UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DATA COMMUNICATION PROBLEMS USING MOBILE AGENTS EXCHANGING ENERGY

ÉCHANGE D'ÉNERGIE ENTRE AGENTS MOBILES POUR SOLUTIONNER DES PROBLÈMES DE COMMUNICATION

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

PAR
JEAN MOUSSI

SOUS LA SUPERVISION DU PROFESSEUR JUREK CZYZOWICZ

DÉCEMBRE 2018

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS
Département d'informatique et d'ingénierie

Cette thèse intitulé :

DATA COMMUNICATION PROBLEMS USING MOBILE AGENTS EXCHANGING ENERGY

ÉCHANGE D'ÉNERGIE ENTRE AGENTS MOBILES POUR SOLUTIONNER DES PROBLÈMES DE COMMUNICATION

présentée par
Jean Moussi
pour l'obtention du grade de Philosophiea Doctor (Ph.D)

Membres du jury :

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

Thèse acceptée le : .....

À ma famille.

À mon épouse et mes enfants.

À tous les collègues qui m'ont encouragés et qui m'ont supportés.

Remerciement

La réalisation de cette thèse a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

Je tiens à remercier mon directeur de recherche le Dr. Jurek Czyzowicz, qui m'a encadré tout au long de cette thèse. Il a toujours été présent à mes côtés pour m'orienter. Il m'a permis d'approfondir au maximum mes travaux afin de pouvoir être fier aujourd́hui du travail réalisé. Qu'il soit aussi remercié pour sa gentillesse, sa disponibilité et pour les nombreux encouragements qu'il m'a prodigués ainsi que pour son soutien tant au niveau personnel qu'intellectuel.

Au terme de ce parcours, je remercie celles et ceux qui me sont chers, ma femme Rita et mes enfants Dominique, Natacha et Alain, que j'ai quelque peu délaissés ces derniers mois pour achever cette thèse. Leurs attentions et encouragements m'ont accompagnés tout au long de ces années.

Membres du jury, Dr. Paola Flocchini, Dr. Kamel Adi et Dr. Marek Zaremba, je tiens à vous dire comment je suis honoré que vous ayez accepté d'être présents à la défense de ma thèse.

La thèse de doctorat représente un travail de longue haleine, pour cette raison je n'ai jamais pensé qu'un jour je serais là pour présenter cette recherche. Je suis très fier de cet accomplissement et je suis confiant que mes collègues et les membres de ma famille sont fiers de moi et surtout contents pour moi.

Contents

Résumé **iii**

Abstract **iv**

1 Introduction **1**
1.1 Motivation 1
1.2 Mobile Agents 2
1.3 Data Delivery Problems 3

2 Literature Review **5**
2.1 Introduction 5
2.2 Mobile Agents in Every-Day Life 7
2.3 Environment Exploration 10
2.3.1 Exploration by a Single Agent 10
2.3.1.1 Graph Exploration 10
2.3.1.2 Exploration of Geometric Environment 14
2.3.2 Exploration and Searching Problem by a Team of Mobile Robots 16
2.4 Rendezvous and Gathering Problems 23
2.5 Other Search-Type Problems for Teams of Mobile Agents 29
2.5.1 Evacuation Problem 29
2.5.2 Boundary Patrolling Problem 31
2.5.3 Cops and Robbers Problem 33
2.6 Data Delivery 34

3 Summary of results **40**
3.1 Preliminaries 40
3.2 The Problems 41
3.3 The Results Obtained 43

4 Data Delivery and Convergecast **45**
4.1 The Line Environment 46
4.1.1 Data Delivery 46
4.1.2 Convergecast 51
4.2 The Tree Environment 53
4.2.1 Data Delivery 53
4.2.2 Convergecast 54
4.3 NP-Completeness for Digraphs and Graphs 58

5 Broadcast when Agents Start at the Same Node **63**
5.1 Agents Starting from the Source Node 64
5.1.1 Scheduling Agent Movements when Critical Leaves are Known 66
5.1.2 A Schematic Algorithm Computing the Minimal Cost 69
5.1.3 Efficient Implementation of the Algorithm Schematic-MinCost 75
5.1.4 Unlimited Number of Agents in the Source 78
5.2 All Agents Start from the Same Node r that is Different from the Source s 80

6 Broadcast with Mobile Agents Distributed on a Tree **86**
6.1 Introduction 86
6.1.1 Preliminairies 89
6.2 Testing Feasibility of Broadcast 91
6.3 Constructing broadcast schedule 101
6.4 Proof of correctness of the algorithm Broadcast-in-Tree(T) . . . 106

7 Final remarks **110**

Résumé

Des sommets d'un réseau informatique ont la capacité de stockage de données. Ces données peuvent être transportées vers les autres sommets par les agents mobiles les visitant. Des agents traversent les arcs du réseau en consommant l'énergie proportionnellement à la distance parcourue. Des agents peuvent se rencontrer dans le réseau et au moment d'un tel rendezvous peuvent échanger les informations possédées ainsi qu'une portion quelconque de l'énergie. Des agents collaborent afin de transferer l'information entre des sommets du réseau.

Dans cette thèse les agents opèrent dans le réseau sous forme d'un arbre pondéré. Initialement un sous-ensemble de sommets contient des agents mobiles, chacun avec une quantité d'énergie (possiblement différente pour chaque agent). Le poids associé à un arc de l'arbre correspond à la distance qui doit être parcourue en traversant cet arc. Trois protocoles de communication sont analysés:

1. La *livraison de données* ou l'information initiale d'un sommet s doit être transportée par les agents vers un sommet t de l'arbre.

2. La *diffusion* ou l'information d'un sommet s doit être transportée vers tous les autres sommets de l'arbre.

3. La *consolidation* ou l'information de tous les sommets de l'arbre doit être transportée par les agents afin d'être consolidée dans le sommet t.

L'objectif de cette thèse est de donner les algorithmes de décision pour chaque protocole de communication. Chaque algorithme proposé doit décider si la configuration initiale des agents dans le réseau et leurs quantités initiales d'énergie permettent de réaliser le protocole de communication correspondant.

Abstract

A computer network contains nodes at which data may be stocked. Such data packets may be carried by mobile agents and deposited at other nodes. The agents travel along the network using energy proportionnally to the distance traversed. When agents meet, they can exchange the possessed data as well as any amount of energy. The agents collaborate in order to realize a data transfer between network nodes.

In this thesis the considered environment is a weighted tree network. Initially agents are placed in a sub-set of network nodes, each agent with some fixed amount of energy (possibly different). The weight of an edge equals the distance which needs to be traversed by walking along this edge.

Three communication protocols are being studied:

1. The *data delivery*, when the initial data of node s needs to be delivered by the agents to node t of the tree.

2. The *broadcast*, when the data packet of node s needs to be transferred to all other nodes of the tree.

3. The *convergecast*, when the data packets of all tree nodes must be transported by the agents to node t.

The objectif of this thesis is to provide decision algorithms for all communication protocols. Each proposed algorithm must decide whether the initial configuration of the agents in the tree and their energy levels allows to realize the corresponding communication protocol.

Chapter 1

Introduction

1.1 Motivation

The mobile agents are an important component of people everyday life such as autonomous vehicles that are controlled by sensors of extraordinary intelligence. These vehicles render an important service in the daily life of the blind, the elderly, and those who do not have a driving license. With more research these vehicles can deliver merchandise from a source point s to a terminal point t. Who knows, thanks to the intelligence of distributed computing and the technology of a system based on the model of the mobile agents we will one day have autonomous freighter planes to transport goods from one city to another and one country to another.

Autonomous vehicles are the future, the next challenge for research. Certainly this application is the residue of the mobile-agent model, a model that is based on collections of mobile agents executing their tasks in parallel. This technology will have important impacts on mobility behaviour and spatial planning.

The school transportation system is based on a model of parallel mobile agents to transport students. Buses play the role of mobile agents who are controlled by qualified drivers. The main research problem here is to schedule the movements

of the buses in order to do it using minimum time and/or energy. This is a fundamental problem in operational research known as the Vehicle Routing Problem (VRP). This is an old problem and several versions of it has been solved using mathematical optimization algorithms. Such a problem is one of the numerous examples of theoretical problems in mathematics and computer science involving mobile agents

1.2 Mobile Agents

Mobile agents emerged in the mid-1990s and have raised considerable interest in the research community. Proponents associate several benefits with their use. A mobile agent is a process that can move from machine to machine, from one state to another, in a database from one record to another, from one field to another, from one node to another in a tree or any connected graph, in order to perform a specific task.

A mobile agent is related to an algorithm that guides the agent in its move. It is a mobile robot able to observe, communicate and perform calculations necessary for its operation. There are as many different types of robots as there are tasks for them to perform. Mobile agents are particularly useful if they work in teams, collaborating with one another, by exchanging individually collected information, in view of achieving a common goal. Depending on the application, mobile agents may have different capabilities with the most fundamental ones being the following:

1. Mobility, i.e. an agent is capable of moving inside its environment.

2. Perception, i.e. an agent has the capability to observe elements of its environment, including the presence of other agents.

3. Computation, i.e. an agent can execute calculations by an algorithm, using some input data that it may acquire during its work.

4. Communication, i.e. agents may exchange possessed information or send it to the central authority.

Computer science research projects related to physical mobile robots are usually done in the domain of robotics, where the capacities of robots, their physical dimensions, and their reliability are all taken into account.

Within theoretical computer science, mobile agent problems are mainly investigated in the domain of operational research, and algorithms (often distributed algorithms). Most often, related problems are formulated as some optimization questions, where researchers look to minimize time needed to complete the task, energy used, memory needed, communication (e.g. the number of synchronous rounds), etc. The fundamental task often given to mobile robots is exploration of their environment, sometimes in order to find some target or in order to learn or produce a map of the environment.

1.3 Data Delivery Problems

In the present thesis we address the question of data communication by mobile agents in tree networks. We suppose that agents may communicate only when being in the same time moment at the same location (a so-called face-to-face (F2F) communication model) and they need to perform the following tasks:

1. Data delivery, i.e. transferring a packet of data from some node to some other node of the tree

2. Convergecast, i.e. integration of the data from all other nodes in the hands of a single agent

3. Broadcasting, i.e. transferring a data packet of some specific node to all other nodes of the tree

We assume that each agent possesses an amount of energy that it may exchange with other agents when meeting them. The objective is to verify whether the desired communication task is feasible.

Chapter 2

Literature Review

2.1 Introduction

We briefly start with few arbitrarily chosen examples of mobile robots used in every day life.

The fundamental research problem concerning mobile agents is related to searching, when the agents need to find a target placed at an unknown position in the environment. This task implies exploration of the environment, which may be:

1. An environment entirely known to the agents, (e.g., [10, 23, 12, 13, 22, 36, 54, 46, 62, 90, 91]).

2. An environment completely unknown to them (e.g., [2, 24, 25, 30, 31, 33, 53, 50, 63, 64]).

3. Some parameters of the environment, or at least its type may be known to the agents, but not the entire environment (e.g., [2, 8, 39, 40, 57, 65, 70, 74, 73, 78, 81, 82, 87]).

In some scenarios, the target may be mobile and its movement is usually set by the adversary so that the "cost" of the search performed by the proposed algorithm

(whatever is defined by the cost in the analyzed setting) is maximized.

Even if the target is considered motionless (e.g., see [13, 22, 36, 54, 62, 87]), the adversary places it at a position, that maximizes the cost of the proposed algorithm. Consequently, the search problem may be considered a game between the two players, the mobile agent (or a group of collaborating agents), and the target, which is represented by the omnipotent adversary, having the full knowledge of the environment and the applied algorithm. Both players clearly have contradictory goals with one trying to complete the search as soon as possible, and the other attempting to delay or even prevent it, if possible.

Somewhat inverse perspective is present in the case of the *rendezvous problem* when two or more agents need to meet as soon as possible. Consequently the rendezvous problem is sometimes considered a game between two or more players having compatible goals of meeting as fast as possible and collaborating in order to achieve this goal (see [3]).

In light of the above, after briefly reviewing signalling scattered positions of literature related to agents in every-day life, the surveyed literature will primarily be devoted to search and rendezvous problems, as well as some variants of research problems related to them.

The last section of this chapter concerns the data delivery problems, which are closely related to the present thesis. Most of the scenarios considered were studied recently, i.e. in the last $4 - 5$ years.

2.2 Mobile Agents in Every-Day Life

The agents are usually of two categories:

- physical robots operating in real environments
- software agents, which are programs that usually "travel" in computer networks

Below, we provide some references to applications of mobile agents in everyday life.

In the Canadian climate, an important winter task is snow removal and the distribution of salt on road networks. Large teams of mobile agents represented by snow plows and salt spreaders need to complete their tasks as quickly as possible so that the population may restart their every-day activities. This problem is a typical question asked in the domain of vehicle routing, cf. [103].

In recent decades there was a lot of progress in the domain of drone technology. A drone, in a technological context, is an unmanned aircraft. Drones are often addressed as unmanned aerial vehicles (UAVs) or unmanned aircraft systems (UASs). Essentially, a drone is a flying robot that may be controlled remotely or flown autonomously through software-controlled flight plans in their embedded systems, working in conjunction with on-board sensors and GPS. Drones are used for monitoring forest fires, fields, and lakes, to detect animal presence in certain places, for military operations, etc. (e.g. see [105]).

A drone is a physical robot that is particularly useful for military operations. However, there are several other types of mobile robots that have military usage. A very well known mobile military robot can save lives and destroy mines. Typi-

cally a military robot is either autonomous or it is controlled from the outside by the owner. In some cases we use two mobile robots to perform this operation, one that locates the bomb, and the other to destroy it. An interested reader may see [108, 109] for recent surveys of mostly mobile military robots. Clearly for security reasons, the above surveys are far from representing the state of current research projects, with most of them involving mobile robots being developed by the Defence Advanced Research Project Agency (DARPA), see [110, 111].

Almost all car-producing companies actively participate in research on autonomous cars called unmanned surface vehicles (USVs) with advance guidance, navigation and control (GNC) capabilities. An autonomous car is a vehicle capable of driving without the intervention of a human being. With many sensors and particularly elaborate calculation software, it is able to move through traffic and to make decisions alone, without a driver. See, for example [112, 113] for recent surveys.

All the examples previously given in this section concern physical robots. Although some of them are controlled by an external authority, the most attractive physical mobile robots are the autonomous ones (cf. [111, 113]). The question of mobile agent autonomy is fundamental in distributed computing.

Mobile robots have numerous applications in healthcare, for example, modern information management in hospitals need interoperability between different health management entities. This kind of control with an ontology on an accepted public health standard, is based on a multi-agent system providing a framework for interactions in a distributed systems environment. This requires a client-server approach to facilitate the flow of patient information into an entire organization

linking hospitals as well as emergency clinics (e.g. see [107]).

In the field of medicine, doctors use mobile robots to practice their jobs, operating on patients using a mobile agent that acts as a camera, a lens, a surgical scalpel, etc. The aim is to reduce the patient's injury, the number of specialists involved, time, as well as the risk of infections. They often use two or more mobile robots. These mobile agents are most often controlled externally by specialists, doctors or technicians. Therefore, we can say that they are robots controlled by external entities (e.g. see [96]).

Other examples from healthcare applications of mobile agents involve mobile agents software. For example, the authors of [97] discuss a distributed medical systems environment using mobile agent models to circulate patient information across a whole healthcare organization. In [104], using mobile agent technology and agent-driven security is being used to give the high level communication and access to patient data, to do some analysis and browse some patient information. Clearly, the above types of applications are not exclusive for the domain of healthcare. For example, the authors of [80] study the information retrieval from electronic calendars for multiparty event scheduling based on mobile agents technology.

There are other situations in which software agents are useful. For example, in electronic commerce, mobiles agents are programs moving along the network searching the sites of the retailer in order to purchase a product (cf. [95, 114, 115]. In information retrieval systems, mobile agent platforms are an alternative to a traditional client/server approach (e.g. see [80, 116]).

2.3 Environment Exploration

The problem of search or exploration of a given environment is the fundamental research question in computer science, robotics, operational research and the related domains. Some authors consider environment search and environment exploration as equivalent problems while for some other authors search implies the knowledge of the environment, while explorations assumes that the environment is a priori unknown. The objective of an exploration is sometimes not only the search for a target placed in an unknown position but also the creation of the map of the environment.

2.3.1 Exploration by a Single Agent

A mobile robot is used to explore a graph or a geometric environment in order to acquire knowledge about it, to look for information stored at unknown place or, for example, to move information from one node to another. Most research in the last 50 years concerned environment exploration by a single agent (e.g. see [2, 23, 22, 52, 77]).

2.3.1.1 Graph Exploration

Typically a mobile agent explores a graph to search for target information placed in a certain node of the graph. Sometimes, when the graph is weighted, the target may be placed at a point belonging to the interior of some edge. As mentioned previously, depending on the studied scenario, the graph may be a priori known to the searcher [23, 22, 62], it may be completely unknown, (e.g. [2, 33, 63]) or partial information about some parameters of the graph may be available to the searching agent (cf. [4, 74, 94]). In some research papers the goal is to create a map of an unknown graph, where the approach taken needs to be different and

the model studied must be usually stronger (e.g. [24, 33]).

The search problem is interesting even in the case of a very simple environments. Several papers (e.g. see [10, 23, 22, 62]) consider the problem of a single agent on a line looking for a target placed at unknown distance D. As knowing the position of the target permits to perform the search in time D, the authors of [10, 23, 22, 62] ask for the *competitive ratio*, i.e. the time of search when the position (and distance) to the target is unknown to the searcher. It is worth noting that, for agents of the same maximal unit speed, the time used and the distance travelled are commensurable, hence using exchangeably either of these measures is justified. This is the case of the above cited papers as well as the case of the present thesis. Suppose that, as studied in [23], we know that a particle that we are looking for is located in the interval $(x, x + dx)$, somewhere along the real line $-\infty < x < \infty$ with a probability density function $g(x)$. We start at some initial point x_0 and can move in either direction. What policy minimizes the expected time required to find the particle, assuming a uniform velocity? [23] and independently [22] gave algorithms completing the search in time $9D$. The authors of [10] proved that $9D$ is the best possible competitive ratio, i.e. the online algorithms of [23, 22] are optimal.

On the other hand, [62] consider the line search when a cost of d is added to each change of direction of the robot. The authors of [62] find an algorithm solving the problem in $O(9 * OPT + 2d)$ time. They also consider the star search or cow-path problem with turn cost, where the hidden object is placed on one of m rays emanating from the original position of the robot. The paper also discusses a trade-off between the corresponding coefficients, and briefly considers randomized

strategies.

In [2] a robot is used to explore an unknown environment (strongly connected graph) to construct a complete map. The robot needs to visit all nodes and edges, attempting to minimize the number of edge traversals. A similar problem for the directed graph has been considered in [64], where one robot explores all edges of an unknown, directed, strongly connected graph (i.e. when every vertex in the graph is reachable from every other vertex). At every point the robot memorizes all edges and nodes visited. The goal is to minimize the ratio of the total number of edges traversed to the optimum number of edge traversals had the graph been known in advance. For an Eulerian graph this ratio is 2; the ratio is unbounded when the deficiency of the graph (defined as the number of edges that have to be added to make it Eulerian) is unbounded. In this paper the authors provide an algorithm that achieves a bounded ratio when the deficiency is bounded; unfortunately the considered competitive ratio is exponential in the deficiency.

In distributed computing the most often studied scenario concerns anonymous graphs (i.e. graph nodes are unlabeled) and the graph is usually unknown to the agents. In such case, if nothing is known about the graph, the construction of the map of the graph is impossible (e.g., see [24, 25]). In [24] a robot explores a strongly connected directed general graph using pebbles. The authors prove that the robot needs one pebble if it knows the upper bound of the number of nodes of the graph. Otherwise, if no upper bound on the size of the graph is provided, $\Theta(\log \log n)$ pebbles are necessary and sufficient to explore the graph. All algorithms considered in [24] are deterministic.

