OPT_d(I|_{T_v})$. Without loss of generality, consider that $\mathsf{ch}(v) = \{v_1, v_2\}$, for which it holds

$$OPT_d(I|_{T_{v_2}}) < OPT_d(I|_{T_{v_1}}) \tag{4.3}$$

The minimum energy required for visiting nodes v, v_1 and v_2 is calculated in equation 4.2 by choosing the maximum over the following values

$$\begin{aligned} x &= w(u,v) \\ y &= w(u,v) - g(v) + OPT_d(I|_{T_{v_1}}) \\ z &= w(u,v) - g(v) + OPT_d(I|_{T_{v_2}}) - OPT_g(I|_{T_{v_1}}) \end{aligned}$$

Let us now consider the traversal of v_1 and v_2 where the non-decreasing ordering is preserved. In this case equation 4.2 calculates

$$\begin{aligned}
x' &= w(u,v) \\
y' &= w(u,v) - g(v) + OPT_d(I|_{T_{v_2}}) \\
z' &= w(u,v) - g(v) + OPT_d(I|_{T_{v_1}}) - OPT_g(I|_{T_{v_2}})
\end{aligned}$$

We are only interested for the case where $w(u,v)$ is not greater than y, z, y' and z', which is the case where $OPT_d(I|_{T_{v_1}}) > g(v)$. Otherwise x would be the dominant term and both ordered sets would result in the same cost. Therefore, we have that $y > x$ and $y > x'$. By inequality 4.3, we get that $y > z$, hence, $OPT_d(I|_{T_v}) = y$. By inequality 4.3, we also get that $y > y'$ and $y > z'$, since $OPT_g(I|_{T_{v_i}}) > 0$, $\forall v_i \in p_{OPT}$. As a result, $y > \max\{x', y', z'\}$, which contradicts the optimality of $OPT_d(I|_{T_v})$. □

Algorithm 4 computes the minimum amount of energy needed to achieve a maximum gain of $g_{\mathsf{max}}(I)$ for an instance I and returns the optimal walk. At the first step of Algorithm 4, Algorithm 3 is called with the initial input of Algorithm 4. The output of Algorithm 3 is the subset of nodes V' of T in which we can collect at the root $g_{\mathsf{max}}(I)$ units of energy, therefore, we can restrict the calculations to the subtree T' induced by V', ignoring the non-profitable nodes. As in the previous algorithm it treats the nodes in a bottom-up manner, starting from the leaf nodes of T' which are contained in the auxiliary set of leaf nodes X. Algorithm 4 is an iterative algorithm and at each iteration i, it picks a node v from V' for which all of its children are leaves in X and calculates the minimum energy needed by the agent to visit v and its children. Next, it computes the maximum gain of energy the agent collects by visiting v and its children. During step 14, it creates the optimal walk of v and its children, which is in a non-decreasing order regarding the energy cost of visiting each child. Last, it removes the children of v from X, therefore, v becomes a leaf and is added in X. Once all the nodes of V' are checked, it exits the while loop and it returns the minimum amount of energy the agent needs in order to collect $g_{\mathsf{max}}(I)$ units of energy at the root and $walk(r)$ which is the optimal walk for I.

Theorem 23. *Given an instance I for* CollectMaxMinBudget *problem, Algorithm 4 correctly computes the minimum amount of initial energy B we need to provide to the agent in order to collect a total gain of $g_{\mathsf{max}}(I)$ at the root and returns the optimal walk for I.*

Proof. For any $v \in V'$, let $g^\star(v)$ denote the final value of $gain(v)$ and $d^\star(v)$ denote the final value of $d(v)$. We will prove the correctness of the algorithm by induction. The induction hypothesis is that at the end of each iteration i of the while loop at line 17, for any node $v \in X$, $g^\star(v)$ is the optimal gain in T_v, $g^\star(v) = OPT_g(I|_{T_v})$, and $d^\star(v)$ is the optimal amount of energy we need to provide to the agent initially in order to collect $g^\star(v)$ energy in T_v, $d^\star(v) = OPT_d(I|_{T_v})$.

Algorithm 4 An algorithm for CollectMaxMinBudget.

```
Input: $I = \langle T(V,E), r, (g(v))_{\forall v \in V}, (w(e))_{\forall e \in E} \rangle$.
 1: $V' \leftarrow$ CollectMaxBigBudget($I$);
 2: $gain(r) \leftarrow g(r)$; $d(r) = 0$; $walk(r) = \{r\}$;
 3: for all $v \in V' \setminus \{r\}$ do
 4:     $gain(v) \leftarrow g(v)$;
 5:     $d(v) \leftarrow w(parent(v), v)$;
 6:     $walk(v) \leftarrow \{v\}$;
 7: end for
 8: $X \leftarrow \{v \in V' : deg(v) = 1\}$;
 9: while $X \neq \{r\}$ do
10:     choose $u \in V'$ s.t. $\mathsf{ch}(u) \subseteq X$, breaking ties arbitrarily;
11:     let $v_1, \dots, v_{|\mathsf{ch}(u)|}$ be the children of $u$ in non-decreasing order w.r.t. $d(v_i)$
12:     $d(u) \leftarrow d(u) + \max\left\{0, -gain(u) + \max_{1 \le j \le |\mathsf{ch}(u)|}\{d(v_j) - \sum_{i=1}^{j-1} gain(v_i)\}\right\}$;
13:     $gain(u) \leftarrow gain(u) + \sum_{i=1}^{|\mathsf{ch}(u)|} \left(gain(v_i) - 2 \cdot w(u, v_i)\right)$;
14:     $walk(u) \leftarrow \{u; walk(v_1); \dots; walk(v_{|\mathsf{ch}(u)|}); u\}$;
15:     $X \leftarrow X \setminus \mathsf{ch}(u)$;
16:     $X \leftarrow X \cup \{u\}$;
17: end while
18: Return $d(r)$ and $walk(r)$;
```

For $i = 0$, it trivially holds, since X contains only the leaves of T' and the optimal gain of any leaf l is its initial gain $g(l)$ and the minimum amount of energy needed to reach the leaf is equal to the weight of its edge $w(parent(l), l)$. Assume that the claim holds for $i = k$, we will prove that it holds for $i = k + 1$. Let v be the node chosen at line 10 in the $k + 1$ iteration. By the induction hypothesis we have that $\forall u \in \mathsf{ch}(v)$ $g^\star(u) = OPT_g(I|_{T_u})$ and $d^\star(u) = OPT_d(I|_{T_u})$. By Lemma 19 it follows directly that the final value of $gain(v)$ calculated at line 13 of the algorithm is the optimal gain of v, $g^\star(v) = OPT_g(I|_{T_v})$ and by Lemma 22 it follows directly that the final value of $d(v)$ calculated at line 12 of the algorithm is the optimal amount of energy needed by the agent to collect $g^\star(v)$ energy in T_v, $d^\star(v) = OPT_d(I|_{T_v})$, which completes the induction.

Considering the optimality of $walk(r)$, first, we get from step 1 of the algorithm that $walk(r)$ will contain all the profitable nodes of T. Next, by Lemma 22 we know that the optimal order of traversals of the nodes is a non-decreasing one with respect to the values of $OPT_d(I|_{T_v})$, this property is preserved in line 14 of the algorithm, where the nodes are added in a non-decreasing order with respect to the values of $d(v)$. This completes the proof. □

5 Near-Gathering in Graphs

Introduction

In the problem of gathering a group of k agents is initially placed at the nodes of a given graph and the goal for the agents is to meet at a single node of the graph and stay there. The constraint on the energy levels of the agents that we are considering in this manuscript, may render gathering at a single point impossible. For this reason, we are considering the problem of near-gathering, where the goal is to relocate the agents, respecting their energy constraints, in a way that the maximum or the average distance between any two agents is minimized.

Unlike previous chapters, here, we consider a more general model. The agents start from distinct nodes of an arbitrary graph, having, potentially, distinct energy levels initially. For the objective of minimizing the maximum pairwise distance of the agents, we show that the problem is NP-hard to approximate within a $2 - o(1)$ factor. As a positive result, we present a matching 2-approximation algorithm. Furthermore, for the objective of minimizing the average pairwise distance of the agents, we present a $2(1 - \frac{1}{k})$-approximation algorithm.

The Model

A collection of k mobile agents is initially located at an arbitrary set of nodes $\{h_1, \ldots, h_k\}$ of an undirected general graph $G(V, E)$. The objective is to compute a schedule of agents moves, so that the maximum or the average pairwise distance of the agents is minimized. Each agent a_i has an initial energy budget denoted by b_i. Traversing each edge consumes one unit of energy. Therefore, agent a_i can traverse at most b_i edges. If an agent depletes its battery, then it cannot perform any other action.

Let $d(u, v)$ denote the length of the shortest path between nodes u and v. We denote by $N(h_i)$ the set of all nodes that agent a_i can reach from its initial position h_i, $N(h_i) = \{v \in V \,|\, d(h_i, v) \leq b_i\}$.

5.1 Minimizing the Diameter

In this subsection we describe and analyze a greedy algorithm for minimizing the maximum distance between any pair of agents. The formal definition of the problem is the following

Definition 24 (MinD).
Instance: $I = \langle G, k, \{h_i\}_{i \in 1,\ldots,k}, \{b_i\}_{i \in 1,\ldots,k} \rangle$, where $G(V, E)$ is an undirected, general

graph, k denotes the total number of agents, h_i denotes the homebase node of agent a_i and b_i denotes the initial energy of agent a_i.
Feasible solution: Any configuration $\mathbf{C} = (c_1, \ldots, c_k)$, where $\forall i \in \{1, \ldots, k\}$, $c_i \in V$ and $d(h_i, c_i) \leq b_i$.
Goal: Minimize $\textsc{Diam}(\mathbf{C})$, where $\textsc{Diam}(\mathbf{C}) = \max_{i,j \in 1,\ldots,k} d(c_i, c_j)$.

The main idea of Algorithm 5 is to fix a node in graph G as a *gathering point* and compute, next, the minimum distance each agent can have to this fixed point with respect to their energy constraints, breaking ties arbitrarily. Algorithm 5 greedily tests all nodes in G as possible gathering points and selects the node that minimizes the maximum distance between any pair of agents.

Algorithm 5 An approximation algorithm for MinD.

```
Input: An instance $\langle G, k, \{h_i\}_{i\in 1,\ldots,k}, \{b_i\}_{i \in 1,\ldots,k}\rangle$.
1: for each $v \in V$ do
2:     We define $\mathbf{C}^v = (c_1^v, \ldots, c_k^v)$, where $c_i^v$ is a vertex in $N(h_i)$ that minimizes
   the distance to $v$, breaking ties arbitrarily.
3:     Compute $\textsc{Diam}(\mathbf{C}^v)$.
4: end for
5: Return $\operatorname{argmin}_{\mathbf{C}^v : v \in V} \textsc{Diam}(\mathbf{C}^v)$.
```

Let $OPT_{MinD} = \max_{1 \leq i,j \leq k} d(o_i, o_j)$ denote the value of the optimal solution for MinD problem.

Theorem 25. *Algorithm 5 is a 2-approximation algorithm for MinD.*

Proof. Consider an instance $I = \langle G, k, \{h_i\}_{i \in 1,\ldots,k}, \{b_i\}_{i\in 1,\ldots,k}\rangle$ of MinD. Let $\mathbf{C}^{v^*} = (c_1, \ldots, c_k)$ be the configuration Algorithm 5 returned for instance I and let $\mathbf{C}_{OPT} = (o_1, \ldots, o_k)$ be the configuration of an optimal solution for the same instance. There exists an $o^* \in \mathbf{C}_{OPT}$ for which it holds that

$$o^* = \operatorname*{argmin}_{o_j \in \mathbf{C}_{OPT}} \max_{1 \leq i \leq k} d(o_i, o_j) \tag{5.1}$$

Consider now the configuration $\mathbf{C}^{o^*} = (c'_1, \ldots, c'_k)$ Algorithm 5 computed for node o^* in step 3. It holds that

$$\textsc{Diam}(\mathbf{C}^{v^*}) = \max_{1 \leq i,j \leq k} d(c_i, c_j) \leq \max_{1 \leq i,j \leq k} d(c'_i, c'_j) \tag{5.2}$$

otherwise Algorithm 5 would have chosen o^* as the gathering point and not v^*. Next, we apply the triangle inequality to the right-hand side of inequality 5.2 and we get

$$\begin{aligned}\max_{1\leq i,j\leq k} d(c_i', c_j') &\leq \max_{1\leq i\leq k} d(c_i', o^*) + \max_{1\leq j\leq k} d(o^*, c_j') \\ &\leq 2 \cdot \max_{1\leq i\leq k} d(c_i', o^*) \leq 2 \cdot \max_{1\leq i\leq k} d(o_i, o^*)\end{aligned}$$

where the last inequality holds because of step 2 of Algorithm 5, i.e., for each agent a_i, the final position c_i' has the minimum distance towards o^*. Last, by equality 5.1 we have

$$2 \cdot \max_{1\leq i\leq k} d(o_i, o^*) \leq 2 \cdot \max_{1\leq i,j\leq k} d(o_i, o_j)$$

Therefore, inequality 5.2 becomes

$$\max_{1\leq i,j\leq k} d(c_i, c_j) \leq 2 \cdot \max_{1\leq i,j\leq k} d(o_i, o_j) = 2 \cdot OPT$$

which completes the proof.

□

We will now show that this is the best approximation ratio possible for the MinD problem.

Theorem 26. *There exists no deterministic* $\big(2 - o(1)\big)$*-approximation algorithm for MinD, unless* P = NP*. This holds even if* $b_i = B\,\forall i$*.*

Proof. We will prove the theorem by a reduction from 3-SAT to MinD. 3-SAT is one of Karp's 21 NP-complete problems, [Kar72], and has the following definition:
Instance: A conjunctive normal form formula ϕ where each clause contains at most 3 different literals.
Question: Is ϕ satisfiable?
Initially, we want to construct an instance of MinD from an instance of 3-SAT. Let $x_1, \ldots, x_n$ be the variables and $C_1, \ldots, C_m$ be the clauses of a 3-SAT formula ϕ, where $m, n \in \mathbb{N}$. We can construct now a graph $G(V, E)$ in the following manner.
Set of nodes V:
Let

- $V_1 = \{v_i \,|\, 1 \leq i \leq n\}$ be the set of nodes that correspond to variables $x_1, \ldots, x_n$,
- $V_2 = \{v_i^{\mathrm{T}} \,|\, 1 \leq i \leq n\}$ be the set of nodes that correspond to the true-value assignments of variables x_i,
- $V_3 = \{v_i^{\mathrm{F}} \,|\, 1 \leq i \leq n\}$ be the set of nodes that correspond to the false-value assignments of variables x_i,
- $V_4 = \{c_i \,|\, 1 \leq i \leq m\}$ be the set of nodes that correspond to clauses $C_1, \ldots, C_m$,

- $L = \{\text{TTT, TTF, TFT, TFF, FTT, FTF, FFT, FFF}\}$ be the set of all possible truth assignments of a clause containing 3 literals and
- $V_5 = \{c_i^\ell \,|\, 1 \leq i \leq m, \forall \ell \in L\}$ be the set of nodes that correspond to all possible truth assignments of each clause C_i.

We can now define as the set of nodes of G, the following union of sets

$$V(G) = \{V_1\} \cup \{V_2\} \cup \{V_3\} \cup \{V_4\} \cup \{V_5\}$$

Set of edges E:

For simplicity, we will consider weighted edges, where the weight of each edge represents the length of the edge and is proportional to the amount of energy an agent needs in order to traverse this edge. The weighted graph G can be easily transformed into a graph with unweighted edges by replacing each weighted edge by a node-disjoint simple path of length equal to the weight of the edge. Throughout this proof, unless stated otherwise, we are assuming that the weight of the edges, denoted by $w(u, v)$, for $u, v \in V$, is uniform and equal to B.

Let

- $E_1 = \{(v_i, v_i^{\text{T}}) \mid \forall v_i \in V_1, \forall v_i^{\text{T}} \in V_2\}$ be the set of weighted edges between the set of nodes V_1 and V_2, where each variable is connected to the literal that corresponds to its true-value assignment,
- $E_2 = \{(v_i, v_i^{\text{F}}) \mid \forall v_i \in V_1, \forall v_i^{\text{F}} \in V_3\}$ be the set of weighted edges between the set of nodes V_1 and V_3, where each variable is connected to the literal that corresponds to its false-value assignment,
- $E_3 = \{(v_i, v_j) \mid \forall v_i \in V_2, \forall v_j \in V_3\}$ be the set of weighted edges between all nodes representing literals,
- $E_4 = \{(c_i, c_i^\ell) \mid \forall c_i \in V_4, \forall c_i^\ell \in V_5, c_i^\ell \text{ satisfies } C_i, \forall \ell \in L\}$ be the set of weighted edges that connects each clause node with nodes representing all possible satisfying assignments for this clause,
- $E_5 = \{(c_i^\ell, c_j^{\ell'}) \mid \forall c_i^\ell, c_j^{\ell'} \in V_5, i \neq j, \forall \ell, \ell' \in L\}$ be the set of weighted edges between all the possible satisfying assignments of the clauses,
- $E_6 = \{(v_i, c_j^\ell) \mid \forall v_i \in V_2 \cup V_3, \forall c_j^\ell \in V_5, c_j^\ell \text{ satisfies } C_j, x_i \text{ appears in } C_j\}$ be the set of weighted edges that connects each literal to all the satisfying assignments of the clauses in which it appears, and,
- $E_7 = \{(v_i^{\text{T}}, c_j^\ell), (v_i^{\text{F}}, c_j^\ell) \mid 1 \leq i \leq n, 1 \leq j \leq m, \forall \ell \in L, x_i \text{ does not appear in } C_j\}$ be the set of weighted edges that connects each literal to all the satisfying assignments of the clauses in which the corresponding variable x_i of the literal does not appear.