Sometimes the goal of graph exploration is not to optimize its time, but rather the memory used by the agent. In a network exploration task by a mobile agent with a small memory [4], the agent must traverse all the nodes and edges of a network (represented as a non-directed connected graph) and return to the starting node. The network nodes are not tagged and the edge ports are labeled locally at each node. The agent has no prior knowledge of the network topology or its size, and cannot mark the nodes. Under such weak hypotheses, the cycles in the network can prevent the feasibility of the exploration, therefore [4] restricts the consideration to the trees. They present an algorithm to perform tree exploration (with return) using $O(\log n)$ bits of memory for all trees with n nodes.

The fundamental problem concerning anonymous graph exploration using small memory has been studied in [101]. An algorithm given in [101] implies a way to construct in log-space a fixed sequence of directions that guides a deterministic walk by a robot to visit all the vertices of any connected graph. Specifically, [101] gives log-space constructive universal-traversal sequences for graphs with restricted labeling and log-space constructive universal-exploration sequences for general graphs.

The algorithm presented in [101] is, however, unable to produce the map of the explored graph. In [33] a robot needs to construct a map of an unknown environment represented as an unlabeled undirected graph. The robot starts at a single vertex of the graph and it needs to visit all nodes and return to its starting point while constructing the map of the graph. The robot knows the size n of the graph and the maximum degree of its nodes. [33] gives two algorithms, the first one for the minimal number of edges traversed and the second to minimize the

robot memory.

There were numerous other papers studying different scenarios of graph exploration. A robot in [94] explores an unknown, undirected connected graph starting at some vertex and it must visit all nodes of the graph without having to traverse all its edges. When the robot is located at some node it knows the weights of all edges adjacent to this node. The goal in this paper is to find a tour of minimum total costs. Authors of [94] provide a constant competitive algorithm for the case of general graphs with a bounded number of different weights.

Most recently researchers considered exploration of dynamic graphs, i.e. graphs whose structure changes over time (e.g., see [72]). A robot in [74] explores a periodically varying graph (VP) where the edges only exist at moments (unknown) defined by the periodic movements of the carriers. These graphs naturally model highly dynamic networks without infrastructure such as fixed-time public transport, Low Earth Orbit (LEO) satellite systems, security towers, and so on. Different scenarios are considered depending on the knowledge of the length of the longest path, the memory of the robot, the knowledge of the size of the graph and the uniformity of the length of the roads. The authors of [74] present two optimal solution algorithms in the worst case: one for anonymous graph systems and one for graphs with distinct node identifiers.

2.3.1.2 Exploration of Geometric Environment

The geometric environment is usually represented in two-dimensional Cartesian space. It is either an empty plane or a bounded or unbounded polygonal terrain, possibly with polygonal obstacles.

One of the very first papers on exploration geometric environments is [63], where the terrain considered, as well as the obstacles are rectilinear. The authors of [63] present a 2-competitive algorithm. In [63] the authors explore an unknown polygonal terrain with obstacles by a single robot. They are looking for an algorithm whose complexity depends on the distance of the worst case path in order to see all the visible points of the environment and create a map.

The authors of [10] study the best way to search a possibly unbounded region for an object. The costs for this search algorithm's model is proportional to the distance of the next prob position relative to the current position. This model is meant to give a realistic cost measure for a robot moving in the plane. As a line may be also considered as one-dimensional geometric terrain the papers [10, 23, 22, 62], discussed previously, also fall in the scope of the present section.

The polygon exploration problem belongs to the category known in computational geometry as Art Gallery Problem. The typical problem in this domain is to minimize the number of guards or minimize the length of their trajectories such that at some point in time each point of the environment is seen by some guard. The case of a single mobile guard is studied in [82]. The authors in [82] present an on-line strategy that enables a mobile robot with vision to explore an unknown simple polygon. They prove that the resulting tour is less than 26.5 times as long as the shortest watchman tour that could be computed off-line.

A robot r looks for a target t in a unknown rectilinear polygon in [87], and the position of the target is unknown. The robot recognizes the target if it sees it, if the segment rt belongs to the polygon. The author has found a randomized algorithm to find the target in a minimum delay or a minimum traversal distance.

This algorithm achieves the competitive ration of 5/4.

In [52] a mobile robot explores an unknown terrain with obstacles that have the shapes of arbitrary polygons. The robot is represented by a point p and it must explore all points q, such that $distance(p, q) \leq 1$. For unlimited vision of the robot, the provided exploration algorithm is such that the length of the trajectory of the robot is $O(P + D\sqrt{k})$ where P is the perimeter of the terrain plus those of the obstacles, D is the diameter of the convex hull of the terrain and k is the number of obstacles. P, D and k are unknown by the robot. When the vision of the robot is bounded to a unit circle, the provided algorithm produces an exploration path of length $O(P + A + \sqrt{Ak})$, where A is the area of the terrain without the obstacles, and P and k are as described above.

In the case of the three-dimensional geometric environment, one can talk about a mobile robot that navigates there as a motion of an underwater vehicle, cf. [93]. Thanks to an algorithm based on simple geometric functions and on certain known curves, this vehicle can circulate by carrying a camera and report videos of its trajectory to realize a specific task such as to look for an airplane that fell in the bottom of the sea or a lake, to study the density of fish in a region, to transport explosive materials or realize other operations [93].

2.3.2 Exploration and Searching Problem by a Team of Mobile Robots

Exploration using a team of collaborating robots has as a main difficulty to ensure a good partition of the work between the members of the team. As the environment

is often considered a priori unknown, this task may cause substantial problems.

[78] considers the problem of exploration of an n-node tree by k mobile agents initially starting from the root. The goal is to explore the tree in minimal time. This is an NP-hard problem. In this paper they studied the problem of distributed collective exploration of unknown trees. The algorithm proposed in [78] explores any tree in time $O(D + n/\log k)$ where D is the diameter of the tree. Assuming that agents can communicate by leaving information at visited nodes, an exploration algorithm whose execution time for any tree is only $O(k/logk)$ larger than the optimal exploration time with a complete knowledge of the tree. They proved that in the case where the robots do not communicate with each other then each distributed exploration algorithm operates in time $\Omega(k)$ longer than the time of exploration knowingly optimal for some trees. In [81] it is shown that the strategy presented in [78] is precisely $\Theta(k/\log k)$.

The work of [65] proposes the first exploration algorithm of a graph which works in time $O(D)$, where D is the radius of the graph (i.e. the distance from the node where all agents are initially present to the farthest node). The number of agents used in [65] is polynomial in D and the size n of the graph. The agents do not need to know D and n.

A different, DFS-based algorithm was proposed in [29] where the proposed exploration algorithm works in time $O(n/k + D^{k-1})$. If the tree is sparse then the work of [70] proposes the strategy giving the competitive ratio of $O(D^{1-\frac{1}{p}})$. The parameter p is defined as the density of the tree in [70]. On the other hand, the best lower bound gives the competitive ratio of $\Omega(\log k/\log\log k)$. This bound, given in [71] holds for any deterministic algorithm such that $k < \sqrt{n}$. For so-called

class of greedy algorithms [81] proves a better lower bound of $\Omega(k/\log k)$.

However the results of [29, 70, 71, 81] are valid only for labeled graphs (i.e. each node has a different identifier). For unlabeled graphs the problem seems much harder, as agents cannot recognize whether, arriving at a graph node, this node was already visited in the past. The work [25] shows that having two agents, permits to explore and map any directed graph. Recall that, according to [24], a single agent is not sufficient to learn a graph unless it possesses some additional power, like ability to mark vertices. [24] proved that one pebble (or one bit of marking information) is sufficient to achieve mapping of a graph by a single agent. The extra agent used in [25] may simply be used exclusively for the marking purpose. If the size n of the graph is known in advance [25] propose a polynomial algorithm exploring and mapping an unknown graph of at most n nodes, using a constant number of pebbles.

An exploration is sometimes possible when the team of agents can exploit some additional knowledge of the environment. For example, in [73] the authors provide an exploration of an n-node ring by k identical, unconscious and asynchronous mobile robots, assuming that k and n are relatively prime numbers. These robots are able to see the environment but cannot communicate. Despite the difficulties the authors prove that an exploration of the ring described above is possible. They prove that the required time is $O(\log n)$ where n (large) is the size of the ring and that the minimum number of agents is estimated at $\rho(n) = \Omega(n)$. They prove that the problem is insoluble in the case where k divides n.

In [60] there are k identical asynchronous agents, initially located at different nodes of an unknown indirect simple graph. The agents may initially know either

the number n of the graph nodes, or the total number k of agents. The authors provide a deterministic algorithm with $O(km)$ complexity, where m is the number of edges in the graph, that constructs the labeled map of the graph.

[58] studies exploration of an unknown tree using a set of agents having limited energy. The height of the tree is bounded by the agents' energy limit B. The objective of [58] is to design an algorithm using as few agents as possible. The authors give an $O(\log B)$-competitive algorithm (comparing to the offline algorithm that knows the tree in advance).

The case of exploration of geometric environments by collection of mobile robots has been also studied. In the case of one-dimensional environment the problem is not difficult if the agents possess the knowledge of their initial positions. The author of [91] studies the optimal exploration of a line segment by two mobile agents having knowledge of their initial positions in the environment. Each point of the segment must be visited by at least one agent and the exploration time must be minimized. This time of the exploration is defined by the longer between the lengths of the trajectories of both agents or the time of the exploration of the "last" point of the line environment.

The work of [91] was extended in [90], where the cases of n mobile agents in the line and ring environments have been considered. The $O(n^2)$ algorithm computing trajectories of all n agents resulting in optimal exploration time was proposed in [90].

In the case of the two-dimensional plane [11] studies algorithms searching for a point or a line at unknown position but at a known distance from the origin. The

cases of one, two and many mobile searchers are investigated.

Some studied setting considered the case of two non-uniform robots, i.e. robots speeds that may have different maximal speeds. [46] and [12] consider exploration of one-dimensional environments, by n mobile robots initially placed at the same point of the environment. Each of these robots has a search speed s_i and a walking speed w_i where $w_i < s_i$. A search of the environment is completed when each of its points has been searched by at least one of the n robots. More exactly, one of the robots needed to visit such point in its searching "mode". The goal is to complete the search as quickly as possible knowing that each robot knows the speeds of the other ones. [46] considers the case of exploration of the segment, while [12] considered the same problem for ring environment. Two types of algorithms were studied in [12] and [46]: one for which the size of the environment to search is known in advance and the case where it is unknown. It has been proven in [46] that in the former case all mobile robots need to be used while in the latter one some robots must be left unused if the optimal worst-case exploration time must be obtained. Both algorithms are proven to be optimal in [12] and [46].

In some research papers the studied scenarios assume agents having different speeds (e.g. [21]). The search on a line environment using two mobile agents having different maximal speeds was studied in [36]. The search is completed when the second of the two searchers reaches the target (it is called *group search* in [36]). The agents communicate face-to-face (F2F), so the agent finding the target needs to walk to meet the other agent and then they both travel towards the target. Although no general algorithms were given in [36] some provided examples are interesting. The case of two agents performing group search on a line has been solved in [13]. The agents have different maximal speeds and two scenarios were

studied in [13]: the F2F communication and wireless communication. For each case a search schedule has been designed in [13] and its optimality has been proven.

Some papers consider exploration by teams of agents with some unknown "imperfections". In [54] the authors consider a problem of finding a target on a line by n mobile robots such that up to f of them may be defective ($f < n$). The position of the target is unknown by the robots. The robots are placed at the same starting point. Defective robots can not detect the target. Reliable robots can find the target when they reach its position. The authors found a parallel algorithm that minimizes the competitive ratio, represented by the worst case between the time of arrival of the first reliable robot on the target, and the distance from the source to the target. Some results are found that if $n \geqslant 2f + 2$, the algorithm is simple with a competitive ratio 1 and for any $f < n < 2f + 2$, the algorithm found is a proportional schedule algorithms $A(n, f)$, whose competitive ratio is

$$\left(\frac{4f+4}{n}\right)^{\frac{2f+2}{n}} \left(\frac{4f+4}{n} - 2\right)^{1-\frac{2f+2}{n}} + 1$$

[88] subsequently proved that the above competitive ratio is optimal for any n and f, such that $f < n < 2f + 2$.

A different situation is presented in [50] when the robots may experience byzantine faults. The byzantine robots mail fail to report the target or they may report it being found at wrong positions. The authors of [50] present specific algorithms for different ratio f/n of faulty robots. Some of these algorithms are proven optimal, i.e. finding the target in the shortest possible time assuming that the f faulty robots are chosen by an adversary.

Some of the papers on exploration by a team of robots seek to optimize agents energy. In [57] a tree is explored by mobile robots that hold a limited amount of energy B (the height of the tree is limited by the bound energy B). The robots are installed at the root of the tree. The goal is to exfoliate the tree with minimal number of agents and minimum energy expenditure in the cases, whether the agents know the environment or not. Without knowledge of the tree the cost is estimated at $O(\log B)$ independent of the number of nodes in the tree. The authors prove in [57] that it is the best possible competitive ratio for the exploration of unknown trees.

The robot power awareness was also studied in [69], where k mobile robots installed at the root of a tree explores all the leaves of the tree minimizing the distance traveled by each agent. An algorithm is developed for this NP-hard problem in the case where the tree is known by the agents, and an optimal algorithm is presented if k is constant. The authors show that for any number k their solution gives 2-approximation algorithm. They also prove that any such algorithm cannot be better than an 1.5-approximation.

The question of power awareness has been also studied in other context present in operation research (e.g. see [1]), especially in scheduling. The authors in [84] considered battery-based systems, so their interest is to save energy. They studied two mechanisms, the first is the standby state if it is inactive and the second is to vary the speed at which the tasks are executed. In the first case we need a quantity of energy to replenish the system and bring it back to the active state. In the second case they have found a function $P(s)$ which indicates the level of energy consumption given at a particular speed where $P(s)$ is convex, non decreasing and not negative for $s \geqslant 0$. The problem is to plan the arrival work in order

to minimize the total energy consumption and finish each job after its launch and before it expires. A 2-approximation algorithm is provided in [84].

2.4 Rendezvous and Gathering Problems

A rendezvous between two or more agents is very important in distributed computing. The meeting agents may then exchange information (e.g., individually collected by each agent during its work) or energy. In some cases, following a rendezvous between agents, some of them change their search and exploration opinions.

In [3] the rendezvous problem is considered a task that is dual to search - both agents collaborate in order to meet one another as soon as possible. The literature on rendezvous is very rich; we cite below only a small portion of chosen positions. The most recent survey on deterministic rendezvous may be found in [98].

The rendezvous (or gathering) problem may be considered as a special case of pattern formation problem (cf. [106, 75, 61]) where the set of robots need to finish their walks when forming a required pattern (in this case a point). One of the very first papers on rendezvous is [8], an algorithm with complexity $O(n)$ is presented to minimize the maximal length to visit all nodes of a tree by two traveling salesmen.

Among the first papers on asynchronous rendezvous in the domain of distributed computing [66] and [92] considered the graph environment. [66] studies rendezvous between two agents having distinct identifiers in an unknown, anony-

mous, connected graph under two scenarios, when both agents start executing the algorithm at the same time, and when starting times of the agents are arbitrarily decided by an adversary. In both cases the complexity is $O(n+\log l)$ on any n-node tree, and l is the smaller of the two identifiers. In the case of the simultaneous startup in the ring, the complexity is $\Theta(D \log l)$, where D is the initial distance between agents.

An asynchronous rendezvous studied in [92] occurs between two mobile robots with distinct tags located at nodes of an unknown connected graph. The rendezvous can be inside an edge of the graph, not necessarily at a node. If the agents are initially located at a distance D in an infinite line, the authors of [92] present an algorithm achieving a rendezvous with the cost $O(D|L_{min}|^2)$ when D is known in advance and $O((D+|L_{max}|)^3)$ if D is unknown, where $|L_{min}|$ and $|L_{max}|$ are the lengths of the shorter and longer label of the two agents, respectively.

A gathering of n mobile robots in the plane without any means of direct communication, operating in the (LCM) Look-Compute-Move cycle is studied in [37]. The venue of the meeting is not planned in advance. These robots are unconscious and fully asynchronous. Existing algorithmic contributions for such robots were previously limited to solutions for $n \leq 4$ or for restricted sets of initial robot configurations. The question of whether such weak robots could assemble deterministically remained open. In this article, the authors prove that indeed the gathering problem is solvable, for all $n > 2$ and any initial configuration.

Some papers considered the rendezvous problem when the agents have limited radius of visibility. In [76] the authors study the problem of gathering by a number of identical mobile agents in the same location of the plane. The case where the

visibility of the agents is unlimited was approached in the past, while for the case where the visibility is limited, the existing previous algorithmic results are only for the convergence (towards a common point, without ever reaching it) and only for the semi-synchronous environments, where robot motions are assumed instantly. The authors of [76] considered a totally asynchronous environment, where the actions, calculations and movements of robots require a finite but unpredictable time. They presented an algorithm that allows unconscious anonymous robots with limited visibility to congregate at the same place in a finite time, provided that they have an orientation (i.e. agreement on a coordinate system). The result indicate that, with respect to gathering, the common orientation is the very powerful knowledge that the team of robots may use.

In [99] the authors study the problem of meeting anonymous, totally asynchronous autonomous mobile robots in the 2-dimensional plane. The robots have no memory of the past calculations and they can not communicate explicitly between them. The robots act executing WAIT-LOOK-COMPUTE-MOVE cycle attempting to meet at any unplanned point of the plane. It is proven in [99] that such weak robots cannot meet in the plane.

[102] considered a deterministic rendezvous problem in general indirect graphs (introduced as the so-called "deterministic treasure hunt problem"). The authors of [102] used the powerful concept of Universal Traversal Sequences (see also [101]). The concepts of Universal Traversal Sequences has been also applied in [53].

A rendezvous problem considered in [53] between two identical, anonymous mobile agents is in an unknown connected graph. The same problem may be considered in an unknown terrain in the plan. A recent well-known result on explo-

ration, due to Reingold [101], indicates that deterministic exploration of arbitrary graphs can be performed in logarithmic space, i.e. by using an agent equipped with $O(\log n)$ memory bits, where n is the size of the graph. A deterministic algorithm has been provided in [53], which ensures the rendezvous between the two agents using $O(\log n)$ memory for each agent.

The asynchronous rendezvous in infinite plane has been considered in [56]. The authors prove that if the agents, placed at any two positions in the plane, have non-zero visibility radius they can always meet in finite time, despite the fact that their movements can be arbitrarily delayed by an adversary. The authors of [56] prove that the same result holds also in the case of an arbitrary, possibly infinite, graph. On the other hand, the authors of [68], again using the powerful concept of Universal Traversal Sequences proved that the above results from [56] may be obtained using polynomial-time algorithms.

It happens that the time of the rendezvous algorithm in the infinite plane may be substantially improved if the robot may use a GPS device. The authors of [40] study a rendezvous of two asynchronous agents with limited visibility in the plane (Euclidean 2-dim space). Each agent knows its own initial position in the plane given by its Cartesian coordinates, but it doesn't know about other agent position. Such agents are called *location-aware* in the literature. The algorithm complexity is measured by the sum of the lengths of the trajectories of both agents. The authors propose an algorithm depending on d being the original distance between the agents. The algorithm achieves the rendezvous in $O(d^{2+\epsilon})$ time. This result is almost optimal, as the $\Omega(d^2)$ lower bound is easy to prove.

The original work of [40] introduced the concept of the *central partition*. The concept may be adapted to work in other types of environment. A rendezvous problem studied in [41] concerns two anonymous robots travelling in some environment where each agent knows its own initial position. The goal is to design an algorithm achieving the rendezvous in minimum time in the worst case. The time is measured by the number of synchronous rounds that agents need to meet. In this paper the authors study two types of environments: finite or infinite graphs and Euclidean spaces. The authors found an asymptotically optimal rendezvous algorithm. For the line, the agents can rendezvous in time $O(d)$ where d is the distance between the initial positions of the agents. In general n-node graphs, the rendezvous can be achieved in $O(d \log^2 n)$ time.

Some papers consider rendezvous when the environment may be subject to some faults. [34] studies a rendezvous problem between two synchronous labeled agents at a node in a graph of size n using a deterministic algorithm. Agents do not know the graph size and they can incur delay fault; every agent knows its label but not the label of the other robot. If an agent incurs a fault in a given round, it remains in the current node, regardless of its decision. If it planned to move and the fault happened, the agent is aware of it. The authors consider three cases of fault distribution: random (independently in each round and for each agent with constant probability $0 < p < 1$); unbounded adversarial (the adversary can delay an agent for an arbitrary finite number of consecutive rounds); and bounded adversarial (the adversary can delay an agent for at most c consecutive rounds, where c is unknown to the agents). The performance of the rendezvous algorithm is measured by the number of edges traversed. For random faults, [34] proposes an algorithm with cost polynomial in the size n of the network and polylogarithmic in the larger label L, which achieves rendezvous with very high probability in

arbitrary networks. By contrast, for unbounded adversarial faults they show that a rendezvous is not possible, even in the class of rings. Under this scenario the authors of [34] give a rendezvous algorithm with cost $O(n\ell)$, where ℓ is the smaller label, working in arbitrary trees, and they show that $\Omega(\ell)$ is the lower bound on rendezvous cost, even for the two-node tree. For bounded adversarial faults, [34] gives a rendezvous algorithm working for arbitrary networks, with cost polynomial in n, and logarithmic in the bound c and in the larger label L.

Mobile agents that could experience Byzantine faults were considered in [27]. Every agent has its label (positive integer number) and the agent does not know the other agents' labels or their positions. Agents move in synchronous rounds and can communicate when they are at the same time in the same node. Among the agents we can have up to f of them that are Byzantine. A Byzantine agent can choose an arbitrary port when it moves, can convey arbitrary information to other agents and can change its label in every round, in particular by forging the label of another agent or by creating a completely new one. The authors of [27] prove that having $M = f+1$ reliable agents guarantees a deterministic gathering of all of them when the agents initially know the size of the network. If the size of the network is unknown it is shown in [27] that $M = f+2$ reliable agents guarantee the gathering.

Gathering in the presence of Byzantine agents was also studied in [67]. In this search two cases are being discussed: among the agents there are some who are strongly Byzantine and the other case in which the faulty agents are weakly Byzantine. A strongly Byzantine agent may choose an arbitrary port when moving and may transmit arbitrary information to other agents, including its incorrect label. On the other hand, a weakly Byzantine agent may do the same, except changing its label. The authors of [67] designed an algorithm that determines the number

of reliable agents that guarantee the gathering. In the weakly Byzantine case, any number of good agents solve the problem for networks of known size. If the size of the network is unknown, a deterministic polynomial algorithm is used that gather all the good agents placed in an arbitrary network, provided that there are at least $f + 2$ reliable agents. In the case of strongly Byzantine agents [67] provides a deterministic gathering algorithm for at least $2f + 1$ reliable agents when network size is known and for at least $4f + 2$ good agents when it is unknown.

In [38] the authors consider n robots whose sensors are not perfect, where motion and internal calculations may have small inaccuracies to perform the task of convergence. A distributed convergence control algorithm is presented under certain conditions, motions and computational errors.

2.5 Other Search-Type Problems for Teams of Mobile Agents

In this section we discuss variations of some other problems for teams of collaborating mobile agents.

2.5.1 Evacuation Problem

The evacuation of the agents' addresses an important scenario from everyday life such as saving people from a fire, or from dangerous or flooded places. The authors of [47] study the evacuation of a number k of mobile robots from a unitary disk through an exit door which is on the perimeter of the disk. They considered that the robots start at the center of the disk in both cases: wireless communication or

local communication ($F2F$), i.e., communication when the robots are at the same point. To complete the evacuation the time needed is at least

$$3 + \frac{2\pi}{k}$$

and a lower bound of

$$3 + \frac{2\pi}{k} - O(k^{-2})$$

is proven in [47] for the $F2F$ communication.

In the wireless communication case [47] proves the upper bound of

$$3 + \frac{\pi}{k} + O(k^{\frac{-4}{3}})$$

and the lower bound of

$$3 + \frac{\pi}{k}$$

.

In [45] the authors study evacuation of two robots from a domain that is represented by a unitary circle that admits k exit doors. These two robots can communicate on wireless mode. The robots do not know their initial positions but they know where the exit doors are and the distances between them. In other words, the robots possess the map of the environment with all the exits provided, but they do not know their own position on this map. The algorithm results in an evacuation (not necessarily through the same exit door for both robots) in minimal time in the worst case.

Some results of [45] has been later improved. In [51] the lower bound and the upper bound for the $F2F$ communication model were improved for two robots. The authors of [51] suggested a surprising algorithm in which the two robots are forced to meet at some fixed point, independently whether one of them already found an exit or not. The upper bound 5.628 of [51] has been improved to 5.625 in [28].

The same problem is considered under the wireless communication setting in [89] in the case when the robots have different speeds. The authors provided an optimal algorithm for the case when the fastest robot is at least 2.75 times faster than the other. For the case when robots speeds are not that different [89] provided lower and upper bounds.

2.5.2 Boundary Patrolling Problem

In the patrolling problem the environment must be regularly explored in a periodic basis. The efficiency of the patrolling algorithm is measured by the worst-case maximal period of time between two explorations of the same point.

[48] is the first paper on boundary patrolling by robots of different speeds. A boundary patrol problem by k mobile robots as discussed in [48] is to protect the border of an intruder who tries to enter the environment. The paper discusses the case which is known in the robotics literature as *fence patrolling*, i.e. the environment is homeomorphic with an open curve, and *boundary patrolling*, when the environment is homeomorphic with a Jordan curve. Two strategies are discussed: the *cyclic strategy* and the *partition strategy*. Each agent has its own predefined maximum speed and is able to exceed this limit without. The intruder needs a

certain time T to complete the intrusion. In both strategies, and in the case of $2, 3$ and 4 robots, an optimal result is found.

Some of the results of [48] have been subsequently approached in [85]. For example [85] disproved the conjecture stated in [48], that the partition strategy is optimal. The authors of [85] provided a counter-example involving 6 agents for which the partition strategy was not optimal.

The patrolling problem was also considered for the same-speed mobile agents. In [39] n mobile robots are distributed and must patrol a simple finite curve composed of a finite set of vital segments separated by neutral segments. The robots do not have to patrol neutral segments. The authors proved that either the partition strategy or cycling strategy leads to the optimal idle time if the robots are having the same bound on their maximal speed.

The problem of patrolling by k mobile robots with distinct visibility is discussed in [55]. It has been proven in [55] that it is an NP-hard problem if the environment is a general graph and if the robots may have distinct visibility radii. The same problem for point visibility robots (i.e. robot see only the point at which it is present) has an algorithm of polynomial complexity. Other algorithms are provided for fence patrolling and boundary patrolling, when all robots have the same maximum speed. In addition they show that the problem of patrolling by robots with distinct ranges of visibility is essentially different. The case of the fence patrolling by two robots with distinct maximal speeds and different ranges of visibility is discussed, and an algorithm is provided.

In [49] the authors discuss a patrolling algorithm of mobile robots that are deployed on a weighted graph, knowing that among them there are some that are unreliable. The goal is to minimize the time between successive visits of each edge point by a reliable robot. An optimal algorithm is provided in [49] for the case of fence patrolling.

2.5.3 Cops and Robbers Problem

In the classic search or exploration problem a mobile agent or a team of agents need to find a stationary target. The time of the deterministic search or exploration is often measured using a worst-case analysis, i.e. the adversary places the target at the portion of the environment which is the last to be searched.

The problem of pursuing a thief or malefactor (Robber) is a problem of exploring a terrain, a domain equipped with obstacles or a graph in order to catch this robber or robbers who are mobile and can hide in order to flee the police. The Cops and Robbers literature is primarily interested whether the assumed capacities of the searchers (Cops) and the topology of the environment are sufficient so that the capture of the robber is always possible. An interesting book [26] provides mathematical foundations of the game of Cops and Robbers.

The game is considered won by the Cops if there exists their motion so that eventually, in finite time, the Robber is captured by them. The Robber win if, knowing the Cops strategy, it can indefinitely continue to move while never being captured. Most papers on the Cops and Robbers game consider the case of the graph environment. the authors of [26] analyse several variations of the problem discussing the cases when the Robber or the Cops have a winning strategy.

[77] provides a survey of the graph-searching literature and a large collection of discussed results contains a moving target. In [79] the authors study the ques-

tion of the cop number, i.e. the minimum number of cops that are needed to catch the robber. For example it is shown in [79], that if the robber can move as fast as R edges per one unit of time, then there exist graphs needing $n^{1-\frac{1}{R-2}}$ cops moving at the speed of one edge per time unit so that the capture of the robber is possible. For the case of $R = 1$, [79] discusses the case of strongly connected directed graphs on n vertices. They prove that the cop number of such graphs is $O(n\frac{(\log\log n)^2}{\log n})$.

The cops and robbers literature is very vast (cf. [77]). The interested reader may consult [26] where numerous scenarios are discussed as well as the survey [77].

2.6 Data Delivery

The information dissemination has been discussed in the past in the model of message passing systems for interconnection networks (e.g., see [83]). The typical questions involve how to communicate in parallel between the processors, represented by the network nodes, while avoiding conflicts inside local communication channels, so that the communication protocol is realized in the smallest possible number of communication rounds. For example, message broadcasts from an arbitrary processor to all other ones was studied in [9] from the perspective of message complexity. The complexity depends on unbounded or bounded message length, knowledge of the network and synchronous and asynchronous network models. The typical communication protocols studied for the interconnection networks are *broadcast, convergecast* and *gossiping*. [83] analyzes numerous variants of the problems arising in this context. Some of these protocols are studied for wireless sensor networks, e.g. [7, 100] and ad-hoc networks [86].

One of the interesting problems for the interconnection networks communication is the advantage of using randomized algorithms. It happens that the use of

the randomization permits to construct algorithms fundamentally more efficient than the possibilities offered by deterministic algorithms. For example, in [18] the authors study deterministic and randomized protocols for achieving broadcast (distributing a message from a source to all other nodes) in arbitrary multi-hop radio networks. They show that a randomized algorithm may achieve broadcast using $O((D+\log n\epsilon)\log n)$ time-slots, where n is the number of processors in the network and D is its diameter. On the other hand, [18] proves a $\Theta(n)$ time-complexity in the deterministic broadcast protocol. This shows an exponential gap in complexity between the two scenarios.

In data delivery problems considered in this thesis the mobile agents transport packets of data between the nodes of weighted networks. The agents use energy proportionally to the distance traveled. The typical problems studied are related to different communication protocols:

1. One-to-one data delivery, where the data packet originally possessed by one network node needs to be transported to some other node.

2. Broadcast, in which the packet initially possessed by some network node must be delivered to all remaining nodes.

3. Convergcast, i.e. the data packets initially present at all nodes must be consolidated at some node.

4. Gossiping, where each network node needs to collect the data packets initially present at all other nodes.

For each communication protocol the problems studied are:

1. The feasibility of the problem, i.e. whether an algorithm exists if the corresponding communication protocol may be realised for the initial distribution of agents in the network.

2. Realization of the communication protocol using the minimal possible energy.

3. Realization of the communication protocol in the minimal possible time.

In the typical problem of data delivery a set of k agents is distributed in some nodes of an n-node graph, each agent having some initial amount of energy.

The data delivery problem was initiated in [5], where the authors study the problem of convergecast by a set of mobile agents distributed on a line. Each agent has the same amount of energy. The $O(n)$ algorithm given in [5] computes the minimal amount of energy needed by each agent so that convergecast on a line may be successfully achieved. The authors of [5] prove that the same problem on a tree network is NP-hard, and they give a 2-approximation distributed algorithm. For general graph networks [5] gives a centralized 2-approximation algorithm.

Four years later the full version of the above work appeared in [6], where the problem of broadcast was also studied, which also proved to be NP-hard. The authors gave a 4-approximation distributed algorithm for broadcast. Moreover, it was proven as well that the constant of 2 is the best possible for distributed convergecast as for no $\epsilon > 0$ there exists a $(2 - \epsilon)$-approximation distributed convergecast algorithm.

The problem of one-to-one data delivery on the line was studied in [35] for agents having not necessarily the same amount of energy. Surprisingly this simple problem was proved to be NP-hard. For instances where all input values N

are integers is quasi-pseudo-polynomial.The algorithm for solving this problem has a complexity of $O(\Delta^2 N^{1+4\log\Delta})$ where Δ is the distance between the points s and t.

The data delivery problem was the subject of the PhD thesis by Andreas Bärtschi (see[15]). In this thesis k mobile agents located in distinct nodes in a indirect graph cooperate to move packages in the graph. Every agent i has a velocity v_i and a weight ω_i. The goal is to deliver all packages, minimize the total energy consumption of all agents and respect the constrained resources of the agents. The thesis studied various scenarios of the problem under the assumption that agents may have bounded capacities, i.e. each agent could carry a limited number of data packets.

In [17] the authors considered agents data delivery of several packets, each one between a specific source and target nodes of a given graph. The agents of [17] are heterogeneous as each of them may consume energy at a different rate. The objective of [17] was to minimize the total energy needed to be spent by the team of agents. When the agents capacities were limited to one data packet, the problem was proved to be NP-hard, even for a single agent. However a polynomial time 2-approximation is proposed in [17]. When the energy-consumption rates are private, i.e. each one is known only by a corresponding agent, [17] proposes a solution resulting in a constant approximation of a total energy spent.

Bärtschi, [15], considered also a time-efficient delivery (see also [19]). The authors of [19] proved that the problem is NP-hard even for planar graphs and NP-hard to approximate in general. However [19] gives a polynomial time algorithm for delivery of a single package.

The papers [19] and [20] considered bicreteria-efficient delivery, in which the authors attempt to minimize the energy consumption E and simultaneously the delivery time T. Clearly, minimizing one of these parameters may be performed at the expense of the other one. In [20] the authors propose a polynomial-time dynamic program that finds the behaviour of the agents that lexicographical minimizes (E, T). However the result of [19] states that when the priorities are reversed, i.e. when (T, E) must be minimized, the problem becomes NP-hard.

The problem studied in [16] concerns the scenario similar to [5] where a single packet must be delivered to the target position while agent budgets (i.e. initial energy levels) must be respected. The authors of [16] analyze the Non-Returning Delivery problem, when the agents may terminate their walk anywhere and the Returning Delivery, when the agents are required to return to their initial positions. While the non-returning version of delivery has been previously proven in [35] to be NP-hard for lines, [16] proposes a polynomial-time solution for tree networks. However for the class of planar graphs the Returning Delivery is proven to be NP-hard in [16]. Because of the hardness of this problem, the authors of [16] consider the question of resource-augmentation: by what factor must the energy level of all robots be augmented, so that if the delivery problem was feasible for the original energy levels, there is a known algorithm for the resource-augmented case. [16] gives tight lower bounds for the resource augmentation necessary for the returning and non-returning versions of the delivery problem.

Closely related to the data delivery problem is the tree exploration by a collection of mobile agents investigated in [57]. Each agent of [57] is placed at the same initial node and it has the same amount B of energy, that is known to be sufficient to reach any leaf of the tree. The tree is a priori unknown to the agents,

but they can exchange partially acquired information while they are collocated at a same node of the tree. The objective of [57] is to minimize the number of agents used for exploration. The exploration algorithm of [57] has a competitive ratio of $O(\log B)$, compared to the best offline algorithm having the full knowledge of the tree. This competitive ratio is proven to be the best possible.

When the tree is known to the agents, the exploration investigated in [57] is done more efficiently in [59]. The authors of [59] give an algorithm which covers a given tree with the minimal number of routes starting and ending at the same node (*root*) of the tree.

The only other paper investigating data delivery by energy-exchanging agents is [14]. The agents are placed at given nodes of the input graph and each of them has two units of energy. The objective is to verify whether the delivery from a given source node s to a given target node t is possible. The authors prove that this problem is NP-complete. However, consider the following problem: ″Given a subset H of nodes of the graph and an integer k, is it possible to place k agents in some nodes of H so that the delivery is possible?″ The authors of [14] prove that this problem has a polynomial time solution.

Chapter 3

Summary of results

3.1 Preliminaries

We are given a tree T containing n nodes $v_1, v_2, \ldots, v_n$. The tree is edge-weighted i.e. each edge (v_i, v_j) has a weight $w(i,j) \geqslant 0$.

We assume that if nodes v_i and v_j are not adjacent, then we extend the weight function so that $w(i,j) = \infty$.

Therefore we can say that $w : V \times V \longrightarrow Z^+ \cup \{\infty\}$.

At selected nodes of the tree are placed k mobile agents denoted by the integers $0, 1, \ldots, k-1$. Each node i, for $0 \leqslant i < k$ has an initial amount of energy e_i being a non-negative real number. The mobile agents can walk along the edges of the tree while spending energy proportionally to the distance traveled.

Some nodes of the tree initially contain the data packets.

During the algorithm, a subset of data packets may be carried by an agent as well as being held by a tree node. We assume that node and agent capacities are unlimited, i.e. that each agent and each tree node may be in possession of any number of data packets.

Suppose that at some time moment agent i carries a set of packets P_i, a node v possesses a set of packets P_v and agent i is about to visit node v. Then, when agent i leaves node v it departs counting the set of packets $P_i \cup P_v$ and the same set of packets remains in possession of node v.

Similarly, consider a meeting of agents i and j, respectively possessing sets of packets P_i and P_j, before the meeting. We assume that after the meeting both agents continue their walks possessing the set of packets $P_i \cup P_j$.

The agents may meet when arriving at the same time moment to the same point of the network. We assume that at a moment of meeting the agents may exchange any amount of currently possessed energy.

3.2 The Problems

The basic problems related to energy-exchanging mobile agents concern different protocols of communication of data packets. In each problem is given a tree T and a collection of k mobile agents placed at same nodes of T, each agent having some non-negative initial energy level.

Data Delivery Problem: Let be given a source node s and a target node t. Decide if the initial configuration of the agent positions and their energy levels permit to construct a schedule of agents movements and energy transfers between meeting agents, which result in the initial data packet of source s being delivered to the target node t.

Convergecast Problem (to a given node): Let t be a given node of T. Suppose that every node of T initially possesses a data packet. Decide if the initial configuration of the agent positions and their energy levels permit to construct a schedule of agents movements and energy transfers between meeting agents, such all data packets being delivered to the target node t.

Convergecast Problem (to an unspecified node): Suppose that every node of T initially possesses a data packet. Decide it there exists some node $t \in$ T such that the initial configuration of the agent positions and their energy levels permit to construct a schedule of agents movements and energy transfers between meeting agents, which result in all data packets being delivered to the target node t.

Broadcast Problem: Suppose that a given node $s \in$ T contains a data packet. Decide if the initial configuration of the agent positions and their energy levels permit to construct a schedule of agents movements and energy transfers between meeting agents, which result in the initial data packet of s reaching every node of T.

3.3 The Results Obtained

In Chapter 4 we present first an algorithm solving the Data Delivery Problem for line networks. Then we observe that our solution can be extended on tree networks. We also solve the Convergecast Problem for tree networks. The algorithm works for the convergecast to a specific tree node. However, it may be modified in order to function when the target node is not specified. All algorithms from this chapter work in an optimal $O(n)$ time (where it is assumed that $k \leqslant n$). However, we show that for general graph networks the Data Delivery Problem is NP-hard.

The problem of broadcasts, when agents start at the same initial position, is studied in Chapter 5. We consider first the case when the source, which is the broadcast node, is the starting node for each agent. We give an $O(n \log n)$ algorithm solving the Broadcast Problem for this case. We also propose an $O(n)$ algorithm for this setting when the number k of available agents is at least equal to the number of leaves of tree T. Finally, we show that our algorithms may be extended to the case when the source node s is different from the starting position of all agents.