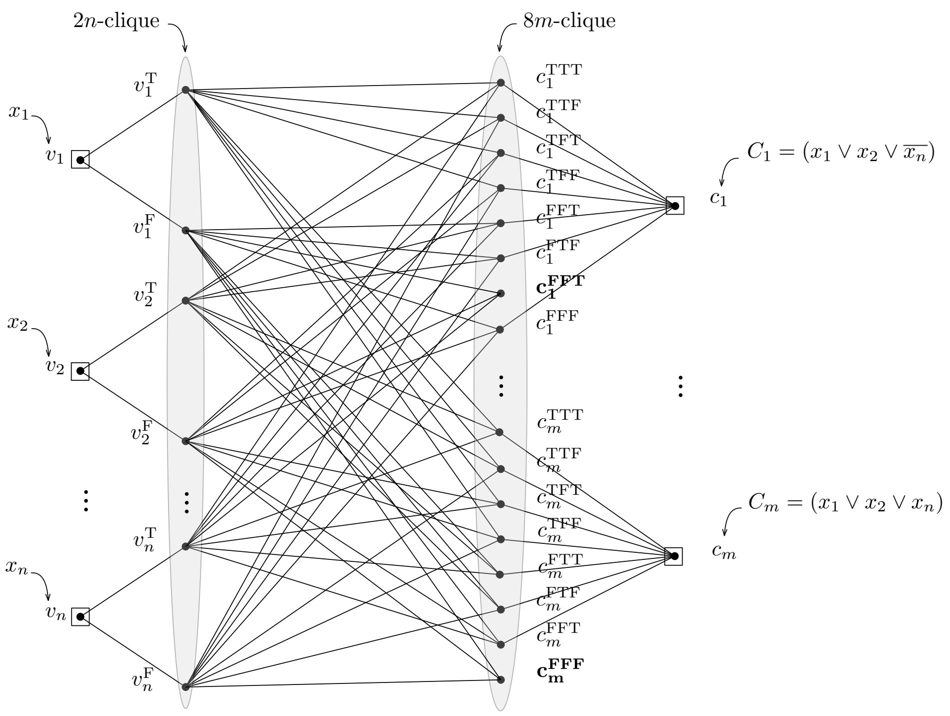

Figure 5.1: A part of an instance of MinD, constructed from 3-SAT instance $C_1 \wedge \cdots \wedge C_m$ with variables $x_1, \ldots, x_n$, displaying the connections between nodes v_1, v_2, v_n, c_1 and c_2. Notice that nodes c_1^{FFT} and c_m^{FFF} are not connected to nodes c_1 and c_m, respectively. The rectangles denote the presence of mobile agents.

We define as the set of edges of G, the following union of sets

$$E(G) = \{E_1\} \cup \{E_2\} \cup \{E_3\} \cup \{E_4\} \cup \{E_5\} \cup \{E_6\} \cup \{E_7\}$$

Agents:

Finally, we consider that $n + m$ agents are initially positioned at nodes $\{v_i \mid \forall v_i \in V_1 \cup V_4\}$ and each agent has B units of available energy. If an agent is initially placed on a node v, then we write $a(v)$ to refer to this agent at any point of the strategy.

Fig. 5.1 shows a part of an instance of MinD which is constructed from an instance of 3-SAT as described above. Before continuing with our proof we need to argue that no agent would stop in the middle of an edge. To prove this we state the following lemma.

Lemma 27. *For any solution* $\textsc{Diam}(\mathbf{C})$ *of MinD, with* $\mathbf{C} = (c_1, \ldots, c_k)$*, for which* $\exists i \in \{1, \ldots, k\}$ *such that* $c_i \notin \{V_2 \cup V_3 \cup V_5\}$*, there exists another solution*

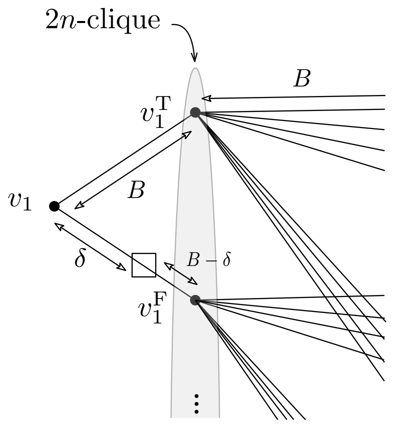

Figure 5.2: An instance where agent $a(v_1)$, represented by the rectangle, stops at distance δ from its starting node.

$\textsc{Diam}(\mathbf{C}')$, with $\mathbf{C}' = (c_1', \ldots, c_k')$, for which $\forall i \in \{1, \ldots, k\}$, $c_i' \in \{V_2 \cup V_3 \cup V_5\}$ and $\textsc{Diam}(\mathbf{C}') < \textsc{Diam}(\mathbf{C})$.

Proof. Consider an agent $a(v_i)$ which corresponds to the variable x_i and without loss of generality, suppose that $a(v_i)$ chooses to move towards node u_i^{F}. Assume now, for the sake of contradiction, that agent $a(v_i)$ has stopped at distance $0 \leq \delta < B$ on the edge (v_i, v_i^{F}), therefore $a(v_i)$ has spent δ units of energy and has $B - \delta$ units of energy remaining (Fig. 5.2). Agent $a(v_i)$ is connected with the rest of the agents through two possible paths, the first one is through the segment $(\delta, v_i^{\mathrm{F}})$ of length $B - \delta$ which we will call p^{F} and the other one is through the path $(\delta, v_i), (v_i, v_i^{\mathrm{T}})$ of length $\delta + B$, which we will call p^{T}. It holds that, $a(v_i)$ has a total distance of $B - \delta + B = 2B - \delta$ towards the other agents through p^{F}, where $B - \delta$ corresponds to the distance of $a(v_i)$ to node v_i^{F} and B corresponds to the distance between node v_i^{F} and any node $v_j \in \{V_2 \cup V_3 \cup V_5\}$, and a total distance of $B + \delta + B = 2B + \delta$ towards the rest of the agents through p^{T}, where $B + \delta$ corresponds to the distance of $a(v_i)$ to node v_i^{T} (through v_i) and B corresponds to the distance between node v_i^{T} and any node $v_j \in \{V_2 \cup V_3 \cup V_5\}$.

It is easy now to notice that if $a(v_i)$ moves to node v_i^{F} (recall that it has $B - \delta$ units of energy remaining) its distance to the rest of the agents can only be reduced, more precisely, it will be equal to B through node v_i^{F} and $2B$ through node v_i^{T}, which contradicts the optimality of the solution. The same argument holds for the agents that correspond to the clauses. Therefore, in any optimal solution no agent stops in the middle of an edge.

□

Continuing with our proof, first, we prove that if ϕ is satisfiable then there exists a solution to MinD of size less than or equal to B.

$(\Longrightarrow)$

Given the fact that ϕ is satisfiable we have a truth assignment to the variables which satisfies every clause of ϕ. For each $i \in \{1, \ldots, n\}$, if variable $x_i :=$ TRUE then agent $a(v_i)$ moves to node v_i^{T}, otherwise, it moves to node v_i^{F}. Next, for each $i \in \{1, \ldots, m\}$, agent $a(c_i)$ moves to node c_i^{ℓ}, which corresponds to the unique truth assignment to the variables that makes all the literals that are contained in c_i clause TRUE. Note that any agent initially located in either a variable node or a clause node, has sufficient energy to reach any node in the union of sets of nodes $V_2 \cup V_3$, or in the set of nodes V_5 respectively.

Let us examine the maximum distance of any two agents in this final configuration. Notice that the agents $\{a(v_i) \mid v_i \in V_1\}$ moved to nodes that belong to $V_2 \cup V_3$. By construction, we have $\forall v_i, \forall v_j \in V_2 \cup V_3, i \neq j$, $w(v_i, v_j) = B$ (refer to the set of edges E_3). Therefore, the maximum distance of any two agents that have moved to the union of sets of nodes $V_2 \cup V_3$ is equal to B. Similarly, the agents $\{a(c_i) \mid c_i \in V_4\}$ have moved to nodes that belong to the set of nodes V_5. Again, by construction we have $\forall c_i, \forall c_j \in V_5, i \neq j$, $w(c_i, c_j) = B$ (refer to the set of edges E_5), as a result the maximum distance of any two agents in V_5 is equal to B. It remains to compute the distance between any agent that is located in a node in the union of sets of nodes $V_2 \cup V_3$ and any agent that is located in a node of V_5. Each agent $a(c_i)$, located in the set of nodes V_5, by construction has distance equal to B from the nodes that correspond to the truth assignment to the variables contained in clause c_i (refer to the set of edges E_6). Moreover, each agent $a(c_i)$ has distance B from the nodes that belong to the set $V_2 \cup V_3$ and correspond to the truth assignment to variables that are not contained in clause c_i (refer to the set of edges E_7). Therefore, the maximum distance between any two agents is equal to B.

For the other direction of the reduction, we need to prove that if ϕ is not satisfiable then every solution to MinD is of size greater than or equal to $2B$.

$(\Longleftarrow)$

We can assume now that no agent will stop in the middle of an edge, according to Lemma 27.

If ϕ is not satisfiable then for every possible truth assignment to the variables, there exists at least one clause in ϕ that is not satisfied. Let us note here that in any solution to MinD, the final positions of the agents that are initially located in variable nodes correspond to a truth assignment to the variables. Therefore, any final configuration will correspond to a truth assignment to the variables which will not satisfy ϕ. Consider now an arbitrary final configuration of an instance of MinD. For the corresponding truth assignment to the variables, without loss of generality, assume that the clause that is not satisfied is $C_l = (x_i \vee \overline{x_j} \vee x_k)$. This implies that agents $a(v_i), a(v_j)$ and $a(v_k)$ are located in nodes v_i^{F}, v_j^{T} and v_k^{F}, respectively.

Let us examine the maximum distance of any two agents in this final configuration. Recall that the set of edges E_4 connects each clause node to nodes corresponding to all possible satisfying assignments for this clause, as a result, nodes c_l and c_l^{FTF} are not connected by an edge. Moreover, the shortest path between nodes c_l and c_l^{FTF} is equal to $2B$ (refer to the set of edges E_5). Therefore, agent $a(c_l)$ cannot reach node c_l^{FTF}. Any other node c_l^{ℓ}, where $\ell \in L \setminus \{\text{FTF}\}$, to which agent $a(c_l)$ could relocate, corresponds to a truth assignment to x_i, x_j and x_k where at least one of the variables has the opposite value of its assignment. Say that $a(c_l)$ chooses to move to node c_l^{TTF}, then $a(c_l)$ will have a distance of $2B$ from agent $a(v_i)$ since $a(v_i)$ has moved to node v_i^{F}. Recall that node v_i^{F} is not connected by an edge to node c_l^{TTF} since v_i^{F} does not appear in it. Therefore, agents $a(c_l)$ and $a(v_i)$ will have a distance of $2B$. This completes the proof.

□

Notice that if B is very small, we can achieve an additive approximation of $+2B$, by assigning to the agents to remain in their initial positions.

5.2 Minimizing the Average Distance

In this subsection we describe and analyze a greedy algorithm for minimizing the average pairwise distance between agents. The formal definition of the problem is the following

Definition 28 (MinSum)**.**
Instance: $I = \langle G, k, \{h_i\}_{i \in 1,\dots,k}, \{b_i\}_{i \in 1,\dots,k} \rangle$, where $G(V, E)$ is an undirected, general graph, k denotes the total number of agents, h_i denotes the homebase node of agent a_i and b_i denotes the initial energy of agent a_i.
Feasible solution: Any configuration $\mathbf{C} = (c_1, \dots, c_k)$, where $\forall i \in \{1, \dots, k\}$, $c_i \in V$ and $d(h_i, c_i) \leq b_i$.
Goal: Minimize $\textsc{Sum}(\mathbf{C})$, where $\textsc{Sum}(\mathbf{C}) = \frac{1}{2} \sum_{1 \leq i \neq j \leq k} d(c_i, c_j)$.

The main idea of Algorithm 6 is similar to the idea of Algorithm 5. We fix a node in the graph G as a gathering point and we compute, next, the minimum distance each agent can have to this fixed point with respect to their energy constraints, breaking ties arbitrarily. Algorithm 6 greedily tests all nodes in G as possible gathering points and selects the node that minimizes the average distance between any pair of agents.

Algorithm 6 An approximation algorithm for MinSum.

```
Input: An instance $\langle G, k, \{h_i\}_{i \in 1,\dots,k}, \{b_i\}_{i \in 1,\dots,k} \rangle$.
1: for each $v \in V$ do
2:     We define $\mathbf{C}^v = (c_1^v, \dots, c_k^v)$, where $c_i^v$ is a vertex in $N(h_i)$ that minimizes
   the distance to $v$, breaking ties arbitrarily.
3:     Compute $\textsc{Sum}(\mathbf{C}^v)$
4: end for
5: return $\operatorname{argmin}_{\mathbf{C}^v : v \in V} \textsc{Sum}(\mathbf{C}^v)$.
```

Let $OPT_{MinSum} = \frac{1}{2}\sum_{1\leq i\neq j\leq k} d(o_i, o_j)$ denote the value of the optimal solution for MinSum problem.

Theorem 29. *Algorithm 6 is a* $2(1-\frac{1}{k})$*-approximation algorithm.*

Proof. Consider an instance $I = \langle G, k, \{h_i\}_{i\in 1,\ldots,k}, \{b_i\}_{i\in 1,\ldots,k}\rangle$ of MinSum. Let $\mathbf{C}^{v^*} = (c_1, \ldots, c_k)$ be the configuration Algorithm 6 returned for I and let $\mathbf{C}_{OPT} = (o_1, \ldots, o_k)$ be the configuration of an optimal solution for the same instance. There exists an $o^* \in \mathbf{C}_{OPT}$, which has the following property

$$OPT_{MinSum} = \frac{1}{2}\sum_{1\leq i\neq j\leq k} d(o_i, o_j) \geq \frac{1}{2}\sum_{1\leq i\neq j\leq k} d(o^*, o_j) = \frac{k}{2}\sum_{1\leq j\leq k} d(o^*, o_j)$$

hence, we have

$$\sum_{1\leq j\leq k} d(o^*, o_j) \leq \frac{2}{k}\cdot OPT_{MinSum} \tag{5.3}$$

Consider now the configuration $\mathbf{C}^{o^*} = (c'_1, \ldots, c'_k)$ Algorithm 6 computed for node o^* in step 3. It holds that

$$\textsc{Sum}(\mathbf{C}^{v^*}) = \frac{1}{2}\sum_{1\leq i\neq j\leq k} d(c_i, c_j) \leq \frac{1}{2}\sum_{1\leq i\neq j\leq k} d(c'_i, c'_j) \tag{5.4}$$

since Algorithm 6 at step 5 rejected o^* and chose v^* as the gathering point. By applying the triangle inequality to the right-hand side of inequality 5.4 we have

$$\begin{aligned}\frac{1}{2}\sum_{1\leq i\neq j\leq k} d(c'_i, c'_j) &\leq \frac{1}{2}\sum_{1\leq i\neq j\leq k}\Big(d(c'_i, o^*) + d(o^*, c'_j)\Big)\\ &\leq (k-1)\sum_{1\leq j\leq k} d(o^*, c'_j) \leq (k-1)\sum_{1\leq j\leq k} d(o^*, o_j)\end{aligned}$$

where the last inequality holds because of step 2 of Algorithm 6, i.e., for each agent a_i, the final position c'_i has the minimum distance towards o^*. We can apply now inequality 5.3 to the right-hand side of the inequality above, we get

$$\frac{1}{2}\sum_{1\leq i\neq j\leq k} d(c'_i, c'_j) \leq (k-1)\frac{2}{k}\cdot OPT_{MinSum}$$

Therefore, inequality 5.4 becomes

$$\textsc{Sum}(\mathbf{C}^{v^*}) \leq 2\,(1-\frac{1}{k})\cdot OPT_{MinSum}$$

which completes the proof.

□

We expect that our gap introducing reduction for MinD, can be modified to provide inapproximability results for the MinSum problem as well.