The problem of broadcast is continued in Chapter 6, where each agent is initially placed at arbitrary node of the network. This solves the Broadcast Problem in its general setting. However, as the setting of this chapter assumes arbitrary distribution, the time complexity of the proposed algorithm is $O(nk^2)$, that is much larger than $O(n)$ and $O(n \log n)$ that we obtained in Chapter 5 for agents starting together.

Our approach in each chapter permits not only to solve the decision problem whether the analyzed communication protocol is feasible, but also finds the largest amount of energy that may be saved while realizing the protocol. Such an optimisation problem (that is more general that its decision version) is a byproduct of the chosen methodology, rather than the goal by itself.

Chapter 4

Data Delivery and Convergecast

We consider two problems:

1. ***Data delivery* problem:** Given two nodes s, t of G, is it possible to transfer the initial packet of information placed at node s to node t?

2. ***Convergecast* problem:** Is it possible to transfer the initial information of all nodes to the same node?

We will look for schedules of agent movements which will not only result in completing the desired task, but also attempt to maximize the unused energy. We call such schedules *optimal*.

We conservatively suppose that, whenever two agents meet, they automatically exchange the entire collections of data packets that they currently hold. This information exchange procedure is never explicitly mentioned in our algorithms, supposing, by default, that it always takes place when a meeting occurs.

We consider first the line environment and then we extend our results for trees. We show that both communication have linear time algorithms on trees. On the other hand, for general undirected and directed graphs we show that these problems are NP-complete.

4.1 The Line Environment

In this section we start with a cartesian line environnement and suppose that we are given a collection of k agents, each agent i having initial energy e_i.

4.1.1 Data Delivery

On the line are given two points s (source) and t (target) and the agents need to collaborate in order to carry the data packet initially placed at point s to the target position t. We suppose $s < t$.

Observe that, without a loss of generality, we may replace many agents starting at the same point by a single agent whose initial energy equals to the sum of the amounts of all agents present at this point.
Note also limited use of the agents initially present outside the interval $[s, t]$ of the line. Indeed the agents on the left-hand side of s (starting from the leftmost one) walk left-to-right collecting energy of the encountered other agents. If some energy can be brought this way to s, we obtain an extra agent which will start at s. Symmetrically, the agents on the right-hand side of t act in order to possibly bring the maximal amount of energy to point t. It is easy to see that this is the best use of agents placed outside the interval $[s, t]$. Consequently, we may assume $s \leq a_1,\ a_k \leq t$.

Our first algorithm is only a decision version. Its main purpose is to show how certain useful table can be computed; all subsequent algorithms are based on computing similar type of tables.

Consider the partial delivery problem $\mathcal{D}_i$, in which agents larger than i are removed, together with their energy, and the goal is to deliver the packet from point a_1 to point a_i. We say that the problem $\mathcal{D}_i$ is *solvable* iff such a delivery is possible.

We define the following table $\overrightarrow{\Delta}$:

- If $\mathcal{D}_i$ is not solvable then $\overrightarrow{\Delta}_i = -\delta$, where δ is the *minimal* energy which needs to be added to e_i (to the energy of i-th agent) to make $\mathcal{D}_i$ solvable.
- If $\mathcal{D}_i$ is solvable then $\overrightarrow{\Delta}_i$ is the *maximal* unused energy which can remain in point a_i after delivering the packet from a_1 to a_i. Note that it is possible that $\overrightarrow{\Delta}_i > e_i$ since during delivery the unused energy of some other agents can be moved to point a_i.

Assume that points s and t are the starting points $s = a_0$ and $t = a_{k+1}$ of virtual dummy agents 0 and $k+1$, respectively. Each virtual agent 0 and $k+1$ has zero initial energy (or a positive energy if the agents initially placed left to s or right to t, could having such energy to s or t, respectively). Therefore, we may assume that the original positions of the agents are $s = a_0 \leq a_1 < a_2 < a_3 < \ldots < a_k \leq t = a_{k+1}$.

We have the following decision algorithm.

ALGORITHM DELIVERY-TEST-ON-THE-LINE ;

1. $A := e_0 = 0; a_0 := s; a_{k+1} := t; e_{k+1} := 0;$
2. **for** $i = 1$ **to** $k + 1$ **do**
3. $\quad d := a_i - a_{i-1};$
4. $\quad$ **if** $A \geq d$ **then** $A := A - d$
5. $\quad$ **else if** $A \geq 0$ **then** $A := -2(d - A)$
6. $\quad\quad$ **else** $A := A - 2d;$
7. $\quad A := A + e_i;\ \overrightarrow{\Delta}_i := A;$
8. **return** $(A \geq 0)$;

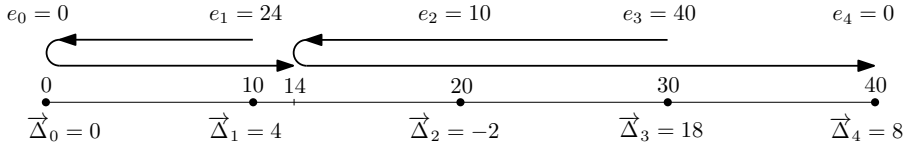

Figure 4.1: Schedule of agent movements for a_i's and energies given in Example 1.

Example 1. *Assume* $[a_0, a_1, \ldots, a_4] = [0, 10, 20, 30, 40, 50], [e_0, e_1, \ldots e_4] = [0, 24, 10, 40, 0]$. *Then (assuming, by convention,* $\overrightarrow{\Delta}_0 = 0$*, see also Figure 4.1) we have* $\overrightarrow{\Delta} = [0, 4, -2, 18, 8]$.

Remark. The values of $\overrightarrow{\Delta}_i$ are not needed to solve the decision-only version. However they will be useful in creating the delivery schedule and also in the *convergecast* problem.

Lemma 1. *The algorithm* DELIVERY-TEST-ON-THE-LINE *correctly computes the table* $\overrightarrow{\Delta}$ *(thus it solves the decision version of the delivery problem) in linear time.*

We prove by induction on i, that the value of $\overrightarrow{\Delta}_i$ is correctly computed in line 7 of the algorithm.

Suppose first the case $i = 1$. In the case $a_0 = a_1$, as $A = e_0 = 0$, in lines 4 and 7 we compute the value of $A = \overrightarrow{\Delta}_1 = e_1$, which is correct as agent 1 does not need to use any energy to pick up the packet at point a_0. Otherwise, if $a_0 < a_1$ we have $A = 0$ and $A < d$, so lines 5 and 7 are executed, in which case we have $A = \overrightarrow{\Delta}_1 = e_1 - 2d$. As agent 1 needs to cover distance d in both directions to bring the packet to point a_1 this is correct, independently whether the computed value negative or not.

Suppose now, by inductive hypothesis, that the algorithm computed correctly $A = \overrightarrow{\Delta}_{i-1}$ in the previous iteration. There are three cases:

Case 1 (line 4 of the algorithm). The instance $\mathcal{D}_{i-1}$ was solvable and after moving the packet from a_1 to a_{i-1} the maximal remaining energy was $\overrightarrow{\Delta}_{i-1}$. As in this case we have $\overrightarrow{\Delta}_{i-1} = A \geq d$, the energy $\overrightarrow{\Delta}_{i-1}$ is sufficient to move the packet from a_{i-1} to a_i. Consequently, we spent d energy to travers the distance d in one direction and we have $\overrightarrow{\Delta}_i = \overrightarrow{\Delta}_{i-1} - d + e_i$ as correctly computed in lines 4 and 7.

Case 2 (line 5). The instance $\mathcal{D}_{i-1}$ was still solvable but after moving the packet from a_1 to a_{i-1} the remaining energy $\overrightarrow{\Delta}_{i-1}$ is not sufficient to reach a_i without *help* from agents to the right of a_{i-1}. Then the $(i-1)$-st agent moves only one-way by distance $\overrightarrow{\Delta}_{i-1}$. The remaining distance $d - \overrightarrow{\Delta}_{i-1}$ to point a_i should be covered both-ways from a_i. Hence we need to use the amount of $2(d - \overrightarrow{\Delta}_{i-1})$ energy, which is expressed by statement 5. The value of $\overrightarrow{\Delta}_i$ is computed correctly independently whether the addition of e_i makes it positive or not.

Case 3 (line 6). In this case the instance $\mathcal{D}_{i-1}$ was not solvable, i.e. the agents $1, 2, \ldots, i-1$ could not deliver the packet to point a_{i-1}. Consequently, the

interval $[a_{i-1}, a_i]$ has to be traversed entirely in both direction and we obtain $\overrightarrow{\Delta}_i = \overrightarrow{\Delta}_{i-1} - 2d + e_i$, which is correctly computed in lines 6 and 7.

The cases correspond to the statements in the algorithm, and show its correctness. This completes the proof.

Once the values of $\overrightarrow{\Delta}_i$ are computed, the schedule describing the behaviour of each agent is implicitly obvious, but we give it below for reference. Note that the action of each agent a_i is started once the process involving lower-numbered agents has been completed. We are not interested in this chapter in finding the shortest time to complete the schedule (allowing agents to work in parallel).

ALGORITHM DELIVERY-SCHEDULE-ON-THE-LINE;

{ Delivering packet from s to t }

$pos := s$;

for $i = 1$ to k **do**

if $\overrightarrow{\Delta}_i \geq 0$ and $pos < a_i$ **then**

1. The i-th agent walks left collecting energy of all encountered agents until arriving at the packet position. It picks up the packet.
2. The i-th agent walks right collecting energy of all encountered agents until exhausting its energy or reaching t.
3. The i-th agent leaves the packet at the actual position pos.

Delivery is successful iff $pos = t$;

Figure 4.1 illustrates the execution of the above algorithm for Example 1.

We conclude with the following theorem.

Theorem 1. *The algorithm* DELIVERY-SCHEDULE-ON-THE-LINE *decides in* $O(n)$ *time, where* n *is the number of agents, whether the information of any agent can*

be delivered to any other agent and, if it is possible, it produces the centralized schedule which performs such a delivery.

4.1.2 Convergecast

The convergecast consists in communication in which the union of initial information of all nodes arrives to the same node. In some other papers (e.g. [5]), the convergecast problem consists in determining whether the union of the entire information may be transferred to the same agent. However, for energy exchanging agents, this is not a problem: if convergecast is possible then any agent may be its target, as agents may swap freely when meeting.

We present below the algorithm finding if convergecast is possible. We will use algorithm DELIVERY-TEST-ON-THE-LINE to compute the values of $\overrightarrow{\Delta}_i$ as defined before, assuming that $s = a_1$ and $t = a_k$. Similarly we denote by $\overleftarrow{\Delta}_i$ the values of the energy potential at point a_i that the symmetric algorithm would compute while transferring the packet initially situated at the point a_k towards the target position at a_i. Therefore, $\overleftarrow{\Delta}_i$ equals the deficit or the surplus of energy during the transfer of information initially held by agent n to agent i using agents $i, i+1, \cdots, k$.

ALGORITHM CONVERGECAST-ON-THE-LINE;

1. **for** all $i = 1, 2, \cdots, n$ compute the values of $\overrightarrow{\Delta}_i$ and $\overleftarrow{\Delta}_i$ representing the energy potentials at a_i, for deliveries from a_1 to a_i and a_n to a_i, respectively
2. **for** $i = 1$ to n **do**
3. **if** $\overrightarrow{\Delta}_i \geq 0 \wedge \overleftarrow{\Delta}_{i+1} \geq 0 \wedge \overrightarrow{\Delta}_i + \overleftarrow{\Delta}_{i+1} - (a_{i+1} - a_i) \geq 0$ **then**
4. **return** Convergecast possible;
5. **return** Convergecast not possible;

We have the following theorem.

Theorem 2. *Algorithm* CONVERGECAST-ON-THE-LINE *in $O(n)$ time solves the convergecast problem.*

Proof. The convergecast is possible if and only if the information of agent a_1 and the information of agent a_n may be transferred to the same point of the line. This is equivalent to the existence of a pair of agents i and $i+1$, such that transferring the information from point a_1 to a_i using agents $1, 2 \cdots, i$ results in a surplus of energy brought to point a_i, as well as that transferring the information from point a_n to a_{i+1} using agents $n, n-1 \cdots, i+1$ results in a surplus of energy brought to point a_{i+1}. Moreover, the sum of these two surpluses of energy must be sufficient to complete a walk along the entire segment $[a_i, a_{i+1}]$ permitting agents i and $i+1$ to meet. This is exactly what is verified at line 3 of algorithm CONVERGECAST-ON-THE-LINE. □

An interested reader may observe, that the condition of the if clause from line 3 may be simplified to $\overrightarrow{\Delta}_i + \overleftarrow{\Delta}_{i+1} - (a_{i+1} - a_i) \geq 0$ as in such case the convergecast is also possible although the convergecast point may not be inside the interval $[a_i, a_{i+1}]$. However, the current condition at line 3 permits to identify all points of the environment to which the union of all node information may be transported. We call such points *convergecast points*. Indeed, if $\overrightarrow{\Delta}_i + \overleftarrow{\Delta}_{i+1} - (a_{i+1} - a_i) = 0$, then there exists a unique convergecast point inside the interval $[a_i, a_{i+1}]$. The surplus of energy permits to deliver the convergecast information to an interval of the line larger than a single point. We have the following Corollary.

Corollary 1. *If the condition in line 3 of algorithm* CONVERGECAST-ON-THE-LINE *is true, then the set of convergecast points of the line equals* $[a_{i+1} - \overleftarrow{\Delta}_{i+1}, a_i + \overrightarrow{\Delta}_i]$.

4.2 The Tree Environment

In this section we observe that our solutions for lines may be adapted so they work for a more interesting tree environment.

4.2.1 Data Delivery

The technique developed for delivery in lines can be extended easily to delivery in undirected trees. In this case, the agents are placed at the nodes of the tree. Observe that from the original tree we can remove subtrees which do not contain s, t or any agents. Consequently, we obtain a connected tree whose every leaf either contains s or t or an initial position of some agent.

The delivery problem for a tree is easily reducible to the case of a line.

Theorem 3. *We can solve delivery problem and construct delivery-scenario on the tree in linear time.*

Proof. Consider the path π in the tree T connecting s with t. Suppose we remove from T all edges of path π. The tree splits into several subtrees *anchored* at nodes of π. For each such subtree we direct all edges towards the root, which is a node of π. The agents initially present at the leaves of such trees are walking up along the directed paths towards their roots accumulating energies at intermediate nodes. To avoid having two agents walking along the same edge it is sufficient to move agents present at leaves only and remove every such edge after the move is made. Agents having energy use it during their walk bringing the remainder to the intermediate nodes. Agents with zero energy are moved freely bringing no energy. The process terminates when the subtree is reduced to a single root belonging to path pi. This way we optimize the energy that can be brought to path pi. The problem of the delivery on the tree is now reduced to the delivery on the line π. Consequently,

all steps of this construction may be computed in linear time. This completes the proof. □

4.2.2 Convergecast

In this section we extend to the case of trees the basic ideas developed for the problem of convergecast for the line environment. The tables $\overleftarrow{\Delta}$ and $\overrightarrow{\Delta}$ for lines were computed locally, looking only at neighboring nodes. Simiarly, the values of the corresponding table $\overrightarrow{\Delta}$ for a node in a tree is computed looking at the neighbors of this node. However, as the flow of the information passing through node v can be made in d_v directions, where d_v is the degree of v, for each node v we will compute d_v different values of Δ . For this purpose, though the input tree is undirected, we will consider direction of edges. For each undirected edge (u, v) we consider two directed edges $u \to v,\ v \to u$. We define the subtree $T_{v\to u}$ as the connected component containing v and resulting by removing from T the edge (v, u), see Figure 4.2. Observe that at the moment of convergecast, there are two agents meeting at a point of some edge, that we call *convergecast point*, where these agents start possessing the initial information of all nodes.

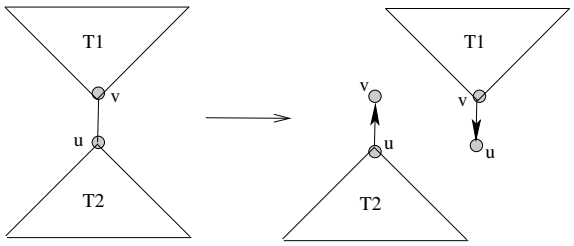

Figure 4.2: Testing if there is a *convergecast point* on the undirected edge (u, v) is reduced to computation of the costs $\Delta_{u\to v}$ and $\Delta_{v\to u}$ of moving all packets in the trees $T2 = T_{u\to v}$ and $T1 = T_{v\to u}$.

In order to compute all needed values of Δ, for each directed edge $u \to v$ of the tree we define $\Delta_{u\to v}$ as the energy potential of moving all packets from the

subtree $T_{u\to v}$ to its root u without interacting with any node outside $T_{u\to v}$. More exactly, if $\Delta_{u\to v} \geq 0$, then it represents the maximal amount of energy that can be delivered to u, together with all data packets originated at the nodes of $T_{u\to v}$. Observe that, if $T_{u\to v}$ initially does not contain any agents, then $\Delta_{u\to v}$ equals twice the sum of weights of all edges of $T_{u\to v}$. Indeed, in such case, the delivery must be performed by an agent starting at u and performing the DFS traversal of $T_{u\to v}$. If $T_{u\to v}$ initially contains some agents, the value of $\Delta_{u\to v}$ is smaller, but always equal at least the sum of weights of its edges. If $\Delta_{u\to v} < 0$ then $-\Delta_{u\to v}$ is the minimal amount of energy that we need to deliver to u by some agent, initially outside $T_{u\to v}$, so that this agent can bring to node u all data packets from the nodes of $T_{u\to v}$. In both cases, will be used all agents initially present inside $T_{u\to v}$ as well as their entire energy.

In order to correctly compute the values of Δ we define an order in which the consecutive directed edges of T will be treated by our algorithm. We denote $x \to y \prec y \to z$, when $x \neq z$, meaning that, for consecutive edges, an edge ending at a node *precedes* (according to relation $\prec$) an edge starting at this node.

Observation 1. *The relation $\prec$ in a tree is a partial order and it can be extended to a linear order X in $O(n)$ time.*

We propose the following algorithm.

ALGORITHM CONVERGECAST-ON-THE-TREE(T);

1. Compute a linear order X of directed edges of T according to relation $\prec$.
2. **for** each directed edge $u \to v$ taken in order X **do**
3. COMPUTE $\Delta_{u\to v}$;
4. **for** each undirected edge (u, v) of T **do**
5. **if** $(\Delta_{u\to v} \geq 0) \wedge (\Delta_{v\to u} \geq 0) \wedge (\Delta_{u\to v} + \Delta_{v\to u} \geq weight(u, v))$
6. **then return** Convergecast is possible
7. **return** Convergecast is not possible

The values of $\Delta_{u\to v}$ are computed by the following procedure.

PROCEDURE COMPUTE $\Delta_{u\to v}$;

1. $\Delta_{u\to v} := e_u$; {initial energy of node u}
2. **for** each indirect edge $x \to u$, such that $x \neq v$ **do**
3. **if** $\Delta_{x\to u} \geq weight(x, u)$
4. **then** $\Delta_{u\to v} := \Delta_{u\to v} + \Delta_{x\to u} - weight(x, u)$
5. **else if** $\Delta_{x\to u} > 0$
6. **then** $\Delta_{u\to v} := \Delta_{u\to v} + 2 * (\Delta_{x\to u} - weight(x, u))$
7. **else** $\Delta_{u\to v} := \Delta_{u\to v} + \Delta_{x\to u} - 2 * weight(x, u)$

We have the following theorem:

Theorem 4. *Algorithm* CONVERGECAST-ON-THE-TREE *in linear time solves the convergecast problem for trees.*

Proof. We show first the following claim:

Claim:, The for loop from line 2 of the algorithm correctly computes the value of $\Delta_{u\to v}$ for every directed edge $u \to v$.

The proof of the claim goes by induction on the consecutive iterations of the for-loop from line 2. Consider the first directed edge $u \to v$ of X. As the tree $T_{u\to v}$ is then composed of a single node, we obtain correctly $\Delta_{u\to v} = e_u$, i.e. the initial energy of node u. The claim is also true for any other edge $u \to v$, treated later by the for loop of line 2, such that u is a terminal node.

Consider now the case when the for loop from line 2 takes an edge $u \to v$ for a non-terminal node u. Let $v_1, v_2, \cdots, v_p$ be all nodes adjacent to u, such that $v_i \neq v$, for $i = 1, \cdots, p$. Note that, at that moment, the values of $\Delta_{v_i\to u}$ for all $i = 1, \cdots, p$ have been already computed. Observe that, bringing packets from all $v_i \neq v$, $i = 1, \cdots, p$ to u needs to be done across the respective edges $v_i \to u$, sometimes bringing the unused energy to u and other times using some energy from $\Delta_{u\to v}$ to traverse twice edge (v_i, u), or its portion, by an agent coming from u.

Take any v_i and suppose first that $\Delta_{v_i\to u} \geq weight(v_i, u)$. Then by inductive assumption, the agents present at $T_{v_i\to u}$ can perform the convergecast to v_i bringing there the amount of $\Delta_{v_i\to u}$ extra energy. This energy is sufficient to transfer all packets of $T_{v_i\to u}$ through edge (v_i, u) and the remaining amount of $\Delta_{v_i\to u} - weight(v_i, u)$ energy is accumulated at $\Delta_{u\to v}$, which is correctly computed at line 4 of procedure COMPUTE $\Delta_{u\to v}$.

Suppose now, that $\Delta_{v_i\to u} \leq 0$. In order to bring all packets of $T_{v_i\to u}$ to u, an agent present at u must traverse the edge $u \to v_i$, bring the packets to node using $-\Delta_{v_i\to u}$ extra energy and then traverse the edge (u, v_i) in the opposite direction $v_i \to u$. For this purpose is needed the extra energy of $-\Delta_{x\to u} + 2 * weight(x, u)$, which is correctly suppressed from $\Delta_{u\to v}$ at line 7 of the procedure.

Consider now the remaining case when $0 < \Delta_{v_i\to u} < weight(v_i, u)$. In this case, all packets of $T_{v_i\to u}$ are brought to node v_i by some agent initially present within $T_{v_i\to u}$, but this agent does not have enough energy to traverse edge $v_i \to u$ by itself.

Such agent will use its entire energy of $\Delta_{v_i \to u}$ to traverse a portion of edge $v_i \to u$ and some other agent need to come from u and to traverse the other portion in both directions in order to transfer the packets to u. The energy needed by the second agent equals $2 * (weight(v_i, u) - \Delta_{v_i \to u})$, which is correctly suppressed from $\Delta_{u \to v}$ at line 6 of the procedure. This completes the proof of the claim.

To complete the proof, consider the moment when in an optimal convergecast algorithm one agent obtains the union of the initial information of all nodes of the network. This happens while two agents meet on some edge (u, v), one of them carrying the union of information from the subtree $T_{u \to v}$, and the other one - from the subtree $T_{v \to u}$. These agents need to have enough positive energy to meet within the edge (u, v). This is equivalent to the condition tested in line 5 of the algorithm. □

The condition from line 5 of algorithm Convergecast-on-the-Tree permits to decide only if the convergecast is possible. However, similarly to the line case, an interested reader may observe that one can easily identify the set of all convergecast points. For this purpose we define the set of $D_{u,v}(d)$ containing a subset of points from the edges of T. Consider a point p and the simple path $\Pi(u, p)$ of T joining p with u. We define $p \notin D_{u,v}(d)$ if the path $\Pi(u, p)$ goes in the direction of edge (u, v) and its length exceeds d, i.e. $|\Pi(u, p)| > d$. All other points of T belong to $D_{u,v}(d)$. We have the following corollary.

Corollary 2. *If the condition in line 5 of algorithm* Convergecast-on-the-Tree *is true, then the set of all convergecast points of the tree equals* $D_{u,v}(\Delta_{u \to v}) \cap D_{v,u}(\Delta_{v \to u})$.

4.3 NP-Completeness for Digraphs and Graphs

We use the following NP-complete problem:

Integer Set Partition (ISP): Given set $X = \{x_1, x_2, \ldots, x_n\}$ of positive integer values verify whether X can be partitioned into two subsets with equal total sums of values.

We have the following theorem.

Theorem 5. *The delivery and convergecast problems are NP-complete for general directed graphs.*

Proof. Denote $E = \sum_{i=1}^{n} x_i$. Given an instance of the ISP problem, we construct the following graph G_X (see Figure 4.3). The set of $n+3$ nodes of G_X consists of three nodes s, t, a and the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$. Each node v_i contains a single agent i having an initial energy $e = x_i$, for $i = 1, 2, \ldots, n$. The weights of edges outgoing from nodes of V are $w(v_i \to s) = x_i/3$ and $w(v_i \to a) = 0$ for $i = 1, 2, \ldots, n$. Moreover we have $w(s \to a) = E/3$ and $w(a \to t) = E/2$. Consider a delivery from s to t. W.l.o.g. we can suppose that this is done by

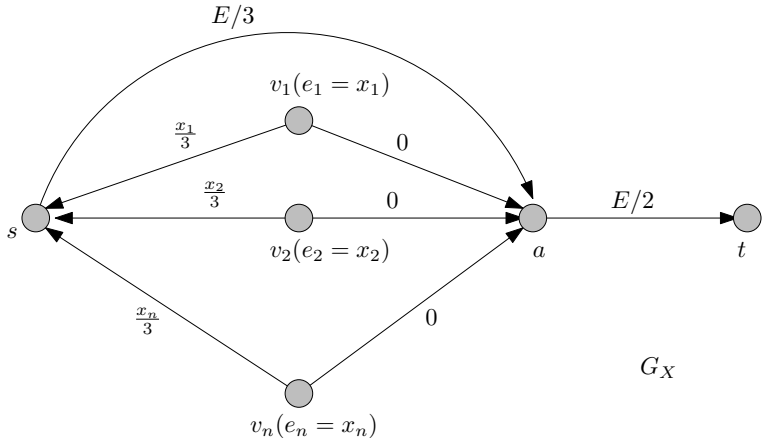

Figure 4.3: Delivery from s to t is possible iff the set of weights x_i can be partitioned into two sub-sets of the same sum.

some agent i, which must traverse the path $v_i \to s \to a \to t$. As no agent can do it using only its own energy (otherwise $x_i \geq 5E/6$ and the ISP trivially has no

solution), some other agents of the collection must walk to s and some other ones must go directly to a, in order to deliver to agent i additional energy needed to complete its path $v_i \to s \to a \to t$.

Assume that X_1, X_2 are the sets of agents which directly move to s and a, respectively. Let

$$\alpha \;=\; \sum_{i \in X_1} x_i,\; \beta \;=\; \sum_{i \in X_2} x_i$$

Hence the energy delivered to s, unused by the agents X_1 incoming to s, is $\frac{2}{3}\alpha$. As this energy must be sufficient to traverse at least edge $s \to a$, we have

$$\frac{2}{3}\alpha \;\geq\; E/3 \tag{4.1}$$

Consider now the maximal energy, which may be available to agent i at point a. It is equal to the sum of energy β, which is brought to point a by agents X_2, and the energy unused by agent i, ending its traversal of edge $s \to a$, which equals $2\alpha/3 - E/3$. As the sum of these energies must suffice to traverse edge $a \to t$ of weight $E/2$ and $\alpha + \beta = E$ we have

$$\frac{E}{2} \leq \beta + \frac{2}{3}\alpha - E/3 = \frac{1}{3}\alpha + \frac{2}{3}\beta = \frac{E}{3} + \frac{1}{3}\beta \tag{4.2}$$

From (4.1) we have $\alpha \geq E/2$ and (4.2) leads to $\beta \geq E/2$, which implies $\alpha = \beta = E/2$.

Consequently, the delivery from s to t in graph G_X is possible if and only if the given instance of the integer partition problem is solvable. This implies NP-completeness of the delivery problem.

As t is the only node having paths incoming from all other nodes, the convergecast for G_X implies the delivery from s to t, hence the convergecast problem is also NP-complete.

□

Theorem 6. *The delivery and convergecast problems are NP-complete for general undirected graphs.*

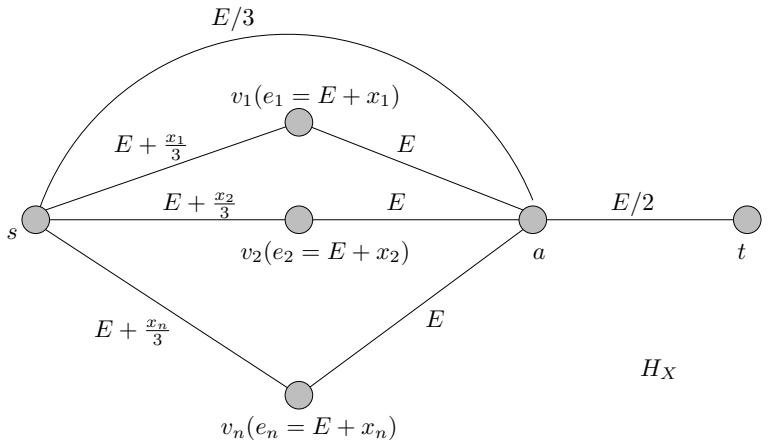

Figure 4.4: The undirected version of the graph from Figure 4.3. The weights of nodes v_i and lengths of edges incident to these nodes are increased by E.

Proof. Consider graph H_X - an undirected version of the graph from the previous proof (see Figure 4.4). Increase the energy of every agent by E, i.e. agent i, initially placed at node v_i, now has energy $E + x_i$, for $i = 1, 2, \ldots, n$. Moreover increase by E the weight of each edge, which is incident to node v_i, i.e. $w(s, v_i) = E + x_i/3$ and $w(v_i, a) = E$, for $i = 1, 2, \ldots, n$.

Delivery. Consider delivery from s to t. Observe that no edge incident to v_i, for $i = 1, 2, \ldots, n$, can be used twice. Indeed, in order to transfer energies between agents they have to meet moving from their initial positions. However, at the moment of such meeting the sum of the remaining energies is smaller than E, which does not permit to traverse any edge incident to x_i for the second time. Clearly traversing directed edges $a \to s$ and $t \to a$ is also useless, hence the delivery from s to t in graph H_X is equivalent to the respective delivery in G_X.

Convergecast. If we consider t as the convergast node, the conergecast problem is equivalent to the delivery from s to t, which implies its NP-completeness.

□

Chapter 5

Broadcast when Agents Start at the Same Node

In this chapter we consider the problem of broadcast when all agents are initially placed at the same node of the tree. We will see, that even in this simplified case the broadcast problem is more involved that the general case of the "symmetric" convergecast problem. Observe that in this case, the energy-exchange aspects of the problem setting are not really relevant. More exactly, we need to attribute energy to all agents so that the agents perform a successful broadcast using the smallest possible total amount of energy. We consider first the case when the source node i.e. the node at which all agents are initially present is also the data source, i.e. it contains the data packet that needs to be broadcast. Our solution to this problem computes the optimal energy needed to perform exploration of a (known) tree by a set of k agents. It is well known that the same problem when the time of the schedule is to be optimized (i.e. the time of arrival to its destination of the last robot), is NP-complete (see [8]).

Then we consider the broadcast problem, in which the source node is different from the data source, i.e. the node from which the data packet needs to be broadcast.

We present almost linear-time, greedy algorithms solving data broadcasting. Our problem can be solved in $O(n \log n)$ time, independently of the number k of available agents. This complexity is reduced to $O(n)$ time in case of unlimited number of agents, or when the number of agents is at least equal to the number of leaves of T.

In the special case when the root, from which all agents start, is also the source node, our approach solves the search problem, when the collection of agents need to search the tree optimally, i.e. using the smallest total energy. Surprisingly, according to our knowledge, this natural setting of the search/exploration problem has not been studied before.

5.1 Agents Starting from the Source Node

We start with an easier case when the root r is the same as the source node s, which contains the initial data packet. Observe that, even if we have unlimited number of agents in r, the problem is nontrivial. In the proposed solution, every agent i initially takes an exact amount of energy needed to traverse some subtree $T(i)$ and its traversal of $T(i)$ must be optimal. The union of all subtrees must sum up to the entire tree T and the choice of the sub-trees must minimize the total energy needed to traverse them.

We consider separately cases of limited and unlimited number of agents. We will show that not all the agents are always activated, i.e. in some cases making walk too many agents would result in a suboptimal algorithm. We say that an agent is *activated* if it is used for walking (consumes a non-zero amount of energy),

copies a data packet to its memory, when arriving to a node at which a copy of data packet is present, and subsequently disseminates it to all nodes visited afterwards.

Denote by T_v the subtree of T rooted at v.

Lemma 2. *Suppose that the source node s is the same as the root r. In every optimal broadcast algorithm, each activated agent should terminate its walk at a leaf of T.*

Proof. The proof goes by contradiction. Suppose that, in an optimal broadcast, some agent i terminates its walk in a non-leaf node v, traversing some edge (w, v) as the last edge of its route. Two cases are possible:

Case1: The traversal of the last edge (w, v) does not coincide with the first visit of node v by agent i. In this case we can remove the traversal of the last edge (w, v) from the route of agent i and the tree explored by agent i remains the same. However, such shortening of the route of agent i reduces its energy cost by $weight(w, v)$, which contradicts the optimality of the original traversal.

Case2: The traversal of the last edge (w, v) by agent i coincides with its first visit of node v. In such a case, agent i could not previously enter the subtree T_v (otherwise this would imply the second visit of v). Consequently, as T_v contains at least one leaf, unvisited by agent i, it must be visited by some other agent j. However, to reach any leaf of T_v from the starting position r, agent j must visit v on its route. As v does not need to be visited by two different agents, we can then again shorten the route of i by the last edge (w, v), reducing its cost. This contradicts optimality of its original route. □

The subset of leaves of T, at which the activated agents of an optimal algorithm terminate their paths, are called *critical leaves*. Each path from root r to a critical leaf is called a *critical path*. The union of all critical paths forms a tree, rooted at r, that we call the *frame* of the algorithm.

5.1.1 Scheduling Agent Movements when Critical Leaves are Known

We start with the presentation of an algorithm which designs the movements of the agents once the set of critical leaves has been obtained. As agents possess the information about the packet from the start, it is sufficient to generate the trajectories of all robots, disregarding synchronization between actual movements of different agents.

Consider a subset $\mathcal{L}$ of critical leaves of tree T. Define $frame(\mathcal{L})$ as the union of all critical paths, i.e. the sub-tree of T induced by $\mathcal{L}$ and all its ancestors, see Figure 5.1. By $|frame(\mathcal{L})|$ we understand the sum of weights of all edges of $frame(\mathcal{L})$. Observe that the edges $T \setminus frame(\mathcal{L})$ form a set of sub-trees rooted at the nodes of $frame(\mathcal{L})$. We call them *hanging s* and we denote the set of hanging sub-trees by $H(\mathcal{L})$.

Once we know the optimal set of critical leaves $\mathcal{L}$, then an optimal schedule is easy to construct. Below we give the algorithm ConstructSchedule, which constructs the optimal schedule for the given set of k agents. In fact we concentrate later only on computing the optimal $\mathcal{L}$ (needed in line 1 of the algorithm Constructschedule). Our main result is the computation of minimum cost in almost linear time, which also implies computing the optimal set $\mathcal{L}$.

Algorithm ConstructSchedule(k);

1. Compute the set of critical leaves $\mathcal{L}$, such that $|\mathcal{L}| \leq k$, which maximizes $\Delta(\mathcal{L})$.
2. Assign to every critical leaf l_i a different agent i which will terminate its walk at l_i.
3. Assign arbitrarily each subtree $T' \in H(\mathcal{L})$ to a single critical leaf $L(T')$, such that T' has the root on the critical path from r to $L(T')$.
4. **for** each leaf $l_i \in \mathcal{L}$ **do**
   - 4.1. Agent i follows the critical path from r to the critical leaf l_i,
   - 4.2. On the way to its assigned critical leaf l_i the agent i makes a full DFS traversal of each hanging subtree $T' \in H(\mathcal{L})$ such that $L(T') = l_i$.

Observe that the total number of edge traversals, generated by the algorithm ConstructSchedule, can be quadratic.

Denote by $path(u, v)$ the set of nodes on the simple path between u and v (including u, v) and let $|path(u, w)|$ denote the distance (sum of edge weights) from node u to w in tree T. We denote also $depth(w) = |path(r, w)|$.

We define below a function $\Delta(\mathcal{L})$ which measures the efficiency of the broadcasting algorithm having $\mathcal{L}$ as its critical leaves.

$$\Delta(\mathcal{L}) \;=\; 2|frame(\mathcal{L})| - \sum_{w \in \mathcal{L}} depth(w) \tag{5.1}$$

The following lemma shows what is the value of $MinCost(T, k)$ - the energy cost of the schedule produced by the algorithm ConstructSchedule for k agents starting at the root of tree T. The energy depends on the choice of the set of critical leaves $\mathcal{L}$. The construction of the set $\mathcal{L}$ minimizing the energy cost will be discussed in the subsequent sections.

Lemma 3. *Assume k agents are placed initially in the source node $r = s$ of T. Then*

$$MinCost(T, k) \;=\; 2|E| - \Delta(\mathcal{L}),$$

where $\mathcal{L}$ is a subset of leaves maximizing $\Delta(\mathcal{L})$ over $|\mathcal{L}| \leq k$.

Proof. The algorithm has enough agents, so that to every critical leaf l_i corresponds a different agent i, which terminates its walk at l_i. The edges of all hanging subtrees $H(\mathcal{L})$, i.e. all edges of $T \setminus frame(\mathcal{L})$, are traversed twice in step 4.2 of the algorithm. Moreover each edge of a critical path is traversed in step 4.1 as many times as there are critical paths containing this edge. Consequently, the total cost of such traversal of T is twice the sum of lengths of edges belonging to $T \setminus frame(\mathcal{L})$, and the sum of the critical path lengths of the $frame(\mathcal{L})$. Hence the total cost equals

$$\begin{aligned} 2|T| \setminus frame(\mathcal{L})| + \sum_{w \in \mathcal{L}} depth(w) \;=\; 2|E| \;-\; (2 \cdot |frame(\mathcal{L})| - \\ \sum_{w \in \mathcal{L}} depth(w)) \;=\; 2|E| - \Delta(\mathcal{L}) \end{aligned} \tag{5.2}$$

By Lemma 2, each optimal algorithm using at most k agents, corresponds to $frame(\mathcal{L})$ for some $\mathcal{L}$. Therefore, the cost represented in equation 5.2 is minimized for maximal $\Delta(\mathcal{L})$. □

Consequently, the broadcasting problem reduces in this case to the computation of $\mathcal{L}$ which maximizes $\Delta(\mathcal{L})$ with $|\mathcal{L}| \leq k$. The set $\mathcal{L}$ will be computed incrementally and in a greedy way. We conclude this section with some observations needed for the incremental construction of the optimal set of critical leaves.

Assume $\mathcal{L}$ is a set of leaves and consider a leaf $w \notin \mathcal{L}$. Denote by $LCA(w, \mathcal{L})$ the lowest common ancestor of w and some leaf from $\mathcal{L}$ (i.e. the lowest node belonging to $path(w, r)$ and $frame(\mathcal{L})$). Define $LCA(w, \emptyset) = r$.

Let

$$\delta(w, \mathcal{L}) \;=\; |path(u, w)| - |path(r, u)|, \tag{5.3}$$

where $u = LCA(w, \mathcal{L})$. Equivalently we have

$$\delta(w, \mathcal{L}) \;=\; depth(w) - 2 \cdot depth(LCA(w, \mathcal{L})) \tag{5.4}$$

Observation 2. *For $\mathcal{L}_1, \mathcal{L}_2$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and for any leaf w we have $\delta(w, \mathcal{L}_1) \geq \delta(w, \mathcal{L}_2)$.*

Indeed the statement of the Observation 2 follows from the fact that $\mathcal{L}_1 \subseteq \mathcal{L}_2$ implies $depth(LCA(w, \mathcal{L}_1)) \leq depth(LCA(w, \mathcal{L}_2))$.