6 Delivery with Energy Sharing

Introduction

We consider the problem of delivering a package from a source node to a target node in a graph. Since the agents have energy constraints, multiple agents need to collaborate to move the package. As in the previous chapter, we consider that the agents may start from distinct nodes of the graph. However, unlike the previous chapters, we allow agents to share energy when they meet. We study the collaborative delivery problem for two cases. In the first case, we assume that there is no restriction on the amount of energy the agents can share (cf. section 6.1). In the second case, we assume that the agents have restricted energy capacity, that is, the amount of available energy the agents have, at any moment of the execution, cannot be greater than the amount of their initial energy B (cf. section 6.2).

For the first scenario, it was shown that collaborative delivery is weakly NP-hard in general graphs ([Czy+16]). In this work, we prove that the problem is strongly NP-hard in general graphs by a reduction from the Steiner Tree problem and we use this idea to get an algorithm for collaborative delivery. We prove, next, that if we augment the initial energy of the agents by a constant factor, we can solve collaborative delivery in polynomial time. For the second scenario, we completely solve the problem of collaborative delivery for $B = 2$. We define two versions of the problem. In the first version of the problem called CollaborativeDelivery *with Fixed Placement*, the initial placement of agents, i.e., the energy distribution, is given as part of the problem. In the second version of the problem called CollaborativeDelivery *with Chosen Placement*, a set of homebase nodes is given and the algorithm may choose the distribution of agents among the homebase nodes. We show that the first version of the problem is strongly NP-hard, while the second version of the problem admits a polynomial time solution and we present such a solution strategy.

This chapter contains results that have appeared in conference publication [Bam+17b].

The Model

Given a simple undirected graph $G = (V, E)$, with two special nodes s (source) and t (target), and a collection of k mobile agents located initially in specific nodes of the graph, the objective is to decide whether there is schedule of agent moves that can deliver a package from s to t. Each agent a_i has an initial energy budget and we denote its value by B. Traversing each edge consumes one unit of energy, thus, a fully charged agent can move a distance of B before running out of energy. When two agents are at the same node, one agent can transfer to the other agent, any integral part of its energy. As we mentioned previously, in section 6.1, we consider that there is no constraint on the amount of energy two agents can share, as opposed to section 6.2, where we introduce the constraint that no agent can have more than B units of energy at any time.

There is a unique package initially at node s that needs to be moved to node t. To simplify the discussion, we will assume that the system is synchronous (any synchronous strategy can also be implemented in an asynchronous system using appropriate waits). An agent a_i located at a node v at time t and having some positive energy, can perform any subset of the following actions:

- Pick up the package, if the package is present at v at time t.
- Transfer a part of its energy to another agent a_j that is located at v at time t. For the constrained version, under the restriction that a_j is not fully charged.
- Move to a neighboring node u, consuming one unit of energy and arriving at u at time $t + 1$.

A solution strategy is a sequence of steps as above, such that after the last step, the package is located at node t.

6.1 (Unlimited Capacity) Energy Sharing

Let us start this section with the formal definition of the problem we will be studying.

Definition 30 (CollaborativeDelivery *with Unlimited Capacity*)**.** CDU
Instance: $\langle G, s, t, k, H, B\rangle$, where $G = (V, E)$ is a simple undirected graph, $s, t \in V$ are, respectively, the source and target nodes, $k \geq 1$ is the number of mobile agents, $H \subset V$ is the set of homebase nodes, where exactly 1 agent is initially located at each homebase, and $B \in \mathbb{Z}_+$ is the initial energy of the agents.
Question: Does there exist a solution strategy for moving the package from s to t?

6.1.1 Hardness

Lemma 31. CDU *is* NP*-hard in general graphs.*

Proof. In order to prove the NP-hardness of CDU we will use the Steiner Tree problem which is a known NP-complete problem by [GJ79].

Definition 32 (Steiner Tree)**.** ST
Instance: $\langle G_{\text{ST}}, R, Q\rangle$, where $G_{\text{ST}} = (V_{\text{ST}}, E_{\text{ST}})$ is a simple undirected graph, $R \subseteq V_{\text{ST}}$ and Q is a positive integer with $Q \leq |V_{\text{ST}}| - 1$.
Question: Does there exist a subtree $T'_{\text{ST}} = (V'_{\text{ST}}, E'_{\text{ST}})$ of G_{ST} that includes all the vertices of R and $|E'_{\text{ST}}| \leq Q$?

Given an instance $I_{\text{ST}} = \langle G_{\text{ST}}, R, Q \rangle$ of ST we can construct an instance $I_{\text{CDU}} = \langle G_{\text{CDU}}, s, t, k, H, B \rangle$ of CDU as follows. Initially, we set $G_{\text{CDU}} = G_{\text{ST}}$, $H = R$, $k = |R|$ and $B = 2|V_{\text{ST}}|$. Next, we choose arbitrarily a node v that belongs to H and we set it to be the source node, $s = v$. Last, we attach to s a node-disjoint simple path of length $kB - Q$ where the node on the other endpoint of this path we set it to be the target node t.

For the next step of our proof, we need to show that if there exists a solution for I_{ST} then there exists a solution for I_{CDU}. By definition, if there exists a solution $\mathcal{S}$ for I_{ST}, then there exists a subtree T'_{ST} of G_{ST} for which $|E'_{\text{ST}}| \leq Q$. Subtree T'_{ST} of G_{ST} corresponds to a subtree T'_{CDU} of G_{CDU}, where every leaf node of T'_{CDU} corresponds to a homebase node. Let us note here that by construction, the trees T'_{ST} and T'_{CDU} have the same sets of nodes and edges, for this reason we can drop the indices ST and CDU on T', V' and E' without causing ambiguity. Next, we assign to each agent to move towards the source node s following a simple path in T'. We know that each agent has sufficient energy to reach s, since we have set their energy to be equal to $2|V_{\text{ST}}|$ and by definition $|V'| \leq |V_{\text{ST}}|$. If in this strategy, more than one agents are assigned to traverse the same edge, then one agent collects the energy of all others, so that a single agent traverses the edge. This way, every edge in T' is traversed exactly once. Following this strategy, in the end, we gather at s, $kB - |E'| \geq kB - Q$ units of energy. Next, if more than one agents are located at s, one of the agents collects all the $kB - |E'|$ units of energy, picks up the package and moves to t through the unique simple path of length $kB - Q$ in order to deliver the package. Hence, we get a solution for I_{CDU}.

For the other direction of the reduction, we need to prove that if there exists no feasible solution for I_{CDU} then there exist no solution for I_{ST}. If there exist no solution for I_{CDU}, then it does not exist a schedule of moves of the agents to collect $kB - Q$ units of energy at the source. In other words, the total number of edge traversals they perform to reach s, is greater than Q. Therefore, there exist no tree T' of G_{CDU} spanning all the nodes of H with total number of edges less than or equal to Q. Which implies that there is no solution for I_{ST}.

Notice that the hardness of the problem does not come from the fact that we have to choose a subset of the agents. Even if all the agents are to be used, we still cannot solve the problem in polynomial time.

□

6.1.2 Solution

Since the problem is NP-hard, we will try to find a *resource augmented* solution for the problem that can be computed in polynomial time. Given an instance I of CollaborativeDelivery *with Unlimited Capacity*, we consider augmenting the amount of initial energy of the agents by a constant factor of λ, we denote the new instance by λI. We call a solution for such an instance, a resource augmented solution. In addition, we say that an algorithm is λ-resource augmented if it correctly returns either that there is no feasible schedule for the original instance, or a feasible schedule for the augmented instance λI.

For the problem of CollaborativeDelivery *with Unlimited Capacity*, consider again an instance I for which there exists a feasible solution $\mathcal{S}$. Without loss of generality, we can assume that the package is carried from s to t by only one agent on a simple path p. We can see that this claim holds, by noticing that there always exists a solution in which the agent that picks up the package from the source, say a_0, also collects the energy each time it meets another agent along its path. Therefore, a_0 can eventually deliver the package to the target. All the other agents which are part of $\mathcal{S}$ either give their energy to a_0 directly (by meeting a_0 on a node of p) or indirectly, by transferring their energy to another agent a_i located on a node that does not belong to p. Such a path p we call it the *data path* of $\mathcal{S}$.

Assume that $\ell + 1$ agents, $a_0, \ldots, a_\ell$, have been relocated to path p. More precisely, consider that a_0 has moved to the source $s = v_0$ (if there was not any agent initially located at the source) and the rest of the agents have relocated to nodes $v_1, \ldots, v_\ell$ in order to meet a_0 and share their energy (we will not consider any agents that possibly meet agents $a_0, \ldots, a_\ell$ before they reach p). In this manner, we have that along the data path p there are $\ell + 1$ nodes, which we will denote by $s = v_0, v_1, \ldots, v_\ell$, where $\ell + 1$ agents are located and the available energy of each one of these agents is denoted by $b_0, b_1, \ldots, b_\ell$, additionally, let $v_{\ell+1} = t$. Let d_i denote the distance between nodes v_i and v_{i+1}, $0 \le i \le \ell$ in p. If there exists an agent that meets a_0 at t then this agent does not contribute its energy for the delivery of the package, thus, it is not part of $\mathcal{S}$.

Remark 1. *It is easy to see that, given that I has a solution $\mathcal{S}$ for* CDU*, for the data path p of $\mathcal{S}$ it holds that $\forall j \in \{0, \ldots, \ell\}$, $\sum_{i=0}^{j} b_i \ge \sum_{i=0}^{j} d_i$.*

The following lemma states that given that there exists a solution for an instance of CDU, if we augment the initial energy of the agents, by a factor of 2, we can collect sufficient energy at the source, such that an agent will be able to deliver the data from s to t by following the data path p.

Lemma 33. *Consider an instance I of* CDU*. If there exists a solution $\mathcal{S}$ for I, then for the data path p of $\mathcal{S}$, it holds that $\exists w \in \{1, \ldots, \ell\}$ such that*

$$2\sum_{i=j}^{w} b_i \ge \sum_{i=j}^{w} d_{i-1}, \quad \forall j \in \{1, \ldots, w\} \qquad (C_{w,j})$$

and

$$2\sum_{i=0}^{w} b_i - \sum_{i=0}^{w-1} d_i \ge \sum_{i=0}^{\ell} d_i \qquad (C_{w,0})$$

Proof. In order to prove Lemma 33 we will use the following two propositions.

Proposition 34. *If for some λ, $\bigwedge_{w=0}^{\lambda} \bigvee_{j=0}^{w} \neg C_{w,j}$, then $\forall \mu \in \{0, \ldots, \lambda\}$:*

$$\sum_{i=0}^{\mu} b_i < \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{\mu-1} d_i$$

Proof. We will prove this proposition by induction.
For $\lambda = 0$, it has to hold that $\neg C_{0,0}$ implies $b_0 < \frac{1}{2} \sum_{i=0}^{\ell} d_i$, which is true by the definition of $C_{0,0}$, $C_{0,0} \equiv 2b_0 \geq \sum_{i=0}^{\ell} d_i$. For the induction step, assume that it is true for $\lambda = k$. We will show that it is true for $\lambda = k+1$. Assume $\bigwedge_{w=0}^{k+1} \bigvee_{j=0}^{w} \neg C_{w,j}$, which is equivalent to

$$\bigwedge_{w=0}^{k} \bigvee_{j=0}^{w} \neg C_{w,j} \wedge \bigvee_{j=0}^{k+1} \neg C_{k+1,j}$$

which implies by the induction hypothesis

$$\forall \mu \in \{0, \ldots, k\}, \sum_{i=0}^{\mu} b_i < \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{\mu-1} d_i$$

and

$$\bigvee_{j=0}^{k+1} \neg C_{k+1,j}$$

Hence, we need to prove that

$$\forall \mu \in \{0, \ldots, k+1\}, \sum_{i=0}^{\mu} b_i < \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{\mu-1} d_i$$

By the induction hypothesis we have that this inequality holds for all $\mu \in \{0, \ldots, k\}$. It remains to show that $\sum_{i=0}^{k+1} b_i < \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{k} d_i$. Since we have that $\bigvee_{j=0}^{k+1} \neg C_{k+1,j}$ is true, we can distinguish two cases.
*C*ase 1: If $\neg C_{k+1,\xi}$ for some $\xi \in \{1, \ldots, k+1\}$, we have

$$\sum_{i=0}^{k+1} b_i = \sum_{i=0}^{\xi-1} b_i + \sum_{i=\xi}^{k+1} b_i \overset{\text{ind. hyp.}}{<} \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{\xi-2} d_i + \sum_{i=\xi}^{k+1} b_i \overset{\text{def. of } \neg C_{k+1,\xi}}{<}$$
$$\frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{\xi-2} d_i + \frac{1}{2} \sum_{i=\xi}^{k+1} d_{i-1} \leq \frac{1}{2} \sum_{i=0}^{\ell} d_i + \frac{1}{2} \sum_{i=0}^{k} d_i$$

*C*ase 2: If $\neg C_{k+1,0}$, by definition

$$2 \sum_{i=0}^{k+1} b_i - \sum_{i=0}^{k} d_i < \sum_{i=0}^{\ell} d_i$$

which is equivalent to

$$\sum_{i=0}^{k+1} b_i < \frac{1}{2}\sum_{i=0}^{\ell} d_i + \frac{1}{2}\sum_{i=0}^{k} d_i$$

.

□

Proposition 35. *If* $\bigwedge_{w=0}^{\ell-1} \bigvee_{j=0}^{w} \neg C_{w,j}$*, then* $\bigwedge_{j=0}^{\ell} C_{\ell,j}$*.*

Proof. First, we will prove $\forall \xi \in \{1, \ldots, \ell\}$, $C_{\ell,\xi}$ which is equivalent to prove $2\sum_{i=\xi}^{\ell} b_i \geq \sum_{i=\xi}^{\ell} d_{i-1}$. By remark 1, we have that $\sum_{i=0}^{\ell} b_i \geq \sum_{i=0}^{\ell} d_i$ which can be written as $\sum_{i=0}^{\xi-1} b_i + \sum_{i=\xi}^{\ell} b_i \geq \sum_{i=0}^{\ell} d_i$. Thus, we have

$$\sum_{i=\xi}^{\ell} b_i \geq \sum_{i=0}^{\ell} d_i - \sum_{i=0}^{\xi-1} b_i \overset{\text{Prop. 34}}{>}$$
$$\sum_{i=0}^{\ell} d_i - \frac{1}{2}\sum_{i=0}^{\ell} d_i - \frac{1}{2}\sum_{i=0}^{\xi-2} d_i =$$
$$\frac{1}{2}\sum_{i=\xi-1}^{\ell} d_i \geq \frac{1}{2}\sum_{i=\xi}^{\ell} d_{i-1}$$

Finally, to prove $C_{\ell,0}$ we need to prove that $2\sum_{i=0}^{\ell} b_i - \sum_{i=0}^{\ell-1} d_i \geq \sum_{i=0}^{\ell} d_i$. Again, by remark 1 we have

$$\sum_{i=0}^{\ell} b_i \geq \sum_{i=0}^{\ell} d_i \geq \sum_{i=0}^{\ell} d_i - \frac{d_\ell}{2} \geq$$
$$\frac{1}{2}\sum_{i=0}^{\ell} d_i + \frac{1}{2}\sum_{i=0}^{\ell-1} d_i \geq \sum_{i=0}^{\ell} d_i$$

which completes the proof.

□

The proof of Lemma 33 follows directly from Proposition 35.

□

Due to Lemma 33, we know that for any solution using data path p there is a node w on the path such that in the 2-resource augmented version of the problem, the agent arriving at node w can move to the source s collecting the energy of other agents along p and when it arrives at s it will have enough energy left to traverse p. Therefore, it is possible to collect at s enough energy, sufficient to move the package from s to t.

We now want to solve the problem of collecting sufficient energy at s. In order to do so, we will reduce CDU to the rooted version of the K-Steiner tree problem, which is defined as follows.

Definition 36 (Rooted K-Steiner Tree)**.** K-ST
Instance: $\langle G, r, (c(e))_{\forall e \in E}, R, K\rangle$, where $G = (V, E)$ is a simple undirected graph, $r \in V$ is the root node, $c(e)$ denotes the non-negative cost of edge e, $R \subseteq V$ is the set of required nodes (terminals) and K is an integer.
Feasible solution: A tree $T = (V', E')$ in G, with $V' \subseteq V$, $r \in V'$ and $E' \subseteq E$, that spans at least K of the required nodes.
Goal: Minimize $\text{cost}(T) = \sum_{e \in E'} c(e)$.

In a work of [BP89] it was proved that K-ST is an NP-hard problem. It is also known that it admits constant factor approximation algorithms, the currently best known approximation factor is 5 by [CRW04]. In this work, for simplicity we will denote the approximation factor of K-ST by α.