Lemma 4. *For a given subset of leaves $\mathcal{L}$ and a leaf $\tilde{w} \notin \mathcal{L}$:*

$$\Delta(\mathcal{L} \cup \{\tilde{w}\}) \;=\; \Delta(\mathcal{L}) \;+\; \delta(\tilde{w}, \mathcal{L}). \tag{5.5}$$

Proof. If we add $\tilde{w}$ to $\mathcal{L}$, the new path between $LCA(\tilde{w}, \mathcal{L})$ and $\tilde{w}$ is added to $frame(\mathcal{L})$ (cf. Figure 5.1). Hence, according to formula 5.1 we have

$$\Delta(\mathcal{L} \cup \{\tilde{w}\}) - \Delta(\mathcal{L}) = 2|frame(\mathcal{L} \cup \{\tilde{w}\})| - 2|frame(\mathcal{L})| -$$

$\sum_{w \in (\mathcal{L} \cup \{\tilde{w}\})} depth(w) + \sum_{w \in \mathcal{L}} depth(w) = 2|path(LCA(\tilde{w}, \mathcal{L}), \tilde{w})| - depth(\tilde{w}) =$
$depth(\tilde{w}) - 2 \cdot depth(LCA(\tilde{w}, \mathcal{L}) = \delta(\tilde{w}, \mathcal{L})$ □

5.1.2 A Schematic Algorithm Computing the Minimal Cost

The formula (5.5) from Lemma 4 is used to design our algorithm Schematic-MinCost. The idea of the algorithm may be viewed as an incremental, greedy construction of the optimal set of critical leaves $\mathcal{L}$, by adding them, one by one. At each step we have a current version of the frame, which is augmented by a new

subpath when a new leaf is added to $\mathcal{L}$. Consider $frame(\mathcal{L})$ obtained from the leaves $\mathcal{L}$ assigned to the first $i-1$ agents (that terminate their paths at $i-1$ leaves of $\mathcal{L}$). In the i-th iteration of the main loop we try to decide what is the best use of the next available agent. The i-th agent will terminate its traversal at some leaf w_i of T, not yet present in $frame(\mathcal{L})$. Therefore, $frame(\mathcal{L} \cup \{w_i\})$ will contain some new subpath, disjoint with $frame(\mathcal{L})$, starting at some vertex of $LCA(w_i, \mathcal{L})$ and ending at w_i. Observe, that the usage of agent i, permits the subpath from $LCA(w_i, \mathcal{L})$ to w_i to be traversed once (by a new agent) rather than twice (by some other agent which would need to perform a complete traversal of some subtree containing this path), which results in some energy gain. However such energy benefit is at the expense of bringing the agent from the root r to $(LCA(w_i, \mathcal{L})$. The main loop executions continue as long as such gain is possible (i.e. benefit minus expense is positive) and there are still available agents to be used. Such benefit is represented by the function $\delta(w_i, \mathcal{L})$ and our algorithm chooses the leaf offering the largest benefit. We prove later that this greedy approach results in construction of the best possible set of critical leaves.

Algorithm Schematic-MinCost(T, k);

1. $\mathcal{L} := \emptyset$;
2. **while** $|\mathcal{L}| \leq k$ and $\exists(\, w \notin \mathcal{L})\, \delta(w, \mathcal{L}) > 0$ **do**
3. choose a leaf $w \notin \mathcal{L}$ with maximum $\delta(w, \mathcal{L})$;
4. $\mathcal{L} := \mathcal{L} \cup \{w\}$;
5. **return** $|2E| - \Delta(\mathcal{L})$;

Example 2. *Figure 5.1 illustrates the execution of one step of the algorithm. The set $\mathcal{L}$ contains leaves w_1, w_2. The value of $\Delta(\mathcal{L}) = 36 - 19 = 17$, cf. formula 5.1.*

Among the remaining leaves, the maximal benefit is obtained by including w_4 *in the set of critical leaves as* $\delta(w_4, \mathcal{L}) = 3$. *Then* $\Delta(\{w_1, w_2, w_4\}) = 20$. *As for the remaining leaves the values of* δ *are not positive, only three agents are activated (even if more are available) and, by Lemma 3, the cost of the optimal algorithm equals*

$$2|E| - \Delta(\{w_1, w_2, w_4\}) = 56 - 20 = 36.$$

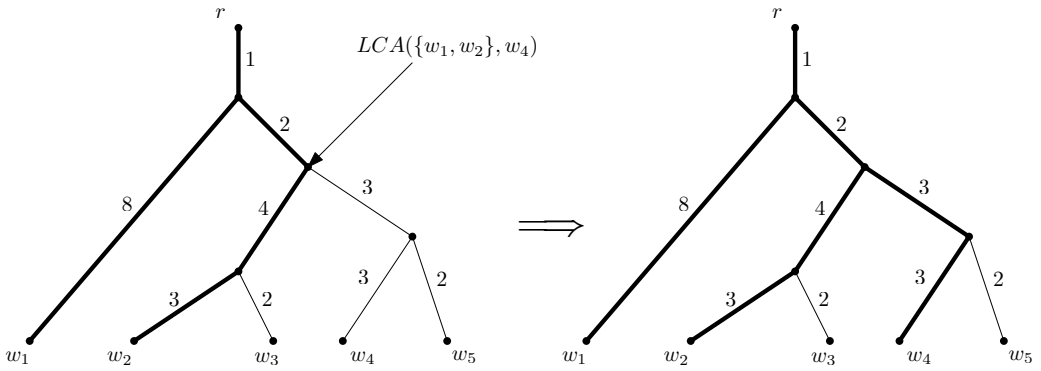

Figure 5.1: The iteration which starts with a set of leaves $\mathcal{L} = \{w_1, w_2\}$, then w_4 is added to $\mathcal{L}$. Bold edges belong to the frames (subtrees $frame(\mathcal{L})$ of paths from the root to set $\mathcal{L}$ of critical leaves) before and after inclusion of w_4. We have $\Delta(\{w_1, w_2, w_4\}) = \Delta(\{w_1, w_2\}) + \delta(w_4, \{w_1, w_2\})$

Observe that algorithm Schematic-MinCost is in fact non-deterministic as it is possible that more than one leaf having the same value of δ may be chosen in line 3. Moreover, among optimal broadcasting algorithms, it is possible that the number of agents used may be different. This is possible if we activate an agent terminating at a leaf w for which $\delta(w, \mathcal{L}) = 0$.

The following lemma will show that the greedy approach of our algorithm, which is based on taking the leaves in decreasing profit for our algorithm is correct. The idea of its proof is to show that the set $\mathcal{L}'$ must contain some leaf w^*, such that $w^* \neq w_i$, for $i = 1, 2, \ldots, t+1$ and that exchanging w^* by w_{t+1} in the set $\mathcal{L}$" will not increase the cost of the corresponding broadcasting algorithm. More precisely we have the following lemma:

Lemma 5. *Assume that in the algorithm Schematic-MinCost we insert the sequence $w_1, w_2, \ldots w_m$ of leaves into $\mathcal{L}$ and there is a set $\mathcal{L}'$ maximizing $\Delta(\mathcal{L}')$ such that $\{w_1, w_2, \ldots w_t\} \subseteq \mathcal{L}'$, $t < m$. Then there exists a set $\mathcal{L}''$, also maximizing $\Delta(\mathcal{L}'')$, which contains $\{w_1, w_2, \ldots w_t, w_{t+1}\}$.*

Proof.

Let $\mathcal{A}$ denote the set of all optimal broadcasting algorithms using s agents with $s \leq k$.

Denote $\mathcal{L}_t = \{w_1, w_2, \ldots, w_t\}$ Suppose, to the contrary, that there exists no optimal algorithm in $\mathcal{A}$ whose set of critical leaves contains all $\{w_1, w_2, \ldots w_t, w_{t+1}\}$. Consider the set of critical leaves $\mathcal{L}'$ and $frame(\mathcal{L}')$. Let $u = LCA(w_{t+1}, \mathcal{L}')$. As $\mathcal{L}_t \subset \mathcal{L}'$, u lies on the path from $LCA(w_{t+1}, \mathcal{L}_t)$ to w_{t+1}. Two cases are possible:

Case 1: $u = LCA(w_{t+1}, \mathcal{L}_t)$.

Due to $LCA(w_{t+1}, \mathcal{L}_t) = LCA(w_{t+1}, \mathcal{L}')$ and formula 5.4 we have:

$$\delta(w_{t+1}, \mathcal{L}_t) = \delta(w_{t+1}, \mathcal{L}') \tag{5.6}$$

We have:

$$\Delta(\mathcal{L}_t) < \Delta(\mathcal{L}_{t+1}) \leq \Delta(\mathcal{L}') \text{ and } \delta(w_{t+1}, \mathcal{L}) > 0,$$

hence, by Equation 5.5 there exists a leaf $w^* \in \mathcal{L}'$, such that $w^* \notin \mathcal{L}_t$ (see Fig 5.2 (a)).

By the condition in line 3 of algorithm Schematic-MinCost, as the algorithm choses w_{k+1} rather than w^*, we have

$$\delta(w_{t+1}, \mathcal{L}_t) \geq \delta(w^*, \mathcal{L}_t) \tag{5.7}$$

Finally by Observation 2

$$\delta(w^*, \mathcal{L}_t) \geq \delta(w^*, \mathcal{L}') \tag{5.8}$$

Combining equations 5.6, 5.7 and 5.8 we get $\delta(w_{t+1}, \mathcal{L}') \geq \delta(w^*, \mathcal{L}')$, hence, by formula 5.5

$$\delta(w_{t+1}, \mathcal{L}') \geq \delta(w^*, \mathcal{L}') \tag{5.9}$$

Using formula 5.4 we have then

$$\Delta(\mathcal{L}' \setminus \{w^*\} \cup \{w_{k+1}\}) = \Delta(\mathcal{L}') - \delta(w^*, \mathcal{L}') + \delta(w_{t+1}, \mathcal{L}') \geq \Delta(\mathcal{L}') \tag{5.10}$$

Therefore, replacing leaf w^* in set $\mathcal{L}'$ by w_{k+1} we obtain a set of critical leaves containing $\{w_1, w_2, \ldots w_t, w_{t+1}\}$, which still leads to the optimal cost. This is a contradiction.

Case 2: $u \neq LCA(w_{t+1}, \mathcal{L}_t)$.

Here, similarly to the previous case, we also look for a leaf in $\mathcal{L}'$ which may be replaced by w_{t+1}, so that the total efficiency Δ of the set of leaves is not be decreased.

Denote $v = LCA(w_{t+1}, \mathcal{L}')$. As $LCA(w_{t+1}, \mathcal{L}_t) \neq v$, the subtree of $frame(\mathcal{L}')$ rooted at v contains at least one leaf $w^* \in \mathcal{L}'$.

Observe that v is the lowest common ancestor of w_{t+1} and w^* in tree T (see Fig. 5.2 (b)). Let $x = LCA(w^*, \mathcal{L}' \setminus \{w^*\})$. Clearly $x = v$ or x is a descendant of v in $\mathcal{L}$ (cf. Fig. 5.2 (b)). By the condition in line 3 of algorithm Schematic-MinCost, as the algorithm chose w_{k+1} rather than w^*, we have $\delta(w_{t+1}, \mathcal{L}_t) \geq \delta(w^*, \mathcal{L}_t)$, hence

$$|path(u, v)| + |path(v, w_{t+1})| - |path(r, u)| \geq$$

$$|path(u, v)| + |path(v, x)| + |path(x, w^*)| - |path(r, u)|$$

This implies that:

$$|path(v, w_{t+1})| \geq |path(v, x)| + |path(x, w^*)|$$

Consequently, we have

$$\delta(w_{t+1},\, \mathcal{L}' \setminus \{w^*\}) \;=\; |path(v, w_{t+1})| - |path(r, v)| \;\geq\; |path(v, w^*)| - |path(r, v)|$$

$$\geq\; |path(x, w^*)| - |path(r, x)| \;=\; \delta(w^*, \mathcal{L}' \setminus \{w^*\})$$

Hence replacing in $\mathcal{L}'$ leaf w^* by w_{k+1} we obtain a set of critical leaves containing $\{w_1, w_2, \ldots w_t, w_{t+1}\}$ which leads to the optimal cost. This is a contradiction completing the proof of the lemma. □

Lemma 6.

(a) The set $\mathcal{L}$ computed by the algorithm Schematic-MinCost maximizes $\Delta(\mathcal{L})$.

(b) The value $2|E| - \Delta(\mathcal{L})$, output by the algorithm Schematic-MinCost, is the minimum amount of energy needed for broadcasting using k agents initially placed in the source $r = s$.

Proof.

(a) Using inductively Lemma 5 we prove that the entire set $w_1, w_2, \ldots w_m$ belongs to a set of critical leaves used by an optimal algorithm. By the exit condition of the while loop at line 2 of the algorithm Schematic-MinCost, there is no other leaf which may be added to such critical set of leaves improving the cost of the algorithm.

(b) This point follows directly from (a) and Lemma 3. □

Observe that every agent possesses the information about the packet at the very beginning of the algorithm. Then, as observed before, once the trajectories

of each activated agent are determined, the timing of the travel of each agent is independent of the timing of the travel of any other agent. We conclude then by the following observation, which will be useful in the next section.

Observation 3. *Any energy-optimal schedule may be designed in such a way that the time intervals, during which agents perform their travel, are pairwise disjoint. In particular, we can choose any agent and make this agent complete its walk before any other agent starts walking.*

5.1.3 Efficient Implementation of the Algorithm Schematic-MinCost

Efficiency of Schematic-MinCost depends on the cost of computing *on-line* the best $\delta(w, \mathcal{L})$. We replace it by introducing a more efficient function Gain(v) which does not depend on $\mathcal{L}$ and can be computed *off-line* in linear time. The algorithm Schematic-MinCost subsequently adds leaves to the set $\mathcal{L}$, each time choosing the leaf w offering the largest gain, i.e. the largest reduction $\delta(w, \mathcal{L})$ in the cost of the broadcasting schedule. The values of function δ for any leaf w, which does not yet belongs to $\mathcal{L}$, may change with subsequent modifications of $frame(\mathcal{L})$. In order to avoid recalculations of the function δ we propose the following solution.

Consider the moment when the leaf w is being added to the current set $\mathcal{L}$. Let v be a child of $LCA(w, \mathcal{L})$, which belongs to the path from $LCA(w, \mathcal{L})$ to w. Let $maxpath(v)$ be the longest path starting at v (and going away from the root). If there is more than one such path, we choose any one of them arbitrarily. We denote by leaf$(maxpath(v))$ the last node on such path. Observe that, at the moment when w is being added to $\mathcal{L}$, we have $|maxpath(v)| = |path(v, w)|$. For any node $v \neq r$ we define

$$\text{Gain}(v) \;=\; |maxpath(v)| + \text{weight}(\text{parent}(v), v) - |path(r, \text{parent}(v))| \qquad (5.11)$$

By convention, we also set $\text{Gain}(r) = |maxpath(r)|$.

Observation 4. *Assume $\mathcal{L}$ is a set of leaves. It follows from Equation 5.3, that*

$$\max\{\text{Gain}(v) : v \notin frame(\mathcal{L})\} = \max\{\delta(w, \mathcal{L}) : w \notin \mathcal{L}\}$$

Following the above Observation, in our algorithm we will be looking for nodes v, which are not in the current $frame(\mathcal{L})$.

Algorithm MinCostLimited(T, k);

1. $X := \{v \in V : \text{Gain}(v) > 0\}$;
2. Sort X with respect to $\text{Gain}(v)$ in non-increasing order;
3. $\Delta := 0$; $\mathcal{L} := \emptyset$;
4. **while** $X \neq \emptyset$ and $|\mathcal{L}| \leq k$ **do**
5. choose $v \in X$ with maximum $\text{Gain}(v)$;
6. $\Delta := \Delta + \text{Gain}(v)$;
7. remove from X all nodes belonging to $maxpath(v)$;
8. $\mathcal{L} := \mathcal{L} \cup \text{leaf}(maxpath(v))$;
9. **return** $2 \cdot |E| - \Delta$

/* $|\mathcal{L}|$ equals the number of *activated* agents */

Theorem 7. *The algorithm MinCostLimited(T, k) correctly computes in $O(n \log n)$ time the minimal amount of energy, which is needed to perform the data broadcast by k agents.*

Proof. We prove, by induction on the iteration of the while loop from line 4, that the node v chosen in line 5 does not belong to the current $frame(\mathcal{L})$. Indeed, in the first iteration of the while loop from line 4, $frame(\mathcal{L})$ is empty. In every other

iteration, because of the leaf added to $\mathcal{L}$ in line 8, $frame(\mathcal{L})$ is augmented by the nodes of $maxpath(v)$, but all these nodes are then removed from set X in line 7. Therefore, in each execution of line 5 no node of X belongs to $frame(\mathcal{L})$.

Consequently, by Observation 4, every value of Gain chosen in line 5 of algorithm MinCostLimited is the same as the value of δ from the corresponding iteration of line 3 of algorithm Schematic-MinCost. Moreover, the same leaf is added to the set of critical set of leafs $\mathcal{L}$ in the corresponding iterations of both algorithms. The final critical set of leaves is then the same for both algorithms.

In the variable Δ is accumulated the sum of the values of function Gain for all nodes chosen in all iterations of the while loop. By Observation 4, after exiting the while loop, Δ equals the sum of values of function δ for all leafs from the final critical set $\mathcal{L}$. By Lemma 4, this sum equals $\Delta(\mathcal{L})$ and the final value of the computed cost equals $2|E| - \Delta(\mathcal{L})$. By Lemma 6 this proves the correctness of algorithm MinCostLimited. We consider now the time efficiency of the algorithm. Observe first, that in the preprocessing, the values of Gain(v) can be computed in linear time. Recall that, by formula 5.11, we need to compute the values of $|maxpath(v)|$, weight$(\text{parent}(v), v)$ and $|path(r, \text{parent}(v))|$

Observe, that all these values may be computed using depth-first-search traversal (DFS) of T. Indeed weight$(\text{parent}(v), v)$ and $|path(r, \text{parent}(v))|$ may be obtained when DFS enters node v from its parent. On the other hand, $|maxpath(v)|$ is obtained when DFS visits v for the last time (arriving from its last child).

The amortized complexity of line 7 is also linear. Assume that X is implemented as a bidirectional list and each node v of the tree T contains a pointer to the element of X corresponding to Gain(v). Then the removal operation in line 7 takes constant time for each considered node v, hence the $O(n)$ time overall. As each other instruction inside the while loop takes constant time, the complexity

of all lines of the algorithm, except line 2, is $O(n)$. The overall complexity is then dominated by the $O(n \log n)$ sorting in line 2. □

5.1.4 Unlimited Number of Agents in the Source

For a set of nodes Y denote by children(Y) the set of all children of nodes in Y.

Algorithm MinCostUnlimited(T);

/* The number of agents is unlimited */

1. $X := \{r\}$; $\Delta := 0$; $\mathcal{L} := \emptyset$;
2. **while** $X \neq \emptyset$ **do**
3. $v :=$ Extract any element of X;
4. Add to X each $x \notin maxpath(v)$ such that
   $parent(x) \in maxpath(v)$ and Gain(x) > 0;
5. $\Delta := \Delta +$ Gain(v); $\$\mathcal{L} := \mathcal{L} \cup \{leaf(maxpath(v)\}$;
6. $/* X = \{ v \in children(frame(\mathcal{L})) : v \notin frame(\mathcal{L}) \} */$
7. **return** $2 \cdot |E| - \Delta$

We show now that in the case of the unlimited number of available agents, or if the number of agents is at least equal to the number of leaves of T, the algorithm computing the minimal amount of energy works in $O(n)$ time. The proof is based on the fact that the set X is now restricted only to the children of the current frame and we choose each of them at some time. Since any two nodes, which are present in X at a same moment, never interfere (i.e. choosing one of them does never affect the Gain function of the other one), they may be treated in any order (as the number of available agents is sufficient for taking each of them at some time). This allows to avoid sorting and the time of the entire treatment is

proportional to the number of edges of T. More precisely we have the following theorem:

Theorem 8. *Assume that the number of agents initially placed at the source node is at least equal to the number of leaves of T. Then the algorithm MinCostUnlimited(T) correctly computes in $O(n)$ time the minimal amount of energy needed for the broadcast in T.*

Proof. We first observe that, at the end of each iteration of while loop from line 2, each element x_i of X has a parent in the current $frame(\mathcal{L})$ (cf. the comment in line 6). Indeed, at the beginning of the first iteration, the root r - the only element of X is extracted from it and since then only elements having parents in $maxpath(v)$ are added to X.

As $maxpath(v)$ leads to the new critical leaf, added in line 5, $mathpath(v)$ belongs to the current frame, so $parent(x_i) \in frame(\mathcal{L})$. By line 4, $x_i \notin frame(\mathcal{L})$ and $\text{Gain}(x_i) > 0$.

From the above observation we conclude that the set of critical leafs $\mathcal{L}$ computed by algorithm MinCostLimited is the same as the set computed by algorithm MinCostUnlimited. Indeed, by the above observation, for each pair of nodes $x_1, x_2 \in X$ we have $maxpath(x_1) \cap maxpath(x_2) = \emptyset$. Therefore the order in which the elements of X are considered is irrelevant and they generate the same sets $\mathcal{L}$ of critical leafs but possibly in different order.

Consequently, the values returned in line 9 of MinCostLimited and the line 7 of MinCostUnlimited are the same. This completes the proof of correctness of MinCostUnlimited.

We prove now $O(n)$ time complexity. Function Gain is precomputed in $O(n)$ time as in MinCostLimited. Observe that line 4 of the algorithm MinCostUnlimited is executed in $O(n)$ total amortized time. Indeed, for each node of the tree T added to $maxpath(v)$ we consider each of its children x to check if $x \notin maxpath(v)$

and if $\text{Gain}(x) > 0$ in order to decide its inclusion in X. The overall time complexity of line 4 does not exceed the number of edges of T. As there is $O(n)$ iteration of the while loop from line 2, all other instructions take $O(n)$ time as well. □

5.2 All Agents Start from the Same Node r that is Different from the Source s

In this section we extend the consideration to the case when the initial position of the packet is not at the root of the tree. We show that this setting may be reduced to the case studied in the previous sections.

It would be helpful if we design the schedule, so that a robot moves along its trajectory independently from the timing of the motion of any other robot. By Observation 3, this was possible when the robots were initially placed at the source node s. However, in the current setting, at every time moment the robots executing an optimal schedule can be divided into two categories: the robots which already know the packet and the robots that do not. Clearly, the former category of robots are not restricted by their movement. On the other hand, the robots not knowing the packet might need to delay their movement as they may have to visit a node after the packet is deposited there.

The path between the root r and the source s we call the *backbone* of tree T and we denote it by B. We start with the following lemma.

Lemma 7. *There exists an optimal broadcasting algorithm in which the first activated agent starts moving towards the source node s, eventually returning to r, before any other agent is activated.*

Proof. The packet initially present at the source node s needs to be transported to all other nodes of the tree, including root r. Therefore, there must exist an agent which travels from r to s to pick up the packet. After that, a copy of the packet must be transported along the backbone B, starting at s and ending at r. During this travel of the packet along B, it may be transported by divers agents. However, when the packet is left by some agent i at a point p of B and picked later by some agent j, we can make agent i wait at point p until the arrival of agent j. At that moment, as agents are identical, we could exchange the roles of agents i and j and it is still agent i which continues to transport the packet. We conclude, by induction, that the packet is transported all the way by the same agent.

Observe as well, that the remaining agents that were exchanging roles with the agent i, in fact, do not need to start their travel before agent i reaches r. Indeed, they may wait at r until the packet is brought there by agent i and start their respective routes afterwards. □

We now construct the reduction from the setting where $r \neq s$ to the case $r = s$. For every instance I of the problem for $r \neq s$ we create an instance I' of the problem where $r = s$. We show that to solve I it is sufficient to solve I', where we use the results from the previous sections.

Let I be a given instance of the broadcast problem, in which we have tree T with k agents $1, 2, \ldots, k$ initially placed at root r and the source $s \neq r$.

We describe an instance I' of the broadcast problem with $s = r$ and $k + 1$ agents. We construct the tree T' by adding to T one extra leaf w_0 and an edge from node s to w_0 of weight W, where W equals the sum of weights of all edges of T. We define its root $r' = r$ in which we place $k' = k+1$ mobile agents, represented by the integers $0, 1, \ldots, k$. We also set the source $s' = r'$ (see Fig. 5.3).

Lemma 8. *[Reduction-Lemma] If $s \neq r$ in T then*

$$MinCost(T, k) \;=\; MinCost(T',\, k+1) \;+\; |path(r, s)| \;-\; W,$$

where in T' the source nodes' equals the initial location of $k+1$ agents.

Proof. Consider first an optimal solution to the instance I' produced by algorithm ConstructSchedule. The weight of the edge incoming to node w_0 is so large that the leaf w_0 must belong to the set of critical leaves $\mathcal{L}$ and some agent 0 must terminate its walk in w_0. By Observation 3, we can suppose that agent 0 is the very first agent activated and the remaining agents didn't start before agent 0 reaches node w_0. Denote by T_H the set of all edges, traversed by agent 0, outside the simple path from r to w_0. By algorithm ConstructSchedule, T_H forms a subset of hanging sub-trees.

Consider now an optimal solution to the instance I, which verifies Lemma 7. In this solution, the first activated agent 1, starting at r, travels along the backbone B to the source s (without any detour) and then continues its walk, eventually returning to r (bringing the packet), before any other agent starts moving. Obviously, on its way back along the backbone (i.e. from s to r) agent 1 may visit some nodes outside the backbone before returning to r.

Assume then, that agent 1, during its return from s to r along the backbone, traverses exactly the subtrees formed by the edges T_H (cf. Fig. 5.3). Consider the time moment in instance I' where agent 0 arrives at leaf w_0 and the time moment when in instance I agent 1 returns to root r. In both cases we have k agents and the packet available at the root r and the part of the tree that still needs to be explored equal to $T \setminus (T_H \cup B)$. Therefore, if we use the trajectories of the remaining k agents $1, 2, \ldots, k$ from the optimal solution of instance I' to complete the instance I the obtained solution of I is also optimal.

Observe that the assumption that agent 1 visited the sub-trees formed by the edges of T_H may be dropped. Indeed, all sub-trees of T_H are the hanging subtrees of the optimal solution and each of them is DFS traversed by some agent. Assigning any such sub-tree to agent 1 or any other agent visiting its root does not change the cost of the solution (recall line 3 of algorithm ConstructSchedule).

As from the moments when the situations in instances I' and I are identical all agents walk along the same trajectories in T and T', respectively, the cost of the solution of instance I differs from the solution of instance I' by the difference in the amounts of energy spent by the first agents of each instance, respectively. As this difference is $W - |path(r, s)|$, we have

$$MinCost(T, k) \;=\; MinCost(T',\, k+1) \;+\; |path(r, s)| \;-\; W$$

□

Theorem 9. *Suppose that in the tree T the root r is different from the source s. We can solve the limited broadcast problem in $O(n \log n)$ time. If k is at least equal to the number of leaves in T we solve the broadcast problem in $O(n)$ time.*

Proof. Due to Lemma 8 limited broadcast reduces in linear time to the case when the source is the same as starting location of agents. In the unlimited case we can use Lemma 8 with $k = n$.

Hence the time complexity is asymptotically of the same order as that of the algorithm MinCostLimited(T, k), which is $O(n \log n)$. The case of unlimited broadcast can be done similarly in $O(n)$ time, by reduction to the algorithm MinCostUnlimited. □

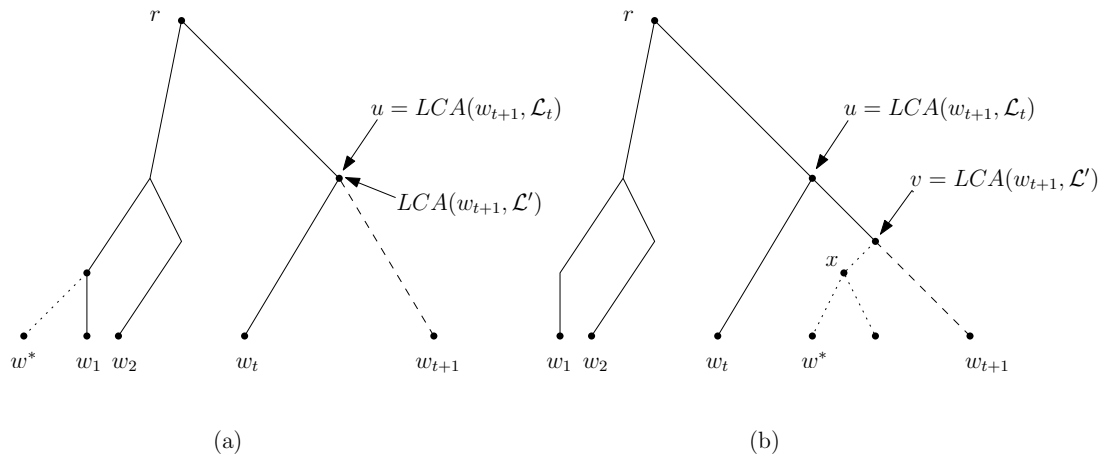

Figure 5.2: The two cases for the proof of the optimality of MinCost. The solid edges belong to both, $\mathcal{L}_t$ and $\mathcal{L}'$. The dashed edges belong only to $\mathcal{L}_t$ and the dotted ones belong only to $\mathcal{L}'$. In both cases, replacing w^* by w_{t+1} does not increase the cost of the schedule.

.

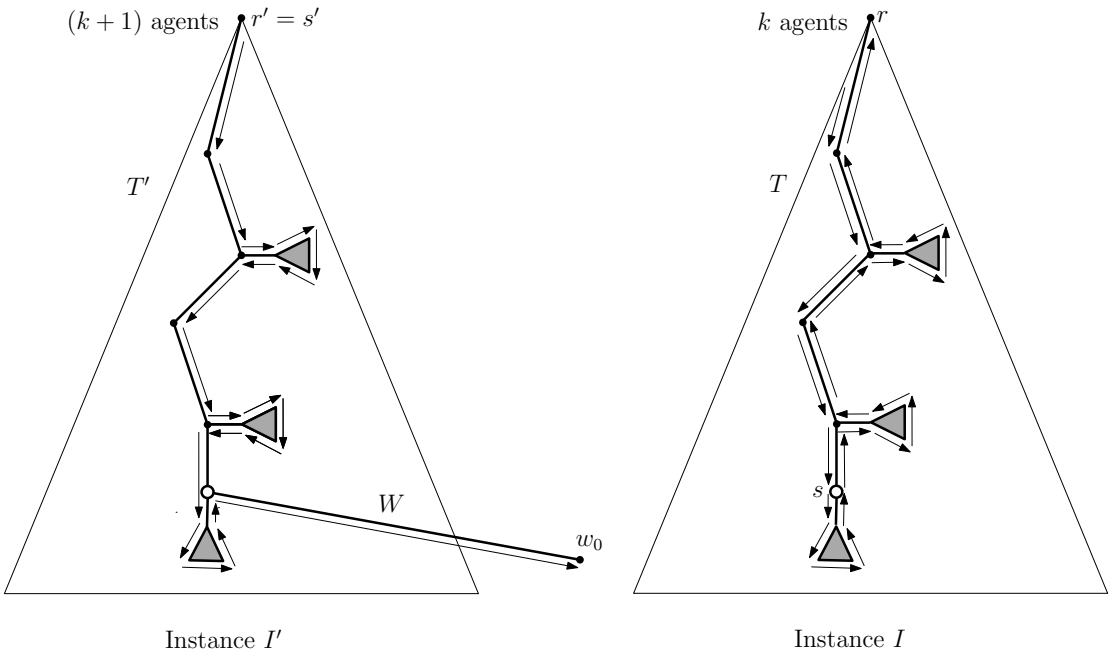

Figure 5.3: On the left: the trajectory of the first activated agent, when the algorithm MinCostLimited is run for instance I'. The first agent terminates its walk in w_0. On the right: trajectory of the first activated agent for instance I. The first agent traverses the same nodes (except w_0) but returns to r to be reused later.

Chapter 6

Broadcast with Mobile Agents Distributed on a Tree

6.1 Introduction

In the previous chapter we presented the solution to the Broadcast problem when the mobile agents start at the same initial position. The energy exchange aspects of the scenario was irrelevant in this case as the agents can distribute freely the available energy at the start of the process. Moreover, as agents did not communicate any data from one to another, no collaboration between the agents was necessary. Also each agent could move independently, i.e. consecutive moves of two different agents could be interweaved arbitrarily.

In this chapter we study the Broadcast Problem when the agents are arbitrarily dispersed in the nodes of the tree. They need to collaborate in order to communicate the data packet present at the root of the tree to all other agents. Each agent is given an amount of initial energy (which may be different for different agents) that is used proportionally to the distance travelled. When two agents meet, they may exchange between them any amount of currently possessed energy. More-

over, if one of the meeting agents possesses the knowledge of the data packet, this knowledge is automatically acquired by the other agent. An agent possessing the knowledge of the packet communicates it also to the nodes that the agent is visiting. Our goal is to schedule the moves of the mobile agents and energy transfers between them that results in the data packet being communicated to all nodes of the tree. Hence, contrary to the model studied in the previous chapter, the agents need to collaborate by scheduling their moves in order to create meetings permitting transfers of energy and the data packet.

Moire exactly we consider the following problem:

General Broadcast Problem

Input :

1. The weighted tree $T = (V, E)$ of n nodes ($|V| = n$).
2. A source node $r \in V$, designed as the root of T containing a data packet P_r.
3. , A collection of k agents $A = \{0, 1, \ldots, k-1\}$
4. A function $i : A \longrightarrow V$ defining the initial position of each agent.
5. A function $e : A \longrightarrow R^+$ defining the initial energy of each agent. R^+ is the set of non-negative real numbers.

Agents' capabilities

1. Agent may walk spending energy proportionally to the distance traveled (sum of weights of the traversed edges).
2. Agent may stop at any node.
3. Agent visiting node r possesses the knowledge of packet P_r and retains this knowledge until the end of the process.

4. Two agents arriving at the same time to a node may exchange between them any amount of actually possessed energy.

5. If one of the two meeting agents possesses the knowledge of P_r then from the meeting time both agents possess the knowledge of P_r until the end of the process.

Output

The decision (YES or NO) whether there exists a schedule, i.e. the sequence of agent's moves and energy transfers between meeting agents such that at some point

a) Each tree node has been previously visited by an agent possessing the knowledge of P_r, and

b) No agent exceeded energy needed for its walk.

The main part of the algorithm will be the computation of the optimal agent's migration flow, i.e. the number of agents traversals of each edge in one direction, minus the number of traversal in the other direction.

The optimal migration flow will be computed using dynamic programming, when for every tree edge the optimal energy usage wil be computed for each possible agent's migration flow through this edge. The dynamic programming will be performed in bottom up traversal of tree T.

Our goal is to solve the following decision problem:

Data broadcasting decision problem: We are given a source node r of an n-node tree T, and a configuration of k agents, placed at selected nodes, each having some initial amount of energy (possibly different for all agents). Is it possible to schedule the moves of the agents and energy transfers so that the initial packet of information placed at node r reaches all nodes of T?

We will look for schedules of agents' movements that will not only result in completing the broadcast, but also attempt to maximize the energy, which is eventually brought to the root. We call such schedules *optimal*. Consequently, our approach permits to solve a more general optimization problem:

Data broadcasting optimization problem: What is the largest amount of energy, which may be deposited at source r while some sequence of moves of the agents and energy transfers result in the initial packet reaching all nodes of T?

6.1.1 Preliminairies

In the remainder of the chapter we assume that the tree T is rooted at its source node r.

In our algorithm we propose a specific treatment for each tree node, related to its number of children, presence or absence of an agent and weight of the incident edges. To ease its understanding, we convert the given tree to another one, in which every node will have only one property that needs to be taken into account by our algorithm. The structure of the tree and the lengths of the corresponding weighted paths remain the same for the converted tree, hence the movements of the agents performing broadcasting produced by our algorithm may be reconverted back to the original tree.

We observe first that using the standard folklore technique, by adding extra nodes and edges of zero weight, the original weighted tree may be converted to a *binary* tree, in which all weighted path lengths between the corresponding tree nodes are the same. Although the depth of such converted binary tree is increased, its complexity remains $O(n)$ and it may be obtained in $O(n)$ time. Hence w.l.o.g. we can assume that the given initial tree is binary.

We have the following lemma.

Lemma 9. *By adding extra nodes and edges of zero weight we can convert a binary tree T with agents placed at its nodes into a tree with the following four types of nodes (see Fig. 6.1):*

(a) if v is a terminal node, then it initially contains no agents,

(b) if v is a parent of two children, then v contains no agents and both children are accessible by edges of zero weight,

(c) if v originally contains one or more agents, it has exactly one child accessible by an edge of zero weight, or

(d) node v may contain one child accessible by an edge of non-zero weight.

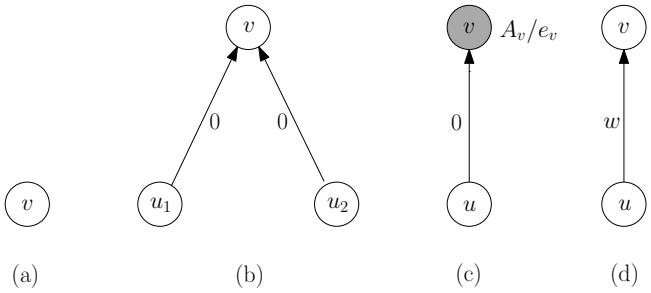

Figure 6.1: Four cases of node v in tree T: (a) terminal node, (b) node with two children, (c) node with an incoming edge of non-zero weight, (d) node containing A_v agents having total energy e_v.

Moreover, the converted tree is of $O(n)$ size and the paths between the corresponding nodes in the converted tree are of the same weight as in the original one.

The proof of the lemma is easy and we omit it. It is possible to make such conversion in linear time with respect to the size of the original tree. For convenience we add to such converted tree one extra node r of type (d), which we make parent of the root of T, using zero-weight edge. In the remainder of the chapter we assume that T denote the converted, rooted version of the tree.

Despite the fact that tree T is undirected, as agents move along the edges in particular direction, we consider also directed version of the edges. In particular, for two adjacent nodes v, w, we denote by (v, w) an undirected edge between v, w and by $v \to w$ and $w \to v$ the two directed edges. For any node $v \neq r$, we denote by T_v a subtree of T, rooted at v, and containing all descendants of v in T. If $w = parent(v)$ we call $v \to w$ the *exit edge* of tree T_v. Hence all exit edges point in the direction of the root.

6.2 Testing Feasibility of Broadcast

In this section we describe an algorithm, which computes the largest amount of energy that may be deposited at the root r by the agents performing a successful packet broadcasting from r to all remaining nodes. The algorithm needs to manipulate its three resources in the form of information (packet), agents and energy. Consequently, energy might be transported by agents from some parts of the tree T to other parts, where it is more needed. Similarly, agents might be more useful when moving from some parts of T so that they finish their walks in other parts. The main idea of our algorithm is to find a combination of transfers of energy and agents across the tree, resulting in the largest unused energy, which is eventually deposited at the root. This is realized by a dynamic programming approach.