Before proceeding to the formal definition of our algorithm for CDU, let us describe informally its the main idea. As it was previously stated, we will make a reduction to the K-ST problem. Given an instance $I_{\text{CDU}} = \langle G, s, t, k, H, B\rangle$ of the CDU problem and an integer $K \leq |H|$, we can construct an instance $I_{K\text{-ST}} = \langle G_{K\text{-ST}}, r, (c(e))_{\forall e \in E}, R, K\rangle$ of K-ST by setting $G_{K\text{-ST}} = G$, $r = s$, $c(e) = 1$, $\forall e \in E$ and $R = H$. A solution T for instance $I_{K\text{-ST}}$ corresponds to a schedule of agent moves for instance I_{CDU}. That is, if a terminal node is included in T, then the agent initially located at the corresponding homebase node in I_{CDU} moves towards the source following the path in T, while it has sufficient energy. Note that, if two or more, agents are assigned to traverse the same edge, one agent collects the energy of all others, so that a single agent traverses the edge. On the other hand, given that I_{CDU} has a solution, then we know from Lemma 33 that if we augment the initial energy of the agents by a factor of 2, we can collect enough energy at the source node so that an agent can deliver the package to the target node. We can, then, start calling the algorithm for K-ST for the constructed instance $I_{K\text{-ST}}$ for different values of K (between 1 and k), until we get a corresponding schedule of agent moves in I_{CDU}, that collects at the source the desired amount of energy.

We denote by $d(u, v)$ the shortest path between nodes u and v.

Algorithm 7 An algorithm for CDU.

```
Input: $I_{\text{CDU}} = \langle G, s, t, k, H, B\rangle$.
1: $D \leftarrow d(s,t)$
2: for $i = 1, \ldots, k$ do
3:     Construct $I_{K\text{-ST}}$ with $K = i$
4:     if $I_{K\text{-ST}}$ has a solution $T_\alpha$ for which holds $2\alpha i B - \text{cost}(T_\alpha) \geq D$ then
5:         Return $T_\alpha$
6:     else Continue
7:     end if
8: end for
```

Theorem 37. *Algorithm 7 returns a 2α-resource augmented solution for* CDU*.*

Proof. Given an instance I_{CDU} of CDU, by Lemma 33, if it is solvable this implies that for the augmented instance $2I_{\text{CDU}}$ we can collect $d(s,t)$ units of energy at source node s. The latter claim holds because the length of $d(s,t)$ is less than or equal to the length of the data path p of any solution. The schedule of agent moves in such a solution correspond to a tree T_{CDU} rooted at s, and j agents, where $j \in \{1,\dots,k\}$. Such a tree T_{CDU} corresponds to a solution for the $I_{K\text{-ST}}$ instance with $K = j$ of K-ST but not necessarily to an optimal solution. For T_{CDU} holds

$$2jB - \text{cost}(T_{\text{CDU}}) \geq d(s,t) \tag{6.1}$$

Let us consider now the optimal solution T_{OPT} for the instance $I_{K\text{-ST}}$ with $K = j$. Because of the optimality of T_{OPT}, we have $\text{cost}(T_{OPT}) \leq \text{cost}(T_{\text{CDU}})$, therefore inequality 6.1 can be written as

$$2jB - \text{cost}(T_{OPT}) \geq d(s,t)$$

We now multiply both sides by α.

$$2\alpha jB - \alpha\text{cost}(T_{OPT}) \geq \alpha d(s,t)$$

Let T_α be the α-approximate solution for $I_{K\text{-ST}}$ with $K = j$. Therefore, it holds that $\alpha\text{cost}(T_{OPT}) \geq \text{cost}(T_\alpha)$ and we get

$$2\alpha jB - \text{cost}(T_\alpha) \geq \alpha d(s,t) \tag{6.2}$$

Algorithm 7 computes an α-approximate solution for $I_{K\text{-ST}}$, for each possible value of j in an increasing order, until inequality 6.2 is satisfied. By inequality 6.2, we can conclude that using tree T_α, we can collect $\alpha d(s,t)$ units of energy at the source for the instance $2\alpha I_{\text{CDU}}$, using j agents. This implies we have 2α-resource augmented solution for CDU.

□

6.2 (Limited Capacity) Energy Sharing

In this section, we consider the limited capacity version of the problem. By limited capacity, we mean that any agent cannot have more than B units of energy at any moment of the execution. This version of the problem seems to be harder to solve. In fact, we show this problem is NP-hard for small constant values of B. In this section we assume $B = 2$, which is the smallest non-trivial constant.

In the general version of the problem described below, the position of the agents is given by an adversary.

Definition 38 (CollaborativeDelivery *with Fixed Placement*)**.** CDX
Instance: $\langle G, s, t, k, h\rangle$, where $G = (V,E)$ is a simple undirected graph, $s,t \in V$

are, respectively, the source and target nodes, $h : \{1, \ldots, k\} \to V$ is the placement function that specifies the initial positions of the $k \geq 1$ agents.
Question: Does there exist a solution strategy for moving the package from s to t, when each agent starts with $B = 2$ units of energy?

If the placement of agents among homebase nodes can be chosen by the algorithm, then we have the following version of the problem:

Definition 39 (CollaborativeDelivery *with Chosen Placement*)**.** CDC
Instance: $\langle G, s, t, k, H\rangle$, where $G = (V, E)$ is a simple undirected graph, $s, t \in V$ are, respectively, the source and target nodes, $k \geq 1$ is the number of mobile agents and $H \subset V$ is the set of homebase nodes.
Question: Does there exist a placement function $p : \{1, \ldots, k\} \to H$ and a corresponding solution strategy for moving the package from s to t, when the i-th agent starts at node $p(i)$ with $B = 2$ units of energy?

A path with nodes $v_1, \ldots, v_n$ and edges $\{v_i, v_{i+1}\}$, $i \in \{1, \ldots, n-1\}$, is denoted by $(v_1, \ldots, v_n)$. The path graph with n vertices and $n-1$ edges is denoted by P_n. The length of the shortest path between two nodes u, v of a graph G is denoted by $d_G(u, v)$, or simply $d(u, v)$ when there is no potential for confusion. If an agent is initially placed on a node v, then we write $a(v)$ to refer to this agent at any point of a strategy.

6.2.1 Collaborative Delivery with Fixed Placement

We prove that CDX is NP-complete. We start by recalling the following *Bounded* 3-SAT problem, which is known to be NP-complete [Tov84].

Definition 40 (*Bounded* 3-SAT)**.**
Instance: A formula $C = C_1 \wedge \cdots \wedge C_m$ with l variables denoted $x_1, \ldots, x_l$, where $C_i = l_{1,i} \vee l_{2,i} \vee l_{3,i}$ and $l_{j,i}$ is either a variable or the negation of a variable, $i \in \{1, \ldots, m\}$, $j \in \{1, 2, 3\}$. Additionally, the number of occurrences of each variable is bounded by four.
Question: Does there exist a Boolean assignment to the variables that satisfies C?

We divide our NP-completeness proof into two parts. In the first part, we prove that the problem of deciding whether there exists a *restricted* strategy for a given CDX instance is NP-complete, where a strategy is said to be restricted if, during the entire strategy, each edge is traversed by at most one agent. We call this restricted problem RCDX and we prove its NP-completeness by reduction from *Bounded* 3-SAT in Subsection 6.2.1.1. In the second part of the proof, we argue that the edges can be replaced by a gadget that enforces the property of being traversed by at most one agent, thus reducing RCDX to CDX. This gadget is described in Subsection 6.2.1.2.

6.2.1.1 The restricted delivery problem

Assume in the following that an instance of *Bounded* 3-SAT is given: $C = C_1 \wedge \cdots \wedge C_m$ on variables $x_1, \ldots, x_l$. The maximum number of occurrences of a variable in C is 4.

Let $i \in \{1, \ldots, l\}$. For the variable x_i we define a *variable component* denoted by $G_i = (V(G_i), E(G_i))$, where

$$V(G_i) = \{s_i, t_i, t'_i\} \cup \left\{ t'_{j,i}, t_{j,i}, f'_{j,i}, f_{j,i} \,\middle|\, j \in \{1,2,3,4\} \right\},$$

and the edges of G_i are placed in such a way that $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, t_i)$ and $(s_i, t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}, t_i)$ are paths, $\{t_i, t'_i\} \in E(G_i)$, the node $t_{j,i}$ is adjacent to $t'_{j,i}$ and $f_{j,i}$ is adjacent to $f'_{j,i}$ for each $j \in \{1,2,3,4\}$. The node s_i is called a *source-terminal*, the node t'_i is a *terminal* and $t'_{j,i}, f'_{j,i}$ are called *left-* and *right-terminals*, respectively, $j \in \{1,2,3,4\}$; whenever the distinction is not necessary, all above-mentioned nodes are simply called *terminals*. Figure 6.1 depicts the graph G_i.

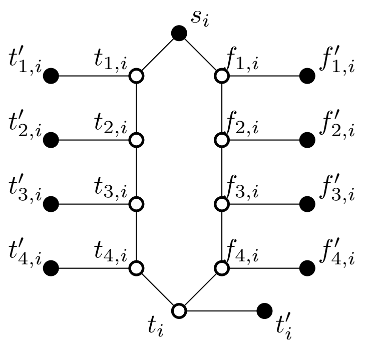

Figure 6.1: The variable component G_i. Dark nodes are the terminals (initial positions of agents).

Having constructed the variable components, we now define the graph $G_C = (V(G_C), E(G_C))$ that is the input to the RCDX problem. Let

$$V(G_C) = \bigcup_{i=1}^{l} V(G_i) \cup \left\{ c_i, c'_i \,\middle|\, ri \in \{1, \ldots, m\} \right\} \cup \{v_1, \ldots, v_{l+m+1}\}$$

and

$$\begin{aligned} E(G_C) \quad = \quad & U \cup \bigcup_{i=1}^{l} E(G_i) \cup \left\{ \{c_j, v_{l+j}\}, \{c_j, c'_j\} \,\middle|\, j \in \{1, \ldots, m\} \right\} \\ & \cup \left\{ \{v_j, v_{j+1}\} \,\middle|\, j \in \{1, \ldots, l+m\} \right\} \cup \left\{ \{t_i, v_i\} \,\middle|\, i \in \{1, \ldots, l\} \right\}, \end{aligned}$$

where U consists of the edges placed between left-terminals and right-terminals of variable components and nodes $c_1, \ldots, c_m$ as follows: (i) if C_i contains a variable x_j (a negation of x_j, respectively), then an edge between some left-terminal (right-terminal, respectively) of G_i and c_i is added to G; (ii) these edges are added in such a way that the degree of each left-terminal and right-terminal in G_C is at most two. The source of G_C is v_1 and the target of G_C is v_{l+m+1}. Figure 6.2 illustrates an example of the graph G_C.

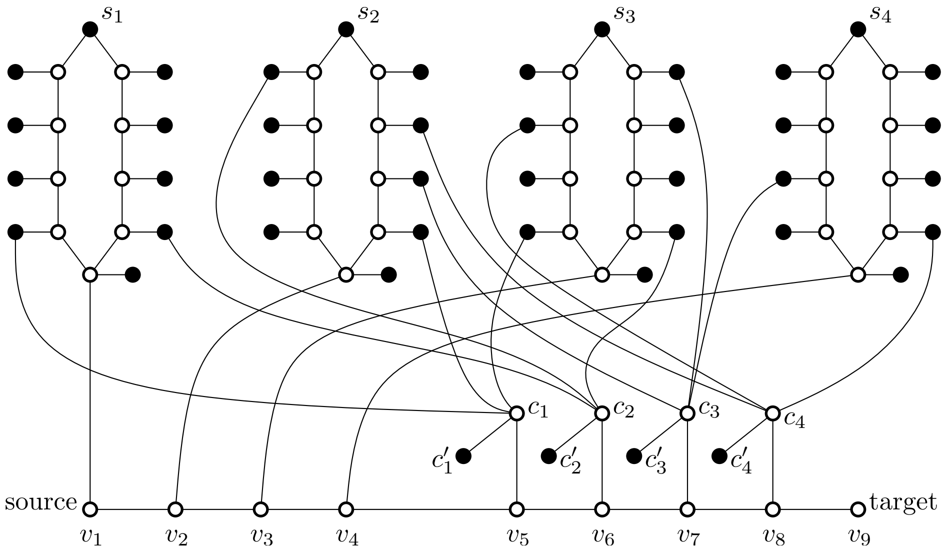

Figure 6.2: The graph G_C constructed for $C = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$ with variables x_1, x_2, x_3, x_4. Here, $l = 4$ and $m = 4$.

When describing restricted strategies we will assume the following: if two agents are located at the same node, each of them having one unit of energy, and a restricted strategy dictates that one of them performs a subsequent move, then without explicitly stating it, the energy sharing takes place so that the moving agent holds two energy units.

Lemma 41. *If C is satisfiable, then G_C admits a restricted strategy.*

Proof. Based on the Boolean assignment that satisfies C, we construct a restricted strategy for G_C as follows:

1. For each $i \in \{1, \dots, l\}$:
   a) if $x_i = \texttt{true}$, then for each $j \in \{1, 2, 3, 4\}$, $a(t'_{j,i})$ moves to its neighbor $c_j \in \{c_1, \dots, c_m\}$, if such a neighbor exists, and $a(f'_{j,i})$ moves to $f_{j,i}$,
   b) if $x_i = \texttt{false}$, then for each $j \in \{1, 2, 3, 4\}$, $a(f'_{j,i})$ moves to its neighbor $c_j \in \{c_1, \dots, c_m\}$, if such a neighbor exists, and $a(t'_{j,i})$ moves to $t_{j,i}$,
   c) $a(t'_i)$ moves to t_i.
2. For each $i \in \{1, \dots, l\}$, if $x_i = \texttt{true}$, then $a(s_i)$ follows the path $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, t_i, v_i)$ and if $x_i = \texttt{false}$, then $a(s_i)$ follows the path $(s_i, t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}, t_i, v_i)$.

3. For each $j \in \{1, \ldots, m\}$, the agent $a(c'_j)$ moves to c_j and then to v_{l+j}.

4. For each $i \in \{1, \ldots, l+m\}$ (with increasing value of i), the agent present at v_i moves from v_i to v_{i+1}.

We will now prove that the above restricted strategy is feasible. Clearly, step 1 is feasible due to the construction of the graph G_C. The correctness of step 2 follows from step 1: the agent $a(s_i)$ follows its path (either $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, t_i, v_i)$ or $(s_i, t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}, t_i, v_i)$) encountering at each node, except for the last node of the path, an agent (either $a(f'_{j,i})$ or $a(t'_{j,i})$, respectively) with one unit of energy and hence $a(s_i)$ by using this energy can reach v_i having there one unit of energy left. This also proves that $a(s_i)$ is able to perform its move from v_i to v_{i+1} in step 4. It remains to argue that for each $j \in \{1, \ldots, m\}$ there is an agent having at least one unit of energy on c_j at the end of step 2, which will prove that step 3 is valid. Since $C = C_1 \wedge \cdots \wedge C_m$ is satisfied by the chosen Boolean assignment, each C_j has a true literal. If this literal is x_i for some $i \in \{1, \ldots, l\}$, then $x_i = \texttt{true}$ and according to the construction of G_C there exists $j' \in \{1, 2, 3, 4\}$ such that $\{t'_{j',i}, c_j\} \in E(G)$ and $a(t_{j',i})$ moves from $t_{j',i}$ to c_j in step 1. An analogous argument holds when the above literal is false, i.e., $x_i = \texttt{false}$. □

We will now prove, in a series of lemmas, that the existence of a feasible restricted strategy for G_C implies that C is satisfiable.

Lemma 42. *If there exists a feasible restricted strategy for G_C, then there exists a feasible restricted strategy for G_C in which no agent with no data and one unit of energy performs a move.*

Proof. The lemma follows from a straightforward observation that such a move, since it ends with the agent having no energy, can be eliminated from a restricted strategy without affecting the movements of other agents. □

Lemma 43. *In a feasible restricted strategy for G_C, the path along which the data moves from v_1 to v_{l+m+1} is $(v_1, v_2, \ldots, v_{l+m+1})$.*

Proof. Let $\mathcal{S}$ be a feasible restricted strategy for G_C. We prove the lemma by induction on $i \in \{0, 1, \ldots, l+m\}$ that the prefix of length i of the path along which the data moves is $P_i = (v_1, \ldots, v_{i+1})$ and the agent that brings the data to v_{i+1} arrives at this node having no energy. The claim is trivial for $i = 0$ and thus suppose that $i > 0$. Suppose first that $i \leq l-1$ and we argue that the claim holds for $i+1$. Suppose for a contradiction that the claim does not hold. By induction hypothesis, the edge $\{v_{i-1}, v_i\}$ has been traversed by an agent who has no energy left when present at v_i. That implies that the data moves along the edge $\{v_i, t_i\}$. Since the edge $\{v_i, t_i\}$ can be traversed once in $\mathcal{S}$, this also implies that an agent that moves the data along $\{v_i, t_i\}$ arrives at v_i along the edge $\{v_i, v_{i+1}\}$. By construction of G_C, and more precisely by the fact that each node in the path P_{l+m} has exactly one neighbor in G_C that does not belong to this path, we obtain that, at any point in $\mathcal{S}$, there is no agent located on a node of P_{l+m} with two units of energy, which gives a contradiction. The argument is analogous for $i > l-1$. □

Lemma 44. *In each feasible restricted strategy for G_C, for each node $v \in \{v_1, \ldots, v_{l+m}\}$, an agent arrives at v having one unit of energy.*

Proof. No node $v \in X = \{v_1, \ldots, v_{l+m}\}$ is initially occupied by an agent. Thus, by Lemma 43, some agent arrives at v through the edge between v and a node in $V(G) \setminus X$, which completes the proof.

□

Lemma 45. *If there exists a feasible restricted strategy for G_C, then there exists a feasible strategy for G_C such that for each $i \in \{1, \ldots, l\}$ one of the two cases holds:*

1. *$a(s_i)$ traverses the path $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, t_i, v_i)$ and $a(f'_{j,i})$ traverses the edge $\{f'_{j,i}, f_{j,i}\}$ for each $j \in \{1,2,3,4\}$, or*
2. *$a(s_i)$ traverses the path $(s_i, t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}, t_i, v_i)$ and $a(t'_{j,i})$ traverses the edge $\{t'_{j,i}, t_{j,i}\}$ for each $j \in \{1,2,3,4\}$.*

Proof. Let $\mathcal{S}$ be a feasible restricted strategy for G_C that satisfies Lemma 42. Clearly, the only possible action for $a(t'_i)$ is to move to t_i. Consider a node v that is a left-terminal or a right-terminal of G_i. By the assumption that each edge can be traversed at most once and v is of degree 2, we may assume without loss of generality that no agent other than $a(v)$ is present at v at any time point in $\mathcal{S}$. This implies that $a(v)$ either stays idle or moves to a neighbor u. By construction of G_C, $u \in \{c_1, \ldots, c_m\}$ or $u \in V(G_i)$; more precisely, in the latter case $u \in \{f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}\}$ if v is a right-terminal and $u \in \{t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}\}$ if v is a left-terminal. By Lemmas 42 and 43, $a(v)$ performs no other move until meeting another agent. Note that by Lemma 44, the edge $\{t_i, v_i\}$ is traversed by an agent moving from t_i to v_i. Thus, by a simple induction on the prefix length of the path $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, v'_i, v_i)$ or $(s_i, t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}, v'_i, v_i)$ one can prove that this traversal may only happen if $a(s_i)$ traverses exactly one of these paths. In the former case all agents initially present at right-terminals have moved to the nodes $f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}$ and in the latter case all agents initially present at right-terminals have moved to nodes $t_{1,i}, t_{2,i}, t_{3,i}, t_{4,i}$, as required by the lemma. □

Lemma 46. *If there exists a feasible restricted strategy for G_C, then C is satisfiable.*

Proof. Consider a feasible restricted strategy $\mathcal{S}$ for G_C. According to Lemma 45, for each $i \in \{1, \ldots, l\}$, we have two possible cases for the agent $a(s_i)$: it either follows the path given in Lemma 45(1) in which case we set x_i to be `true`, or the agent follows the path from Lemma 45(2) in which case we set x_i to be `false`. We now argue that $C = C_1 \wedge \cdots \wedge C_m$ is satisfied under this Boolean assignment. Let $j \in \{1, \ldots, m\}$ be selected arbitrarily and we need to prove that C_j contains a literal that has the value `true`. By Lemma 44 and by the fact that c_j is initially not occupied by an agent, two agents arrive at c_j during the execution of $\mathcal{S}$. One of these two agents must be initially located at a node $t'_{j',i}$ or $f'_{j',i}$ for some $i \in \{1,\ldots,l\}$ and $j' \in \{1,2,3,4\}$. Since both cases are analogous we consider the first one, i.e., $a(t'_{j',i})$ arrives at c_j. Again by Lemma 45, $a(s_i)$ traverses the path $(s_i, f_{1,i}, f_{2,i}, f_{3,i}, f_{4,i}, t_i)$ and according to the definition, $x_i = \texttt{true}$. Since $\{t'_{j',i}, s_i\} \in E(G)$, by construction of G_C we have that C_j contains x_j as a literal, which implies that C_j is satisfied. □

We now conclude our NP-completeness results for two classes of graphs: bounded degree graphs and bounded diameter graphs.

Theorem 47. RCDX *is* NP-*complete in the class of graphs with degree bouded by* 5 *and with each node being initially occupied by at most one agent.*

Note that one may extend the graph G_C in such a way that a path with two edges is added between each pair of nodes in $X = \{v_1, \ldots, v_l, c_1, \ldots, c_m\}$. The newly introduced nodes hold initially no agents. This allows us to use Lemma 44 to conclude that the newly added edges cannot be traversed by any agent in any feasible restricted strategy. On the other hand, this extension turns G_C into a bounded diameter graph, which allows us to state the following.

Theorem 48. RCDX *is* NP-*complete in the class of graphs with diameter at most* 14 *with each node being initially occupied by at most one agent.*

6.2.1.2 The unrestricted delivery problem

We now introduce a gadget that we will use to modify the graph G_C constructed in Section 6.2.1.1. Replacing the edges of G_C with this gadget will simulate the "one-time traversal" property enforced in the RCDX problem. The gadget, denoted as $\hat{G} = (V(\hat{G}), E(\hat{G}))$, is a path on four nodes (z'_1, z_1, z_2, z'_2). Then, given a graph G_C constructed on the basis of a Boolean formula C, we obtain a graph $\hat{G}_C$ by replacing each edge $\{u,v\}$ of G_C by a copy of $\hat{G}$ and adding edges $\{u, z_1\}$ and $\{v, z_2\}$ to $\hat{G}_C$. Additionally, the nodes z'_1 and z'_2 of each gadget in $\hat{G}_C$ are initially occupied by one agent. We make two observations regarding $\hat{G}_C$.

Lemma 49. *Consider any two nodes u, v in $\hat{G}_C$ such that $\{u,v\} \in E(G_C)$. At most one agent, having two energy units when present at u can reach v in a feasible strategy for $\hat{G}_C$, provided that no more than four units of energy are present at any node of $\hat{G}_C$ in total during the entire strategy for $\hat{G}_C$.*

Proof. Clearly, each agent initially placed on a node of any gadget $\hat{G}$ in $\hat{G}_C$, in any feasible strategy, moves from z'_j to z_j, $j \in \{1,2\}$. Then, an agent that goes from u to v in $\hat{G}_C$ is able to traverse the path (u, z_1, z_2, v) with the use of the energy of agents $a(z'_1)$ and $a(z'_2)$. This observation, together with the assumption that in total at most 4 units of energy are present at u, allows us to say that another traversal of this path in $\hat{G}_C$ is not possible. Indeed, an agent with two units of energy is not able to traverse the path and, by a simple case analysis, two agents having four units of energy at u cannot reach v together. □

Lemma 50. *At any point of a feasible strategy for $\hat{G}_C$, at most 4 units of energy are present at any node of $\hat{G}_C$.*

Proof. The claim follows by a simple induction on the number of steps of a strategy for $\hat{G}_C$. Indeed, with the use of Lemma 49 and the fact that the degree of $\hat{G}_C$ is bounded by 5, we can conclude that at most four units of energy are present in total at any node of $\hat{G}_C$. □

Lemmas 49 and 50 give the following theorems.

Theorem 51. CDX *is* NP*-complete in the class of graphs with degree bounded by* 5 *and with each node being initially occupied by at most one agent.*

Theorem 52. CDX *is* NP*-complete in the class of graphs with diameter at most* 42 *with each node being initially occupied by at most one agent.*

6.2.2 Collaborative Delivery with Chosen Placement

Let $I = \langle G, s, t, H \rangle$ be an instance of CDC, with $G = (V, E)$ and $H \subseteq V$. Recall that a solution to I is a strategy that enables a group of energy-sharing agents starting from some or all of the nodes in H with battery capacity $B = 2$ to transfer the package from s to t. The cost of a solution is the total initial energy of the agents that are placed on nodes in H. If $u \in V$, we denote by h_u the node in H that is closest to u, breaking ties arbitrarily, and we denote by $z(u)$ the distance from u to h_u in G, i.e., $z(u) = d_G(h_u, u)$. Let $\hat{G}$ be a weighted complete digraph with vertex set V and arc set E', and let the weight of an arc $e = (u, v) \in E'$ be $w(e) = 2^{z(u)+d_G(u,v)-1}$. If P is a directed path in $\hat{G}$, the *weight of* P is the sum of the weights of the arcs in P. This section is devoted to the proof of the following theorem:

Theorem 53. *An optimal solution to $I = \langle G, s, t, H \rangle$ has cost $d_{\hat{G}}(s, t)$.*

It suffices to show that there exists a directed s-t path in $\hat{G}$ with weight smaller than or equal to the cost of the optimal solution for I and, additionally, that every directed s-t path in $\hat{G}$ corresponds to some solution for I with cost equal to the weight of the path. The latter claim is given in the following lemma:

Lemma 54. *For every directed s-t path in $\hat{G}$, there exists a solution for I with cost equal to the weight of the path.*

Proof. For every arc (u, v) of the path, we send agents from h_u to v in G along the path $h_u \rightsquigarrow u \rightsquigarrow v$, where $h_u \rightsquigarrow u$ is a shortest path from h_u to u and $u \rightsquigarrow v$ is a shortest path from u to v. The agents pick up the package from u and release it at v. The total length is $z(u) + d_G(u, v)$, and therefore $2^{z(u)+d_G(u,v)-1}$ units of energy suffice. Indeed, if, at any given point of the $h_u \rightsquigarrow u \rightsquigarrow v$ path, we have $2k$

units of energy, we group these into k agents and we make them traverse the edge leading to the next node. This wastes k units of energy and thus k units arrive at the next node. This continues up to the penultimate node, where 1 unit of energy arrives after $z(u) + d_G(u, v) - 1$ edge traversals. The last remaining agent uses this unit of energy to reach v with the package.

□

In the rest of this section, we derive some structural properties of optimal solutions for I, which permit us to prove the former claim (cf. Lemma 67). In Section 6.2.2.1 we introduce our main tool in the analysis: an energy flow hypergraph that provides a way of presenting solutions to CDC. Then, Sections 6.2.2.2 and 6.2.2.3 give a series of properties of this hypergraph, allowing us to finish the proof of Theorem 53 in Section 6.2.2.4. We assume that $s \neq t$, otherwise Theorem 53 holds trivially.

6.2.2.1 The energy flow hypergraph

Given a solution $\mathcal{S}$ for I with cost $X > 0$, we represent $\mathcal{S}$ by a triple $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$, where $(\mathcal{V}, \mathcal{E}) = \mathcal{H}$ is a directed hypergraph that represents the flow and eventual consumption of energy units (cf. Definition 55 below) and $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ corresponds to the package moves under $\mathcal{S}$ (cf. Definition 56 below). The nodes of $\mathcal{H}$ are the energy arrival and extinction events of $\mathcal{S}$, as specified below.

We assume that the units of energy that are initially present at nodes in H receive distinct identities from 1 to X. We distinguish two types of *events* during the delivery under $\mathcal{S}$. An *arrival* event occurs whenever an agent with two units of energy i, j with $i < j$ moves from some node u of G to a neighbor v at time step t. We say that the unit of energy j is *wasted by* i during the event and that the unit of energy i *arrives* at v at time $t + 1$. We denote this event as $(i, t + 1, v)$. An *extinction* event occurs whenever an agent with one unit of energy i moves from some node u of G to a neighbor v at time step t. We say that the unit of energy i *wastes itself* during the event. We denote this event as $(\perp_i, t + 1, v)$.

We also consider as *arrival* events the appearance of the X units of energy at the homebases at time 0, and we denote them as $(i, 0, h_i)$, for $1 \leq i \leq X$, where $h_i \in H$ is the homebase where energy unit i was placed.

We are now ready to define the *energy flow hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ in terms of the arrival and extinction events as follows:

Definition 55 (Energy flow hypergraph)**.** The vertex set $\mathcal{V}$ contains all of the arrival and extinction events, as specified above. The hyperarc set $\mathcal{E}$ contains $(\{(i, t_1, u), (j, t_2, u)\}, \{(i, t, v)\})$ if $t \geq 1$, the unit of energy i came from node u during the event (i, t, v), $j > i$ is the unit of energy consumed during the event (i, t, v), and in addition $t_1 < t$, $t_2 < t$, and i (resp. j) is not involved in any other events between times t_1 (resp. t_2) and t. Furthermore, $\mathcal{E}$ contains $(\{(i, t_1, u)\}, \{(\perp_i, t, v)\})$ if $t_1 < t$ and i is not involved in any other events between times t_1 and t.

Definition 56 (Item moves)**.** The set $\tilde{\mathcal{E}}$ is defined as the subset of $\mathcal{E}$ that contains all of the hyperarcs that correspond to package moves. More precisely, a hyperarc $(\{(i, t_1, u), (j, t_2, v)\}, \{(i, t, v)\}) \in \tilde{\mathcal{E}}$ if the agent that arrived with energy unit i at v at time t was carrying the package. Similarly, a hyperarc $(\{(i, t_1, u)\}, \{(\perp_i, t, v)\}) \in \tilde{\mathcal{E}}$ if the agent that arrived with zero energy at v at time t (having wasted energy unit i) was carrying the package.

We illustrate the energy flow hypergraph $\mathcal{H}$ corresponding to a simple solution for an CDC instance in Figure 6.3. Note that the cost of $\mathcal{S}$ is given by the total number of energy units that "arrive" at nodes in H at time 0, which corresponds to the number of nodes of the form $(\,\cdot\,, 0, \,\cdot\,)$ in $\mathcal{H}$.

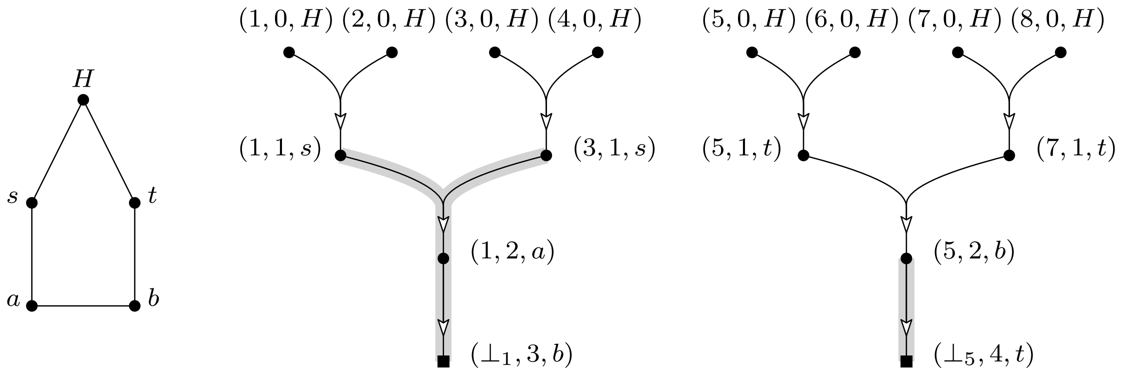

Figure 6.3: An energy flow hypergraph constructed for a graph G shown on the left. The hypergraph consists of two components (the hyperarcs that correspond to package moves are highlighted): the first component dictates two agents to move from H to s and then one of those agents picks up the package and travels along path (s, a, b); the second component makes two agents to move from H to t and then one of them goes to b, picks up the package and returns to t.

By construction, there is no cycle in $\mathcal{H}$. Moreover, the head of every hyperarc of $\mathcal{H}$ has size 1 and every node of $\mathcal{H}$ is contained in at most one hyperarc head and in at most one hyperarc tail. Therefore, $\mathcal{H}$ consists of a number of independent components $\mathcal{H}_1, \ldots, \mathcal{H}_\sigma$, each of which has a tree-like structure, as in Figure 6.3.

Notation If $e \in \mathcal{E}$, we denote by $\mathsf{head}(e)$ the unique node that is in the head of e. If $v \in \mathcal{V}$, we denote by $\mathfrak{g}(v)$ the node of G that is involved in the event v. If $e \in \mathcal{E}$, then we denote by $\mathfrak{g}_{\mathsf{tail}}(e)$ the node of G that is involved in the events in the tail of e (recall that, by definition of the hypergraph $\mathcal{H}$, all events in the tail of e must involve the same node of G), and by $\mathfrak{g}_{\mathsf{head}}(e)$ the node of G that is involved in the unique event in the head of e (i.e., $\mathfrak{g}_{\mathsf{head}}(e) = \mathfrak{g}(\mathsf{head}(e))$).

If $v \in \mathcal{V}$, let Δv denote the subgraph of $\mathcal{H}$ induced by the ancestors of v and v itself. Let $\mathsf{height}(v)$ denote the number of hyperarcs in the longest path that terminates at v. If $e \in \mathcal{E}$, we abuse the notation slightly and we denote by $\mathsf{height}(e)$ the height of $\mathsf{head}(e)$. If $v, v' \in \mathcal{V}$, we write $v \prec v'$ if v is an ancestor of v' and we write $v \sqsubset v'$ if v precedes v' temporally, i.e., $v = (i, t, x)$ and $v' = (i', t', x')$ with $t < t'$. Note that $v \prec v'$ implies $v \sqsubset v'$. As above, we extend the notation to arcs and we write $e \prec e'$ if $\mathsf{head}(e) \prec \mathsf{head}(e')$ and $e \sqsubset e'$ if $\mathsf{head}(e) \sqsubset \mathsf{head}(e')$.

If Δv contains x nodes of the form $(\,\cdot\,, 0,\,\cdot\,)$, then we say that Δv *incurs a cost of* x. This represents the energy units used by the solution in order to generate the event v. We also say that a component $\mathcal{H}_i$ incurs a cost equal to the cost incurred by its maximal node (under $\prec$). The cost of $\mathcal{S}$ is the sum of the costs incurred by the components of $\mathcal{H}$.

6.2.2.2 Properties of optimal solutions

The goal of this section is to prove a property of optimal solutions that can be informally stated as follows: every component of the hypergraph corresponding to the solution contains exactly one chain of item moves and the last hyperarc of this chain is an extinction event of the component.

Proposition 57. *For every solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ in which there exist arcs $f, g \in \tilde{\mathcal{E}}$ with $f \prec g$, there exists a solution $\mathcal{S}' = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}}')$ with $\tilde{\mathcal{E}}' = \tilde{\mathcal{E}} \setminus \{e : f \sqsubset e \sqsubset g\} \cup \{e : f \prec e \prec g\}$.*

Proof. Under $\mathcal{S}$, the arcs in $\tilde{\mathcal{E}} \cap \{e : f \sqsubset e \sqsubset g\}$ are responsible for taking the package from node $\mathfrak{g}_{\mathsf{head}}(f)$ to node $\mathfrak{g}_{\mathsf{tail}}(g)$ in G. Under $\mathcal{S}'$, we achieve the same result by using the hyperarcs $\{e : f \prec e \prec g\}$.

□

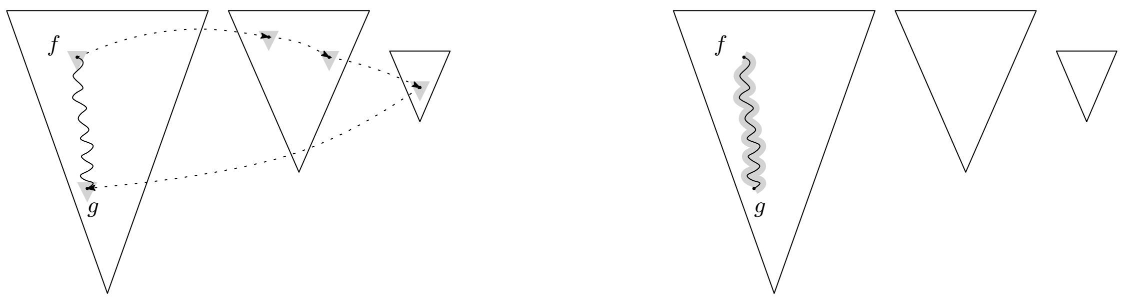

Figure 6.4: Illustration of Proposition 57. Triangles represent components of the solution hypergraph. The solution $\mathcal{S}$ is shown on the left (in which some arcs of the path from f to g are not package moves — those package moves can be possibly in different components; the dotted arrows represent the time succession between package moves that take the package from f to g) and the corresponding $\mathcal{S}'$ is shown on the right.

By repeated application of Proposition 57, we obtain the following:

Corollary 58. *For every solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$, there exists a solution $\mathcal{S}' = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}}')$ and a partition of $\tilde{\mathcal{E}}'$ into sets $\tilde{\mathcal{E}}'_1, \ldots, \tilde{\mathcal{E}}'_\tau$ such that, for every i, the hyperarcs of $\tilde{\mathcal{E}}'_i$ form a chain in $\mathcal{H}$ and, for every $e \in \tilde{\mathcal{E}}'_i$ and $e' \in \tilde{\mathcal{E}}'_j$ with $i < j$, we have $e \sqsubset e'$, $e \not\prec e'$, and $e' \not\prec e$.*

Note that Proposition 57 and Corollary 58 apply to *any* solution (not necessarily an optimal one). Furthermore, in both statements, the obtained solution $\mathcal{S}'$ has the same energy flow hypergraph as $\mathcal{S}$, and therefore it has the same cost as $\mathcal{S}$.

Lemma 59. *For every optimal solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ and for every component $\mathcal{H}_i$ of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with maximum (under $\prec$) hyperarc r_i, we have $r_i \in \tilde{\mathcal{E}}$ and $\mathsf{head}(r_i)$ is an extinction event.*

Proof. For a contradiction, suppose that there exists an optimal solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ that has a component $\mathcal{H}_i$ with maximum (under $\prec$) hyperarc r_i such that $r_i \notin \tilde{\mathcal{E}}$ or $\mathsf{head}(r_i)$ is not an extinction event. Among those optimal solutions, we choose a solution $\mathcal{S}$ with smallest $|\mathcal{E}|$.

If $r_i \notin \tilde{\mathcal{E}}$, then consider the solution $\mathcal{S}' = (\mathcal{V}', \mathcal{E}', \tilde{\mathcal{E}}')$, where $\mathcal{H}' = (\mathcal{V}', \mathcal{E}')$ is the subgraph of $\mathcal{H}$ induced by the set of hyperarcs $\mathcal{E} \setminus \{r_i\}$, and $\tilde{\mathcal{E}}' = \tilde{\mathcal{E}}$. By definition of $\mathcal{H}'$, the nodes that used to be in $\mathsf{tail}(r_i)$ are clearly not extinction events (otherwise they would not be in the tail of a hyperarc) and, if they exist in $\mathcal{H}'$, they are now maximum (under $\prec$) in their respective components in $\mathcal{H}'$. Note that $\mathcal{S}'$ is optimal, its energy flow hypergraph has one less hyperarc than that of $\mathcal{S}$, and it contains at least one component whose maximum node (under $\prec$) is not an extinction event. Therefore, it contradicts our choice of $\mathcal{S}$. On the other hand, if at least one of the nodes from $\mathsf{tail}(r_i)$ does not exist in $\mathcal{H}'$, then this means that that node did not have incoming hyperarcs in $\mathcal{S}$, therefore it was of the form $(\,\cdot\,, 0, \,\cdot\,)$ and, since it does not exist in $\mathcal{S}'$, the cost of $\mathcal{S}'$ is strictly smaller than the cost of $\mathcal{S}$. This contradicts the optimality of $\mathcal{S}$.

If $r_i \in \tilde{\mathcal{E}}$ but $\mathsf{head}(r_i)$ is not an extinction event, then let $\mathcal{S}' = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}}')$ be the solution obtained from $\mathcal{S}$ by applying Corollary 58. Note that $\mathcal{S}'$ has the same energy flow hypergraph $\mathcal{H}$ as $\mathcal{S}$, but a possibly different set of package moves $\tilde{\mathcal{E}}'$. In particular, the cost of $\mathcal{S}'$ is the same as the cost of $\mathcal{S}$. Let u be a node in the tail of r_i such that Δu does not contain any hyperarc in $\tilde{\mathcal{E}}'$ (such a node must exist by Corollary 58). Let u' be the other node in the tail of r_i. We construct a new solution $\mathcal{S}'' = (\mathcal{V}'', \mathcal{E}'', \tilde{\mathcal{E}}'')$ by eliminating Δu from the hypergraph and by replacing the hyperarc r_i with a new hyperarc $r''_i = (\{u'\}, \{v''\})$ that leads to an extinction event v'' with $\mathfrak{g}(v'') = \mathfrak{g}_{\mathsf{head}}(r_i)$ and which occurs at the same time step as $\mathsf{head}(r_i)$. We include r''_i in $\tilde{\mathcal{E}}''$ if and only if $r_i \in \tilde{\mathcal{E}}'$. The new solution $\mathcal{S}''$ has cost strictly smaller than that of $\mathcal{S}$, therefore it contradicts the optimality of $\mathcal{S}$. □

By applying Corollary 58 to an arbitrary optimal solution, we obtain the following corollary in view of Lemma 59:

Corollary 60. *There exists an optimal solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ such that every component $\mathcal{H}_i$ of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with maximum (under $\prec$) hyperarc r_i contains exactly one chain of package moves whose last hyperarc is r_i and, in addition, $\mathsf{head}(r_i)$ is an extinction event.*

6.2.2.3 Canonical nodes

Given a solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ for I, let $v \in \mathcal{V}$ such that Δv does not contain any hyperarc in $\tilde{\mathcal{E}}$. Intuitively, if v is not an extinction event, then the sole function of Δv in the solution is to bring one unit of energy to $\mathfrak{g}(v)$. It thus makes sense that, if $\mathcal{S}$ is optimal, then the energy units that participate in the events of Δv travel along shortest paths from their respective homebases to $\mathfrak{g}(v)$. If v satisfies these conditions, then we say that v is *canonical*. The following definition captures this notion:

Definition 61 (Canonical nodes)**.** Given a solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$, a node $v \in \mathcal{V}$ is called *canonical* if either $\mathsf{height}(v) = 0$, or $\mathsf{height}(v) = h + 1$ for some $h \geq 0$ (in this case, $v = \mathsf{head}(e)$ for some $e \in \mathcal{E}$) and all of the following hold: (i) for every node u in the tail of e, u is canonical and $\mathsf{height}(u) = h$, (ii) $e \notin \tilde{\mathcal{E}}$, and (iii) $z(\mathfrak{g}(v)) = 1 + z(\mathfrak{g}_{\mathsf{tail}}(e))$.

The two propositions below follow easily by induction on the height of v. Recall that, by definition of $\mathcal{H}$, if $\mathsf{height}(v) = 0$ then $\mathfrak{g}(v) \in H$.

Proposition 62. *If v is canonical, then $z(\mathfrak{g}(v)) = \mathsf{height}(v)$.*

Proof. If $\mathsf{height}(v) = 0$, then $\mathfrak{g}(v) \in H$. Therefore, $h_{\mathfrak{g}(v)} = \mathfrak{g}(v)$ and $z(\mathfrak{g}(v)) = 0$.

If $\mathsf{height}(v) = h+1$ and $v = \mathsf{head}(e)$, then let u be a node in the tail of e. Because v is canonical and by the inductive hypothesis, u is canonical and $z(\mathfrak{g}(u)) = h$. Therefore, $z(\mathfrak{g}(v)) = 1 + z(\mathfrak{g}(u)) = h + 1$.

□

Proposition 63. *If v is canonical and it is not an extinction event, then the cost incurred by Δv is $2^{\mathsf{height}(v)}$.*

We can now prove that, in every optimal solution, every node v whose Δv does not contain any package move is canonical.

Lemma 64. *For every optimal solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ and for every $v \in \mathcal{V}$, v is canonical or Δv contains a hyperarc in $\tilde{\mathcal{E}}$.*

Proof. Fix an optimal solution $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$. We prove the claim by induction on $\mathsf{height}(v)$. If $\mathsf{height}(v) = 0$, then v is canonical by definition. We assume that the claim holds for all nodes with height at most $h \geq 0$. Let $v \in \mathcal{V}$ be a node with $\mathsf{height}(v) = h + 1$ and let $e \in \mathcal{E}$ be the hyperarc such that $v = \mathsf{head}(e)$.

If v is an extinction event, then e is maximal under $\prec$. By Lemma 59, we have $e \in \tilde{\mathcal{E}}$ and therefore Δv contains a hyperarc in $\tilde{\mathcal{E}}$.

If v is not an extinction event, then let $u_1, u_2 \in \mathcal{V}$ be the nodes in the tail of e and assume without loss of generality that $\mathsf{height}(u_1) = h$ and $\mathsf{height}(u_2) \leq h$. We can assume that Δv does not contain any hyperarc in $\mathcal{E}$, otherwise the claim is proved. It follows that Δu_1 and Δu_2 do not contain any hyperarc in $\mathcal{E}$, therefore u_1 and u_2 are both canonical by the inductive hypothesis. By Proposition 62 and the fact that $\mathfrak{g}(u_1) = \mathfrak{g}(u_2)$, we have $\mathsf{height}(u_1) = \mathsf{height}(u_2) = h$. In view of these observations, the only reason why v would not be canonical is if $z(\mathfrak{g}(v)) \neq 1 + z(\mathfrak{g}(u_1)) = 1 + h$. However, note that we know $z(\mathfrak{g}(v)) \leq 1 + h$, because $\mathsf{height}(v) = h + 1$ and therefore there exists a homebase at distance at most $h + 1$ from $\mathfrak{g}(v)$. On the other hand, u_1 and u_2 are both canonical with height h, which implies in view of Proposition 63 that the cost incurred by Δv is $2^h + 2^h = 2^{h+1}$. If $z(\mathfrak{g}(v)) < h + 1$, then we can construct a new solution in which we replace Δv by a subgraph that incurs a cost strictly smaller than 2^{h+1}, which contradicts the optimality of $\mathcal{S}$. We conclude that $z(\mathfrak{g}(v)) = h + 1$ and, consequently, v is canonical.

□

6.2.2.4 Completing the proof of Theorem 53

In the following, let $\mathcal{S}^\star = (\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$ be an optimal solution as guaranteed by Corollary 60, with the maximum number σ of components of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. Let $(\mathcal{H}_i)_{i=1,\dots,\sigma}$ be an enumeration of the components of $\mathcal{H}$ in temporal order of their extinction events. For $i \in \{1, \dots, \sigma\}$, component $\mathcal{H}_i$ is responsible for moving the package along a path $P_i = (u_{i,0}, u_{i,1}, \dots, u_{i,\rho_i})$ in G, where $u_{1,0} = s$, $u_{\sigma,\rho_\sigma} = t$, and $u_{i,\rho_i} = u_{i+1,0}$ (for $i < \sigma$).

Lemma 65. *For every i, j in the ranges $1 \leq i \leq \sigma$ and $0 \leq j < \rho_i - 1$, $z(u_{i,j+1}) = 1 + z(u_{i,j})$.*

Proof. For a contradiction, let $\mathcal{H}_i$ be a component in which the claim is false and let j be the smallest index such that $z(u_{i,j+1}) \neq 1 + z(u_{i,j})$. Since $u_{i,j+1}$ and $u_{i,j}$ are neighbors in G, the closest homebase to $u_{i,j+1}$ is at a distance of at most $1 + z(u_{i,j})$, so $z(u_{i,j+1}) \leq 1 + z(u_{i,j})$. It follows, then, that $z(u_{i,j+1}) < 1 + z(u_{i,j})$.

Let $e_{i,j+1} \in \tilde{\mathcal{E}}$ be the $(j+1)$-st package move hyperarc in $\mathcal{H}_i$, that corresponds to the move of the package from $u_{i,j}$ to $u_{i,j+1}$. Let $v = \mathsf{head}(e_{i,j+1})$. Since $j < \rho_i - 1$, we have $j + 1 < \rho_i$ and therefore v is not an extinction event. Let v_1, v_2 be the nodes in the tail of $e_{i,j+1}$, of which at least one, say v_2, must be canonical by Lemma 64.

We construct a new solution $\mathcal{S}' = (\mathcal{V}', \mathcal{E}', \tilde{\mathcal{E}}')$ by modifying $\mathcal{S}^\star$ as follows: We replace hyperarc $e_{i,j+1}$ with a new hyperarc $(\{v_1\}, \{v'\})$ that leads to an extinction event with $\mathfrak{g}(v') = \mathfrak{g}(v) = u_{i,j+1}$. At the same time, we remove Δv_2 and we insert new nodes and hyperarcs so as to make v canonical. The cost of $\mathcal{S}'$ is not greater than that of $\mathcal{S}^\star$, since we removed a subtree which incurred a cost of $2^{z(u_{i,j})}$ and we added a subtree which incurs a cost of $2^{z(u_{i,j+1})} \leq 2^{z(u_{i,j})}$. Therefore, $\mathcal{S}'$ is optimal, each of its components contains exactly one chain of package moves, and it has one more component than $\mathcal{S}^\star$, which contradicts our choice of $\mathcal{S}^\star$.

□

Corollary 66. *The cost incurred by component $\mathcal{H}_i$ is $2^{z(u_{i,0})+\rho_i-1}$.*

Proof. Let $e_{i,1}, \dots, e_{i,\rho_i}$ be the package move hyperarcs in $\mathcal{H}_i$. If $\mathsf{head}(e_{i,1})$ is an extinction event, then $\rho_i = 1$ and the tail of $e_{i,1}$ contains a single canonical node v with $\mathfrak{g}(v) = u_{i,0}$. The cost of $\mathcal{H}_i$ is equal to the cost incurred by Δv, which is equal to $2^{z(u_{i,0})}$ by Propositions 62 and 63.

If $\mathsf{head}(e_{i,1})$ is not an extinction event, then $\rho_i > 1$ and the tail of $e_{i,1}$ contains two canonical nodes v_0, v_0' with $\mathfrak{g}(v_0) = \mathfrak{g}(v_0') = u_{i,0}$. Moreover, for any j in the range $1 < j < \rho_i$, the tail of $e_{i,j}$ contains exactly one canonical node v_{j-1} with $\mathfrak{g}(v_{j-1}) = u_{i,j-1}$. By Lemma 65, $z(u_{i,j-1}) = j - 1 + z(u_{i,0})$. The cost of $\mathcal{H}_i$ is equal to the total cost incurred by $\Delta v_0'$ and $\Delta v_0, \Delta v_1, \dots, \Delta v_{\rho_i-2}$. By Propositions 62 and 63, this sum is equal to

$$2^{z(u_{i,0})} + \sum_{j=0}^{\rho_i-2} 2^{z(u_{i,j})} = 2^{z(u_{i,0})} + \sum_{j=0}^{\rho_i-2} 2^{j+z(u_{i,0})} = 2^{z(u_{i,0})+\rho_i-1}$$

□

Let Q be the directed s-t path in $\hat{G}$ that consists of the arcs $(u_{1,0}, u_{1,\rho_1})$, $(u_{2,0}, u_{2,\rho_2})$,..., $(u_{\sigma,0}, u_{\sigma,\rho_\sigma})$. By definition of $\hat{G}$, the i-th arc of Q has weight $2^{z(u_{i,0})+d_G(u_{i,0},u_{i,\rho_i})-1}$. However, P_i is a path in G from $u_{i,0}$ to u_{i,ρ_i} and its length is ρ_i. Therefore, $d_G(u_{i,0}, u_{i,\rho_i}) \leq \rho_i$. In view of Corollary 66, we conclude that the weight of the i-th arc of Q is at most equal to the cost of $\mathcal{H}_i$ and thus the total weight of the arcs of Q is at most equal to the cost of $\mathcal{S}^\star$. We have proved the following lemma, which concludes the proof of Theorem 53:

Lemma 67. *There exists a directed s-t path in $\hat{G}$ with weight at most equal to the cost of an optimal solution to I.*

6.2.3 Collaborative Delivery for $B > 2$

We conclude this section with some remarks in the case where the energy capacity of the agents is strictly greater than 2.

While we expect that our NP-completeness reduction generalizes to $B \geq 3$, the situation is less clear when it comes to the question of computing the energy allocation to the homebases as part of the solution. A straightforward adaptation of our algorithm from Section 6.2.2 would be to reduce the problem to computing the shortest s-t path in a directed graph $\hat{G}$ similar to the one we construct in Section 6.2.2, except that the weight of an arc (u, v) would be equal to the minimum amount of energy required by agents with capacity B to traverse the path $h_u \rightsquigarrow u \rightsquigarrow v$, where h_u is the nearest homebase to u. Unfortunately, this algorithm is no longer guaranteed to produce an optimal solution for $B \geq 3$ (see Figure 6.5).

The reason is that several nice properties of the optimal solutions for $B = 2$ no longer hold for $B \geq 3$. In particular, since the hyperarcs in the energy flow

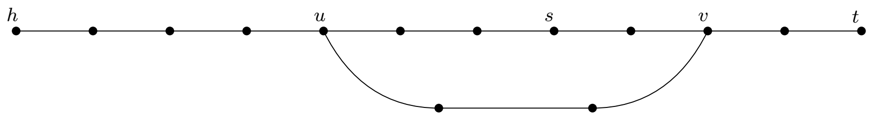

Figure 6.5: An example of a graph in which the straightforward adaptation of our algorithm from Section 6.2.2 to $B = 3$ does not give an optimal solution. Here, $H = \{h\}$ and the shortest s-t path in $\hat{G}$ is $s \to v \to t$, with each arc having a weight of 41 for a total cost of 82. However, the optimal solution has a cost of 81: 27 fully charged agents start from h and they reach u with 16 remaining energy units in total. At u, the agents split into two groups with 8 units of energy each. The first group goes to s and then to v from the top branch, picking up the package from s on the way. The second group goes to v from the bottom branch, picks up the package, and continues until t.

hypergraph can now have up to two nodes in their heads, each component can now have more than one bottommost nodes and it can contain more than one chains of package moves. This is exactly the case in the example of Figure 6.5, where the energy flow hypergraph of the optimal solution has only one component, which contains two chains of package moves. Furthermore, it is no longer the case that all of the nodes in the tail of a given hyperarc have the same height. This can be seen even in cases where the optimal solution consists of only one component with only one chain of package moves (see Figure 6.6).

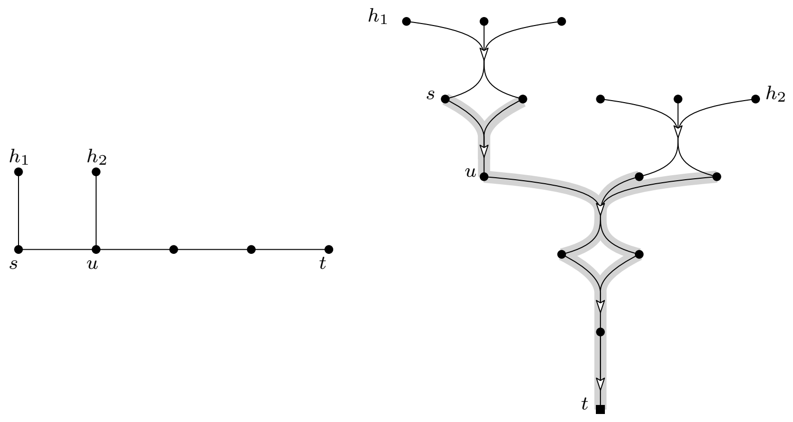

Figure 6.6: An example of a graph (*left*, with $H = \{h_1, h_2\}$) in which the energy flow hypergraph of the optimal solution (*right*) contains a hyperarc with nodes of different heights in its tail.

Conclusion

The results presented in this thesis cover a range of topics related to energy constrained mobile agents. We investigated how this natural constraint on the energy resources of the agents affect their performance capabilities in solving important tasks in mobile agent computing.

For the online graph exploration in Chapter 2 we studied the case of tree exploration with a team of agents, each of which can traverse at most B edges. We gave matching lower and upper bounds of $\Theta(\log B)$ on the competitive ratio of the cost of tree exploration for the local communication model. Unlike previous algorithms for energy constrained agents, the agents in our algorithm do not necessarily return to the root after exploration. This fact allows us to explore trees of larger depth (at least twice more compared to [Awe+99; BRS95]). However, our algorithm can be still used, e.g., to collect information from the leaves of a tree, or to search for a resource and bring it back to the root, since there is always a transfer of information from the leaves to the root.

Following on from the tree exploration with the goal of minimizing the number of agents, in Chapter 3, we studied the partial exploration of trees of arbitrary size and structure, with a fixed number of agents with fixed energy resources. Since exploring the complete tree is not always possible our objective was to to visit as many nodes as possible. We presented an algorithm that achieves a constant factor competitive ratio, particularly $1 + 4\phi < 7.473$, for the global communication model and we also gave a lower bound of $2 - o(1)$ on the competitive ratio of any algorithm.

Next, in Chapter 4, we considered the sustainable exploration of trees where the nodes contain resources that the agent can use as fuel to continue the exploration. We showed, that it is NP-hard to maximize the remaining energy at the end of the traversal. Next, we showed that if we supply the agent with a sufficiently large energy budget, we can solve the problem optimally and last, we presented an algorithm for collecting the maximum gain at the root while using the smallest starting budget.

In Chapter 5 we studied the problem of near-gathering in general graphs. We showed that the problem of minimizing the maximum pairwise distance of the agents is NP-hard to approximate within a factor of $2 - o(1)$ and on the positive side, we presented a 2-approximation algorithm. Furthermore, we studied the problem with the objective of minimizing the average pairwise distance of the agents. To this end, we presented a $2(1 - \frac{1}{k})$-approximation algorithm.

Finally, in Chapter 6 we considered the problem of collaborative delivery under the assumption that the agents can share their energy. We showed that if the agents can share their energy under no constraints, then collaborative delivery is strongly NP-hard. Inspired by the construction we used for the reduction, we designed a constant factor resource augmented algorithm for solving the problem in polynomial time. Moreover, we showed that for the case where the agents have restricted energy capacity, the problem is NP-hard even when $B = 2$, if the agent allocation to the homebase nodes is given as part of the input. However, interestingly enough, we proved next that for $B = 2$, the delivery problem in which one is given the total available energy and is asked if it is possible to distribute this energy to agents at the homebase nodes in order to achieve delivery, is solvable in polynomial time. In fact, what we proved is that the underlying optimization problem, i.e., finding the minimum amount of energy that can be distributed to the homebases so that delivery is feasible, is solvable in polynomial time by reduction to a shortest path computation in a complete directed graph.

Future Work

The topic of energy-aware mobile agents is relatively new, as a result there are many problems in the field of mobile agent computing that remain to be considered under this model. Nevertheless, there are some interesting questions that have arisen out of this work and future work perspectives which we would like to address.

For the online tree exploration, the lower bound of $\Omega(\log B)$ on the competitive ratio of exploration that we proved holds only in the local communication model. An interesting question, therefore, is whether more efficient algorithms are possible for tree exploration in the global communication model as well as if there exists a non-trivial constant lower bound on the competitive ratio. Another open question is the cost of exploring general graphs or other specific classes of graphs. Notice that for classes of graphs other than trees, further assumptions for the model need to be made, for example, we would require unique identifiers for the nodes so that the agents would be able to distinguish between them.

In the partial exploration problem there exist a gap between the lower and the upper bound that we showed in this work and we believe that both bounds can be improved. Furthermore, we only considered the global communication model, it would be interesting to try to develop algorithms to work under the local communication model. Additionally, as for the previous problem, the case of partial exploration in other classes of graphs remains open.

For the sustainable exploration, as in the previous cases, we considered only tree topologies, therefore, the same question still holds, what happens in the case of general graphs and other graph topologies? Another direction for this problem would be to consider that the agent has limited energy capacity. In such a case the goal would be to maximize the number of nodes the agent visits, instead of maximizing the energy it collects.

In the near-gathering problem, for the objective of minimizing the average pairwise distance of the agents, while we provided a constant factor approximation algorithm, the lower bounds for the problem remain open. Moreover, near-gathering belongs to a bigger family of problems, namely the movement problems. Therefore, the study of other problems belonging to this family would be a possible direction for future work.

For the collaborative delivery, as we already discussed in subsection 6.2.3, if $B > 2$, our algorithm no longer returns the optimal solution. As a result, a natural question is how to handle greater battery capacities, i.e., $B \geq 3$. Furthermore, it would be interesting to investigate whether our NP-completeness reduction generalizes for greater values of B.

Bibliography

[AFS13] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. “Local-search Based Approximation Algorithms for Mobile Facility Location Problems”. In: *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’13. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2013, pp. 1607–1621. ISBN: 978-1-611972-51-1. URL: http://dl.acm.org/citation.cfm?id=2627817.2627932 (cit. on p. 18).

[AH00] Susanne Albers and Monika R. Henzinger. “Exploring Unknown Environments”. In: *SIAM J. Comput.* 29.4 (Feb. 2000), pp. 1164–1188. ISSN: 0097-5397. DOI: 10.1137/S009753979732428X. URL: http://dx.doi.org/10.1137/S009753979732428X (cit. on p. 14).

[Amb+11] Christoph Ambühl, Leszek Gąsieniec, Andrzej Pelc, et al. “Tree Exploration with Logarithmic Memory”. In: *ACM Trans. Algorithms* 7.2 (Mar. 2011), 17:1–17:21. ISSN: 1549-6325. DOI: 10.1145/1921659.1921663. URL: http://doi.acm.org/10.1145/1921659.1921663 (cit. on p. 14).

[Ana+16] Julian Anaya, Jérémie Chalopin, Jurek Czyzowicz, et al. “Convergecast and Broadcast by Power-Aware Mobile Agents”. In: *Algorithmica* 74.1 (2016), pp. 117–155 (cit. on p. 19).

[Arc+11] Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, et al. “Improved Approximation Algorithms for Prize-Collecting Steiner Tree and TSP”. In: *SIAM Journal on Computing* 40.2 (2011), pp. 309–332. DOI: 10.1137/090771429. eprint: https://doi.org/10.1137/090771429. URL: https://doi.org/10.1137/090771429 (cit. on p. 16).

[Awe+99] Baruch Awerbuch, Margrit Betke, Ronald L. Rivest, et al. “Piecemeal Graph Exploration by a Mobile Robot”. In: *Information and Computation* 152.2 (1999), pp. 155–172. ISSN: 0890-5401. DOI: http://dx.doi.org/10.1006/inco.1999.2795. URL: http://www.sciencedirect.com/science/article/pii/S0890540199927955 (cit. on pp. 15, 87).

[Bal89] Egon Balas. “The prize collecting traveling salesman problem”. In: *Networks* 19.6 (1989), pp. 621–636. ISSN: 1097-0037. DOI: 10.1002/net.3230190602. URL: http://dx.doi.org/10.1002/net.3230190602 (cit. on p. 16).

[Bam+17a] Evangelos Bampas, Jérémie Chalopin, Shantanu Das, et al. "Maximal exploration of trees with energy-constrained agents". In: *ALGOTEL 2017 - 19èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*. Quiberon, France, May 2017. URL: https://hal.archives-ouvertes.fr/hal-01523302 (cit. on p. 37).

[Bam+17b] Evangelos Bampas, Shantanu Das, Dariusz Dereniowski, et al. "Collaborative delivery by energy-sharing low-power mobile robots". In: *13th International Symposium on Algorithms and Experiments for Wireless Networks, ALGOSENSORS 2017, Vienna, Austria, September 7-8, 2017*. 2017, (to appear) (cit. on p. 64).

[Bär+16] Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, et al. "Collaborative Delivery with Energy-Constrained Mobile Robots". In: *Structural Information and Communication Complexity - 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers*. Ed. by Jukka Suomela. Vol. 9988. Lecture Notes in Computer Science. 2016, pp. 258–274 (cit. on p. 18).

[Bär+17] Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, et al. "Energy-Efficient Delivery by Heterogeneous Mobile Agents". In: *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*. Ed. by Heribert Vollmer and Brigitte Vallée. Vol. 66. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, 10:1–10:14 (cit. on p. 18).

[BGP17] Andreas Bärtschi, Daniel Graf, and Paolo Penna. "Truthful Mechanisms for Delivery with Agents". In: *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*. Ed. by Gianlorenzo D'Angelo and Twan Dollevoet. Vol. 59. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 1–17. ISBN: 978-3-95977-042-2. DOI: 10.4230/OASIcs.ATMOS.2017.2. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7889 (cit. on p. 19).

[BT17] Andreas Bärtschi and Thomas Tschager. "Energy-Efficient Fast Delivery by Mobile Agents". In: *Fundamentals of Computation Theory: 21st International Symposium, FCT 2017, Bordeaux, France, September 11–13, 2017, Proceedings*. Ed. by Ralf Klasing and Marc Zeitoun. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 82–95. ISBN: 978-3-662-55751-8. DOI: 10.1007/978-3-662-55751-8_8. URL: https://doi.org/10.1007/978-3-662-55751-8_8 (cit. on p. 19).

[Ben+02] Michael A. Bender, Antonio Fernández, Dana Ron, et al. “The Power of a Pebble: Exploring and Mapping Directed Graphs”. In: *Information and Computation* 176.1 (2002), pp. 1–21. ISSN: 0890-5401. DOI: http://dx.doi.org/10.1006/inco.2001.3081. URL: http://www.sciencedirect.com/science/article/pii/S0890540101930810 (cit. on p. 14).

[BS94] Michael A. Bender and Donna K. Slonim. “The Power of Team Exploration: Two Robots Can Learn Unlabeled Directed Graphs”. In: *35th Symposium on Foundations of Computer Science, FOCS’94*. 1994, pp. 75–85 (cit. on p. 14).

[BDZ11] Piotr Berman, Erik D. Demaine, and Morteza Zadimoghaddam. “O(1)-Approximations for Maximum Movement Problems”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*. Ed. by Leslie Ann Goldberg, Klaus Jansen, R. Ravi, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 62–74. ISBN: 978-3-642-22935-0. DOI: 10.1007/978-3-642-22935-0_6. URL: https://doi.org/10.1007/978-3-642-22935-0_6 (cit. on p. 17).

[BP89] Marshall Bern and Paul Plassmann. “The Steiner Problem with Edge Lengths 1 and 2,” in: *Inf. Process. Lett.* 32.4 (Sept. 1989), pp. 171–176. ISSN: 0020-0190. DOI: 10.1016/0020-0190(89)90039-2. URL: http://dx.doi.org/10.1016/0020-0190(89)90039-2 (cit. on p. 70).

[BRS95] Margrit Betke, Ronald L. Rivest, and Mona Singh. “Piecemeal learning of an unknown environment”. In: *Machine Learning* 18.2 (Feb. 1995), pp. 231–254. ISSN: 1573-0565. DOI: 10.1007/BF00993411. URL: https://doi.org/10.1007/BF00993411 (cit. on pp. 15, 87).

[Bie+93] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, et al. “A note on the prize collecting traveling salesman problem”. In: *Mathematical Programming* 59.1 (Mar. 1993), pp. 413–420. ISSN: 1436-4646. DOI: 10.1007/BF01581256. URL: https://doi.org/10.1007/BF01581256 (cit. on p. 16).

[Bil+13] Davide Bilò, Yann Disser, Luciano Gualà, et al. “Polygon-Constrained Motion Planning Problems”. In: *9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics ALGOSENSORS’13*. 2013, pp. 67–82 (cit. on p. 18).

[Bil+16] Davide Bilò, Luciano Gualà, Stefano Leucci, et al. "Exact and approximate algorithms for movement problems on (special classes of) graphs". In: *Theoretical Computer Science* 652.Supplement C (2016), pp. 86–101. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2016.09.007. URL: http://www.sciencedirect.com/science/article/pii/S0304397516304807 (cit. on p. 18).

[Cha+13] Jérémie Chalopin, Shantanu Das, Matúš Mihalák, et al. "Data Delivery by Energy-Constrained Mobile Agents". In: *9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics ALGOSENSORS'13*. 2013, pp. 111–122 (cit. on p. 18).

[Cha+14] Jérémie Chalopin, Riko Jacob, Matúš Mihalák, et al. "Data Delivery by Energy-Constrained Mobile Agents on a Line". In: *41st International Colloquium on Automata, Languages, and Programming ICALP'14*. 2014, pp. 423–434 (cit. on p. 18).

[CRW04] Fabián A. Chudak, Tim Roughgarden, and David P. Williamson. "Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation". In: *Mathematical Programming* 100.2 (June 2004), pp. 411–421. ISSN: 1436-4646. DOI: 10.1007/s10107-003-0479-2. URL: https://doi.org/10.1007/s10107-003-0479-2 (cit. on p. 70).

[Czy+16] Jurek Czyzowicz, Krzysztof Diks, Jean Moussi, et al. "Communication Problems for Mobile Agents Exchanging Energy". In: *Structural Information and Communication Complexity: 23rd International Colloquium, SIROCCO 2016, Helsinki, Finland, July 19-21, 2016, Revised Selected Papers*. Ed. by Jukka Suomela. Cham: Springer International Publishing, 2016, pp. 275–288. ISBN: 978-3-319-48314-6. DOI: 10.1007/978-3-319-48314-6_18. URL: https://doi.org/10.1007/978-3-319-48314-6_18 (cit. on pp. 19, 64).

[Czy+17] Jurek Czyzowicz, Krzysztof Diks, Jean Moussi, et al. "Energy-Optimal Broadcast in a Tree with Mobile Agents". In: *13th International Symposium on Algorithms and Experiments for Wireless Networks, ALGOSENSORS 2017, Vienna, Austria, September 7-8, 2017*. 2017, (to appear) (cit. on p. 19).

[DDU17] S. Das, D. Dereniowski, and P. Uznański. "Energy Constrained Depth First Search". In: *ArXiv e-prints* (Sept. 2017). arXiv: 1709.10146 [cs.DS] (cit. on p. 15).

[Das13] Shantanu Das. "Mobile agents in distributed computing: Network exploration". In: *Bulletin of the EATCS* 109.2 (2013), pp. 54–69 (cit. on p. 8).

[DDK15] Shantanu Das, Dariusz Dereniowski, and Christina Karousatou. “Collaborative Exploration by Energy-Constrained Mobile Robots”. In: *22th International Colloquium on Structural Information and Communication Complexity SIROCCO’15*. 2015, pp. 357–369 (cit. on p. 22).

[DDKar] Shantanu Das, Dariusz Dereniowski, and Christina Karousatou. “Collaborative Exploration of Trees by Energy-Constrained Mobile Robots”. In: *Theory of Computing Systems* (to appear) (cit. on p. 22).

[Dem+09] Erik D. Demaine, Mohammadtaghi Hajiaghayi, Hamid Mahini, et al. “Minimizing Movement”. In: *ACM Trans. Algorithms* 5.3 (2009), pp. 1–30 (cit. on p. 17).

[DHM14] Erik D. Demaine, Mohammadtaghi Hajiaghayi, and Dániel Marx. “Minimizing Movement: Fixed-Parameter Tractability”. In: *ACM Trans. Algorithms* 11.2 (Oct. 2014), 14:1–14:29. ISSN: 1549-6325. DOI: 10.1145/2650247. URL: http://doi.acm.org/10.1145/2650247 (cit. on p. 18).

[DP99] Xiaotie Deng and Christos H. Papadimitriou. “Exploring an unknown graph”. In: *Journal of Graph Theory* 32.3 (1999), pp. 265–297. ISSN: 1097-0118. DOI: 10.1002/(SICI)1097-0118(199911)32:3<265::AID-JGT6>3.0.CO;2-8. URL: http://dx.doi.org/10.1002/(SICI)1097-0118(199911)32:3%3C265::AID-JGT6%3E3.0.CO;2-8 (cit. on p. 14).

[Der+15] Dariusz Dereniowski, Yann Disser, Adrian Kosowski, et al. “Fast collaborative graph exploration”. In: *Information and Computation* 243.Supplement C (2015). 40th International Colloquium on Automata, Languages and Programming (ICALP 2013), pp. 37–49. ISSN: 0890-5401. DOI: http://dx.doi.org/10.1016/j.ic.2014.12.005. URL: http://www.sciencedirect.com/science/article/pii/S0890540114001576 (cit. on p. 15).

[DP04] Anders Dessmark and Andrzej Pelc. “Optimal graph exploration without good maps”. In: *Theoretical Computer Science* 326.1 (2004), pp. 343–362. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2004.07.031. URL: http://www.sciencedirect.com/science/article/pii/S0304397504004773 (cit. on p. 13).

[Dik+04] Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, et al. “Tree Exploration with Little Memory”. In: *J. Algorithms* 51.1 (Apr. 2004), pp. 38–63. ISSN: 0196-6774. DOI: 10.1016/j.jalgor.2003.10.002. URL: http://dx.doi.org/10.1016/j.jalgor.2003.10.002 (cit. on p. 14).

[Dis+17] Yann Disser, Frank Mousset, Andreas Noever, et al. “A general lower bound for collaborative tree exploration”. In: *Structural Information and Communication Complexity: 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017. Proceedings* (2017), (to appear) (cit. on p. 15).

[DKK06] Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. “Optimal Constrained Graph Exploration”. In: *ACM Trans. Algorithms* 2.3 (July 2006), pp. 380–402. ISSN: 1549-6325. DOI: 10.1145/1159892.1159897. URL: http://doi.acm.org/10.1145/1159892.1159897 (cit. on p. 15).

[Dyn+06] M. Dynia, J. Kutyłowski, F. Meyer auf der Heide, et al. “Smart Robot Teams Exploring Sparse Trees”. In: *Mathematical Foundations of Computer Science 2006: 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006. Proceedings*. Ed. by Rastislav Královič and Paweł Urzyczyn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 327–338. ISBN: 978-3-540-37793-1. DOI: 10.1007/11821069_29. URL: https://doi.org/10.1007/11821069_29 (cit. on p. 15).

[DKS06] Miroslaw Dynia, Miroslaw Korzeniowski, and Christian Schindelhauer. “Power-Aware Collective Tree Exploration”. In: *Architecture of Computing Systems - ARCS 2006: 19th International Conference, Frankfurt/Main, Germany, March 13-16, 2006. Proceedings*. Ed. by Werner Grass, Bernhard Sick, and Klaus Waldschmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 341–351. ISBN: 978-3-540-32766-0. DOI: 10.1007/11682127_24. URL: https://doi.org/10.1007/11682127_24 (cit. on p. 16).

[DŁS07] Miroslaw Dynia, Jakub Łopuszański, and Christian Schindelhauer. “Why Robots Need Maps”. In: *Structural Information and Communication Complexity: 14th International Colloquium, SIROCCO 2007, Castiglioncello, Italy, June 5-8, 2007. Proceedings*. Ed. by Giuseppe Prencipe and Shmuel Zaks. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 41–50. ISBN: 978-3-540-72951-8. DOI: 10.1007/978-3-540-72951-8_5. URL: https://doi.org/10.1007/978-3-540-72951-8_5 (cit. on pp. 15, 16).

[FPS01] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. “Sharing the Cost of Multicast Transmissions”. In: *Journal of Computer and System Sciences* 63.1 (2001), pp. 21–41. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.2001.1754. URL: http://www.sciencedirect.com/science/article/pii/S0022000001917544 (cit. on p. 17).

[Feo+07] Paulo Feofiloff, Cristina G. Fernandes, Carlos E. Ferreira, et al. “Primal-dual approximation algorithms for the Prize-Collecting Steiner Tree Problem”. In: *Information Processing Letters* 103.5 (2007), pp. 195–202. ISSN: 0020-0190. DOI: https://doi.org/10.1016/j.ipl.2007.03.012. URL: http://www.sciencedirect.com/science/article/pii/S0020019007000907 (cit. on p. 16).

[FT05] Rudolf Fleischer and Gerhard Trippen. “Exploring an Unknown Graph Efficiently”. In: *Algorithms – ESA 2005: 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005. Proceedings*. Ed. by Gerth Stølting Brodal and Stefano Leonardi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 11–22. ISBN: 978-3-540-31951-1. DOI: 10.1007/11561071_4. URL: https://doi.org/10.1007/11561071_4 (cit. on p. 14).

[FPS12] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2012 (cit. on p. 8).

[FW16] Klaus-Tycho Foerster and Roger Wattenhofer. “Lower and upper competitive bounds for online directed graph exploration”. In: *Theoretical Computer Science* 655.Part A (2016). Special Issue on Theory and Applications of Graph Searching Problems, pp. 15–29. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2015.11.017. URL: http://www.sciencedirect.com/science/article/pii/S0304397515010166 (cit. on p. 14).

[Fra+06] Pierre Fraigniaud, Leszek Gąsieniec, Dariusz R. Kowalski, et al. “Collective Tree Exploration”. In: *Networks* 48.3 (Oct. 2006), pp. 166–177. ISSN: 0028-3045. DOI: 10.1002/net.v48:3. URL: http://dx.doi.org/10.1002/net.v48:3 (cit. on pp. 14, 16).

[Fra+05] Pierre Fraigniaud, David Ilcinkas, Guy Peer, et al. “Graph exploration by a finite automaton”. In: *Theoretical Computer Science* 345.2 (2005). Mathematical Foundations of Computer Science 2004, pp. 331–344. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2005.07.014. URL: http://www.sciencedirect.com/science/article/pii/S0304397505003993 (cit. on p. 14).

[FIP08] Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. “Tree exploration with advice”. In: *Information and Computation* 206.11 (2008), pp. 1276–1287. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2008.07.005. URL: http://www.sciencedirect.com/science/article/pii/S0890540108000941 (cit. on p. 13).

[FS11] Zachary Friggstad and Mohammad R. Salavatipour. “Minimizing Movement in Mobile Facility Location Problems”. In: *ACM Trans. Algorithms* 7.3 (July 2011), 28:1–28:22. ISSN: 1549-6325. DOI: 10.1145/1978782.1978783. URL: http://doi.acm.org/10.1145/1978782.1978783 (cit. on p. 17).

[Fuk+99] Munehiro Fukuda, Lubomir F. Bic, Michael B. Dillencourt, et al. “Messages versus Messengers in Distributed Programming”. In: *Journal of Parallel and Distributed Computing* 57.2 (1999), pp. 188–211. ISSN: 0743-7315. DOI: https://doi.org/10.1006/jpdc.1999.1533. URL: http://www.sciencedirect.com/science/article/pii/S0743731599915332 (cit. on p. 8).

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN: 0716710447 (cit. on p. 65).

[Gar05] Naveen Garg. “Saving an Epsilon: A 2-approximation for the k-MST Problem in Graphs”. In: *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*. STOC ’05. Baltimore, MD, USA: ACM, 2005, pp. 396–402. ISBN: 1-58113-960-8. DOI: 10.1145/1060590.1060650. URL: http://doi.acm.org/10.1145/1060590.1060650 (cit. on p. 17).

[Goe09] Michel X. Goemans. “Combining Approximation Algorithms for the Prize-Collecting TSP”. In: *ArXiv e-prints* (Oct. 2009). arXiv: 0910.0553 [cs.DS] (cit. on p. 16).

[GW95] Michel X. Goemans and David P. Williamson. “A General Approximation Technique for Constrained Forest Problems”. In: *SIAM J. Comput.* 24.2 (Apr. 1995), pp. 296–317. ISSN: 0097-5397. DOI: 10.1137/S0097539793242618. URL: http://dx.doi.org/10.1137/S0097539793242618 (cit. on p. 16).

[Hig+14] Yuya Higashikawa, Naoki Katoh, Stefan Langerman, et al. “Online graph exploration algorithms for cycles and trees by multiple searchers”. In: *Journal of Combinatorial Optimization* 28.2 (Aug. 2014), pp. 480–495. ISSN: 1573-2886. DOI: 10.1007/s10878-012-9571-y. URL: https://doi.org/10.1007/s10878-012-9571-y (cit. on p. 14).

[JMP00] David S. Johnson, Maria Minkoff, and Steven Phillips. “The Prize Collecting Steiner Tree Problem: Theory and Practice”. In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’00. San Francisco, California, USA: Society for Industrial and Applied Mathematics, 2000, pp. 760–769. ISBN: 0-89871-453-2. URL: http://dl.acm.org/citation.cfm?id=338219.338637 (cit. on p. 16).

[Kar72] Richard Karp. *Reducibility Among Combinatorial Problems*. Jan. 1972 (cit. on pp. 44, 57).

[Kos13] Adrian Kosowski. "Time and Space-Efficient Algorithms for Mobile Agents in an Anonymous Network". Habilitation à diriger des recherches. Université Sciences et Technologies - Bordeaux I, Sept. 2013. URL: https://tel.archives-ouvertes.fr/tel-00867765 (cit. on p. 8).

[KKM10] Evangelos Kranakis, Danny Krizanc, and Euripides Markou. *The Mobile Agent Rendezvous Problem in the Ring*. 1st. Morgan and Claypool Publishers, 2010. ISBN: 1608451364, 9781608451364 (cit. on p. 8).

[LO99] Danny B. Lange and Mitsuru Oshima. "Seven Good Reasons for Mobile Agents". In: *Commun. ACM* 42.3 (Mar. 1999), pp. 88–89. ISSN: 0001-0782. DOI: 10.1145/295685.298136. URL: http://doi.acm.org/10.1145/295685.298136 (cit. on p. 8).

[MMS11] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. "Online Graph Exploration: New Results on Old and New Algorithms". In: *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*. Ed. by Luca Aceto, Monika Henzinger, and Jiří Sgall. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 478–489. ISBN: 978-3-642-22012-8. DOI: 10.1007/978-3-642-22012-8_38. URL: https://doi.org/10.1007/978-3-642-22012-8_38 (cit. on p. 13).

[OS12] Christian Ortolf and Christian Schindelhauer. "Online Multi-robot Exploration of Grid Graphs with Rectangular Obstacles". In: *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '12. Pittsburgh, Pennsylvania, USA: ACM, 2012, pp. 27–36. ISBN: 978-1-4503-1213-4. DOI: 10.1145/2312005.2312010. URL: http://doi.acm.org/10.1145/2312005.2312010 (cit. on p. 15).

[OS14] Christian Ortolf and Christian Schindelhauer. "A Recursive Approach to Multi-robot Exploration of Trees". In: *Structural Information and Communication Complexity: 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*. Ed. by Magnús M. Halldórsson. Cham: Springer International Publishing, 2014, pp. 343–354. ISBN: 978-3-319-09620-9. DOI: 10.1007/978-3-319-09620-9_26. URL: https://doi.org/10.1007/978-3-319-09620-9_26 (cit. on p. 14).

[PP99] Petrişor Panaite and Andrzej Pelc. “Exploring Unknown Undirected Graphs”. In: *Journal of Algorithms* 33.2 (1999), pp. 281–295. ISSN: 0196-6774. DOI: http://dx.doi.org/10.1006/jagm.1999.1043. URL: http://www.sciencedirect.com/science/article/pii/S019667749991043X (cit. on p. 13).

[PS06] Giuseppe Prencipe and Nicola Santoro. “Distributed Algorithms for Autonomous Mobile Robots”. In: *Fourth IFIP International Conference on Theoretical Computer Science- TCS 2006: IFIP 19th Worm Computer Congress, TC-1, Foundations of Computer Science, August 23–24, 2006, Santiago, Chile*. Ed. by Gonzalo Navarro, Leopoldo Bertossi, and Yoshiharu Kohayakawa. Boston, MA: Springer US, 2006, pp. 47–62. ISBN: 978-0-387-34735-6. DOI: 10.1007/978-0-387-34735-6_8. URL: https://doi.org/10.1007/978-0-387-34735-6_8 (cit. on p. 17).

[Sha51] Claude E. Shannon. “Presentation of a Maze-Solving Machine”. In: *8th Conf. of the Josiah Macy Jr. Foundation (Cybernetics)* (1951), pp. 173–180 (cit. on p. 13).

[Tov84] Craig A. Tovey. “A simplified NP-complete satisfiability problem”. In: *Discrete Applied Mathematics* 8.1 (1984), pp. 85–89. ISSN: 0166-218X. DOI: http://dx.doi.org/10.1016/0166-218X(84)90081-7 (cit. on p. 72).