Consider an isolated subproblem of the broadcast of the packet present at node v to all nodes of the sub-tree T_v. Such a broadcast might be successfully performed using only agents originally present in T_v (and their energy). However, we may also use agents originally from outside T_v and/or extra energy, incoming via its reversed exit edge. On the other hand, unused energy and/or agents which do not need to terminate their walks inside T_v may be transferred through its exit edge to other parts of tree T, where they turn up to be more useful. We will show that

with respect to any given exit edge $v \to w$, an optimal schedule might consist of three steps (in this order):

1. If the total energy available inside T_v is sufficiently large, an agent will first traverse edge $v \to w$ bringing an excessive energy to node w. Such energy may be subsequently transferred and deposited at the root r. Depending on the distribution of energy inside T_v this step may or may not exist.

2. An agent A will traverse edge $w \to v$ in order to transport the packet into T_v.

3. Then

   (a) Either a number of other agents traverse together edge $w \to v$. Then all these agents, together with agent A and the agents initially present inside T_v will transport the packet to all nodes of T_v.

   (b) Or, a number of other agents traverse together edge $v \to w$. Before exiting from T_v these agents together with agent A and the agents initially present inside T_v will transport the packet to all nodes of T_v.

   (c) Or, no other agent traverses either of the edges $w \to v$ and $v \to w$. In this case agent A together with agents initially present inside will transport the packet to all nodes of T_v, eventually terminating their walks inside T_v.

Suppose for a moment that we know some optimal schedule. Suppose also that, for this optimal schedule and for every subtree T_v, the integer i_v denotes the difference between the number of agents' walks traversing the exit edge $v \to w$ and the number of agents' walks traversing the reverse edge $w \to v$. For a given schedule, we call such i_v the *agents migration flow* of the edge $v \to w$ or shortly its M-flow. For any possible value of M-flow i_v, we denote by $\mathcal{B}_v[i_v]$ the *energy*

potential of the exit edge $v \to w$. The value $\mathcal{B}_v[i_v]$ equals the largest energy that could be deposited at node v, which would permit a successful completion of the broadcast from node v to all nodes of T_v, assuming i_v being the M-flow of the edge $v \to w$. More exactly, if $\mathcal{B}_v[i_v] \geq 0$, then it represents the maximal amount of unused energy that may be left at node v after a successful broadcast from v to all nodes of T_v. Similarly, if $\mathcal{B}_v[i_v] < 0$, then $-\mathcal{B}_v[i_v]$ represents the minimal amount of energy that must be added at node v, after which it would be possible to perform a successful broadcast inside T_v.

It is assumed, that if $i_v \geq 0$, there will be a total of i_v agents available at node v after the broadcast and ready to be used outside T_v. Similarly, it is assumed that if $i_v < 0$, the total number of $-i_v$ extra agents that were not initially placed inside T_v will be used to terminate their broadcasting walks inside T_v.

Fig. 6.2 illustrates the values of all tables $\mathcal{B}$ computed for an example tree. At this stage, it may not be completely clear to the reader how the values of tables $\mathcal{B}$ are computed. This is the goal of Algorithm TEST-BROADCAST-FROM-ROOT given below. However, it is interesting to note, that each table contains a non-increasing sequence of values. This is due to the fact that energy and agents are, to some extent, exchangeable resources. Consider a tree with "heavy" terminal edges. If we send one agent to broadcast a packet inside a given subtree T_v, this agent needs to traverse all but one heavy terminal edges twice (once in each direction). If we have more agents available, they may terminate their walks inside T_v, so that many heavy edges will be traversed once only and energy will be saved.

As in fact we do not know in advance any optimal schedule, our algorithm will compute energy potential for any possible M-flow of every edge. It is worth noting that more than one value of M-flow of a given edge may lead to an optimal schedule. The most important part of our algorithm is the computation of M-flows

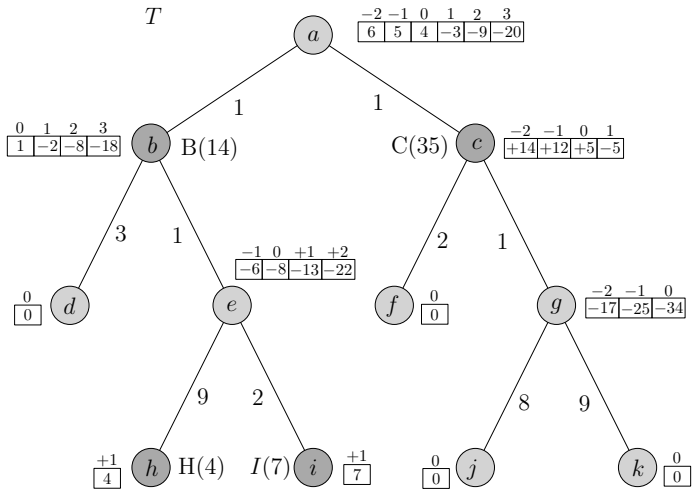

Figure 6.2: Computing $\mathcal{B}$ tables for an example tree. At nodes b, c, h, i are present agents B, C, H, I having the initial amounts of energy $14, 35, 4, 7$, respectively. Each tree node has a $\mathcal{B}$ table associated to it. Only the significant entries of each table are illustrated: for all larger indices of each table all entries are equal to $-\infty$; for all smaller indices of each table the entries are the same as the first one illustrated. As an example note that $\mathcal{B}_b[1] = -2$ corresponds to agent B trajectory *bdbeh* of length 16 and agent I trajectory *ieieb* of length 7. Agent H does not move and its energy is lost in this case. $\mathcal{B}_b[1] = -2$ means that 2 extra units of energy are needed initially at b so that the broadcast resulting in 1 agent eventually arriving at b.

for all edges of T, resulting from some optimal broadcast schedule. This process is done in bottom-up order (i.e. postorder) of tree T. For each exit edge of some tree T_v we compute the array $\mathcal{B}_v[]$ by calculating the energy potentials for all possible M-flows through this edge.

The following algorithm computes the upper bound on the amount of energy that may be brought to the root of T, so that the broadcast may be still performed successfully.

ALGORITHM TEST-BROADCAST-FROM-ROOT(T);
{ input: tree T rooted at r; if node v contains agents, then A_v is the number of agents present at v and e_v is their total energy }

1. **f**or all $v \in T$ and all $-k \leq i \leq k$ **d**o
2. Initialise $\mathcal{B}_v[i] := -\infty$;
3. **f**or all $v \in T$ taken in an ascending order with respect to *root* **d**o
4. COMPUTE $\mathcal{B}(v)$;
5. **if** $\mathcal{B}_r[0] \geq 0$
6. **t**hen Report r as a possible broadcast node;
7. Report $\mathcal{B}_r[0]$ as the maximum energy which may be left at r during a succesful broadcast from r;
8. **e**lse Report broadcast from r is infeasible;

Procedure COMPUTE $\mathcal{B}(v)$ computes the array $\mathcal{B}_v[]$ for each node v, assuming that the same array for the children (or child) of v has been previously computed.

PROCEDURE COMPUTE $\mathcal{B}(v)$;

1. **f**or $i := k$ **t**o $-k$ **d**o
2. **c**ase type of node v **o**f
3. (a) **if** $i \leq 0$ **then** $\mathcal{B}_v[i] := 0$;
4. (b) $\mathcal{B}_v[i] := \max\{\mathcal{B}_{u_1}[j] + \mathcal{B}_{u_2}[h] : j + h = i\}$;
5. (c) **if** $(i - A_v \geq -k)$
6. **then** $\mathcal{B}_v[i] := \mathcal{B}_u[i - A_v] + e_v$
7. **else** $\mathcal{B}_v[i] := \mathcal{B}_u[-k] + e_v$;
8. (d) **if** $i \geq 0 \vee \mathcal{B}_u[i] > 2 \cdot weight(u, v)$
9. **then** $\mathcal{B}_v[i] := \mathcal{B}_u[i] - (|i| + 2) \cdot weight(u, v)$
10. **else** $\mathcal{B}_v[i] := i \cdot weight(u, v)$;
11. **if** $(i < k) \wedge (\mathcal{B}_v[i] < \mathcal{B}_v[i + 1])$ **then** $\mathcal{B}_v[i] := \mathcal{B}_v[i + 1]$;

The following lemma proves that for each node v the sequence of elements of the table $\mathcal{B}_v[]$ is nondecreasing.

Lemma 10. *For any node v and any index i^*, s.t. $-k < i^* \leq k$, we have $\mathcal{B}_v[i^* - 1] \geq \mathcal{B}_v[i^*]$.*

Proof. The proof goes by induction on the height of node v. The claim is clearly true for each node of depth 0 (node of type (a)), for which $\mathcal{B}_v[i] = 0$ if $i \leq 0$ and $\mathcal{B}_v[i] = -\infty$ if $i > 0$ (cf. line 2 of algorithm TEST-BROADCAST-FROM-ROOT(T) and line 3 of procedure COMPUTE $\mathcal{B}(v)$).

Suppose that the claim of the lemma is true for each node of height at most h and consider any node v of height $h + 1$. Three cases are possible:

Case 1 (node v is of type (b), cf. Fig. 6.1). Take any index i^* and suppose that j^* and k^* are such that $i^* = j^* + k^*$ and

$$\mathcal{B}_v[i^*] = \max\{\mathcal{B}_{u_1}[j] + \mathcal{B}_{u_2}[k] : j + k = i^*\} = \mathcal{B}_{u_1}[j^*] + \mathcal{B}_{u_2}[k^*]$$

according to line 4 of procedure COMPUTE $\mathcal{B}(v)$. As nodes u_1 and u_2, which are the children of v, are both of height at most h, by the inductive hypothesis we have

$$\begin{aligned}
\mathcal{B}_v[i^*] &= \max\{\mathcal{B}_{u_1}[j] + \mathcal{B}_{u_2}[k] : j + k = i^*\} \\
&= \mathcal{B}_{u_1}[j^*] + \mathcal{B}_{u_2}[k^*] \leq \\
&\leq \mathcal{B}_{u_1}[j^* - 1] + \mathcal{B}_{u_2}[k^*] \\
&\leq \max\{\mathcal{B}_{u_1}[j] + \mathcal{B}_{u_2}[k] : j + k = i^* - 1\} \\
&= \mathcal{B}_v[j^* - 1]
\end{aligned}$$

Case 2 (node v is of type (c)).

Suppose first that $i^* - A_v - 1 \geq -k$. Then, using the inductive hypothesis, according to line 6 of procedure COMPUTE $\mathcal{B}(v)$ we have

$$\mathcal{B}_v[i^*] = \mathcal{B}_u[i^* - A_v] + e_v \leq \mathcal{B}_u[i^* - A_v - 1] + e_v = \mathcal{B}_v[i^* - 1]$$

Consider the remaining case, when $i^* - A_v - 1 < -k$. Then we have $\mathcal{B}_v[A_v - k] = \mathcal{B}_u[-k] + e_v$ and for which $i < A_v - k$, the value of $\mathcal{B}_v[i]$ is computed in line 7 of procedure COMPUTE $\mathcal{B}(v)$. In this case we have

$$\mathcal{B}_v[A_v - k] = \mathcal{B}_v[A_v - k - 1] = \cdots = [-k]$$

hence the claim of the lemma in this case is also true.

Case 3 (node v is of type (d)). The claim of the lemma follows directly from line 11 of procedure COMPUTE $\mathcal{B}(v)$. Indeed, in this case either we have $\mathcal{B}_v[i^*-1] \geq \mathcal{B}_v[i^*]$ or the condition of the if-clause from line 11 is true and after its execution we obtain $\mathcal{B}_v[i^* - 1] = \mathcal{B}_v[i^*]$.

This completes the proof. □

Before proving that no broadcasting algorithm can bring to the root more energy than Algorithm TEST-BROADCAST-FROM-ROOT(T) in line 7, we give some intuition.

For any node v of type (b), the computation of each component $\mathcal{B}_v[i]$ results in a choice of respective indices j, h for nodes u_1, u_2, which maximize the energy that may be transferred towards the root. This implies the optimal choice of M-flows through the exit edges of T_{u_1} and T_{u_2}. Considering exit edges in the top-down order, we can compute then the index i_v of each table $\mathcal{B}_v[]$ obtaining the M-flows of all edges of tree T, which result in the deposit of $\mathcal{B}_r[0]$ energy at the root. Each such value i_v is the difference between the number of agents traversing the exit

edge $v \to w$ of T_v and the reverse edge $w \to v$. We call i_v an *optimal flow index* of node v.

The broadcasting schedule induces some directed multigraph T_M, built over the set of nodes of T: each directed edge $x \to y \in T_M$ corresponds to a traversal by some agent of the edge $(x, y) \in T$. We can partition the edges of T_M into three classes: *packet transfer* edges T_p, *energy transfer* edges T_e and *agent migration* edges T_a. The class T_p is composed of one copy of each edge $x \to y$, such that x is parent of y in T. As the packet kept in r needs to be broadcast to all nodes of T, each such multi-edge is clearly needed in T_M.

The class T_e contains multi-edges going up the tree T. Their goal is to conjointly move portions of energy in the direction of the root. Clearly only some edges of T induce multi-edges belonging to T_e, as energy is moved towards only when some excess of it is available. Although the subgraph T_e may turn out to be disconnected we create in T_e a multi-edge $v \to y$ whenever $\mathcal{B}[i_v] > 2 \cdot weight(v, y)$ or when $i_v \geq 0$ and $\mathcal{B}[i_v] > weight(v, y)$.

All remaining edges of T_M belong to class T_a and their number and direction observe the M-flow of the corresponding edge of T. In particular, consider any exit edge $v \to y$ of T_v. Let $k = 1$ if there exists a multi-edge $v \to y \in T_e$, otherwise $k = 0$. Then if the optimal flow index $i_v \geq 0$, there are $i_v - k$ copies of $v \to y$ multiedge in T_e. Otherwise, if $i_v < 0$, we have $k - 1 - i_v$ copies of the reverse, $y \to v$ multiedge in T_e.

In the sequel, we denote by the $f(x \to y)$ the element of the M-flow being the number of agents that traverse the edge $x \to y$. Consequently, if x is a child of y in T, we have

$$\mathcal{B}[i_x] = f(x \to y) - f(y \to x)$$

The following lemma shows that the value of $\mathcal{B}_r[0]$, as computed by the algorithm Test-Broadcast-from-Root(T), is the upper bound on the amount of

energy, which may be left at the root of T, when executing a successful broadcasting algorithm.

Lemma 11. *There exists no schedule of agents' movements which performs successful broadcasting and results in depositing at the root r of tree T an amount of energy larger than $\mathcal{B}_r[0]$.*

Proof. Suppose that the claim of the lemma is not true and consider a broadcasting schedule S^* depositing at r an amount of energy $E^* > \mathcal{B}_r[0]$. Let M^* denote the agent migration flow of such schedule and M_v denote (for any node v) the amount of energy E_v, deposited by S^* at node v. We prove by induction on the height of node v that $E_v \leq \mathcal{B}_v[i_v]$, where i_v is the agents' migration flow through the exit edge of T_v, as computed by the algorithm TEST-BROADCAST-FROM-ROOT(T). The claim is clearly true for each node v of height 0 (leaf). Suppose that the claim is true for any node of height h and consider a node v of height $h+1$. Let i_v be the flow of the exit edge of T_v in M^*. Three cases are possible:

Case 1 (node v is of type (b)), see Fig. 6.1. Let i_{u_1} and i_{u_2} denote the flows of M^* through the edges (u_1, v) and (u_2, v), respectively. As $weight(u_1, v) = weight(u_2, v) = 0$, we can assume that in the optimal schedule S^* no agent finishes its walk at node v. Moreover, as no agent was initially present at v we have $i_v = i_{u_1} + i_{u_2}$. By the inductive hypothesis $E_{u_1} \leq \mathcal{B}_{u_1}[i_{u_1}]$ and $E_{u_2} \leq \mathcal{B}_{u_2}[i_{u_2}]$. Hence,

$$E_v \leq E_{u_1} + E_{u_2} \leq \mathcal{B}_{u_1}[i_{u_1}] + \mathcal{B}_{u_2}[i_{u_2}] \leq \max\{\mathcal{B}_{u_1}[j] + \mathcal{B}_{u_2}[k] : j + k = i_v\} = \mathcal{B}_v[i_v]$$

For the remaining Cases 2 and 3 we denote by i_u the flow of M^* along the exit edge of T_u and we assume, by the inductive hypothesis, that $E_u \leq \mathcal{B}_u[i_u]$.

Case 2 (node v is of type (c)). As at node v there are A_v new agents with

total energy e_v, we have $i_v \leq i_u + A_v$, otherwise flow M^* is not feasible. Therefore, by line 6 of the procedure we have

$$E_v \leq E_u + e_v \leq \mathcal{B}_u[i_u - A_v] + e_v = \mathcal{B}_v[i_v]$$

Case 3 (node v is of type (c)). Denote $w = weight(u, v)$. In order to show the bound on E_v, we need to evaluate the cost of energy $\tilde{E}$ spend by the agents traversing the edge $e = (u, v)$ in both directions. Clearly

$$\tilde{E} \geq w \cdot (f(u \to v) + f(v \to u))$$

There is at least one agent which has to traverse edge e in the direction from v to u, namely the agent which brings the broadcast packet to u. Therefore, $f(v \to u) \geq 1$. We will consider 3 sub-cases of Case 3.

Sub-case 3a ($i_u \geq 0$). In this situation, $i_u + 1$ agents have to traverse the edge $u \to v$, i.e. $f(u \to v) = i_u + 1$ and

$$\tilde{E} \geq w \cdot (f(u \to v) + f(v \to u)) = (|i_u| + 2) \cdot w$$

However, this case is treated at line 9 of procedure COMPUTE $\mathcal{B}(v)$, according to which

$$E_v = E_u - \tilde{E} \leq \mathcal{B}_u[i_v] - (i_u + 2) \cdot w = \mathcal{B}_v[i_v] \tag{6.1}$$

Sub-case 3b ($(i_u < 0) \wedge (\mathcal{B}_u[i_u] \geq 2w)$), i.e. there exists an amount of energy that is not needed to perform a local broadcasting inside T_u. If $f(u \to v) = 0$, then no agent can transport this energy so it may be subsequently potentially

transferred to the root r of T. In such a case this energy is lost. We can then assume that one agent transports the amount of $\mathcal{B}_u[i_u] - 2w$ units of energy from u to its parent v and later there is an agent that returns to u with the packet, so that the broadcasting inside T_u may be successfully completed. Note that such energy may later reach the root or lost (if the cost of its subsequent transfer is to large), but its transfer from u to v would never imply that the energy deposited at the root is diminished.

Consequently, we can assume that $f(u \to v) = 1$, hence $f(v \to u) = -i_u - 1$ and the equation (6.1) holds in this sub-case as well.

Sub-case 3c $((i_u < 0) \wedge (\mathcal{B}_u[i_u] < 2w))$. In this case, even if $\mathcal{B}_u[i_u] > 0$ the extra energy available at u cannot be transferred towards the root without a loss of energy deposited at the root. Indeed, as $i_u < 0$, an attempt to transfer energy from u to v results in $f(u \to v) \geq 1$, which in turn implies $\tilde{E} \geq |i_u + 2|w$. In such a case we would obtain

$$\mathcal{B}_v[i_v] \leq \mathcal{B}_u[i_u] - \tilde{E} \leq \mathcal{B}_u[i_u] - |i_u + 2|w < -|i_u| \cdot w = i_u \cdot w$$

where the last amount equals the energy potential computed for this case in line 10 of procedure COMPUTE $\mathcal{B}(v)$.

We conclude by induction that $E_r \leq \mathcal{B}_r[0]$, which completes the proof. □

6.3 Constructing broadcast schedule

Lemma 11 shows the upper bound on the amount of energy which may be deposited at the root of the tree from which a broadcast may be performed. We present now an algorithm generating a broadcasting schedule which succeeds in depositing such maximal amount of energy. Obviously, if this amount of energy

is negative, there exists no broadcasting schedule for the given tree. The rough idea of the algorithm is the following. Firstly, we compute an optimal M-flow of T. Then, using this M-flow, in each sub-tree, the excessive energy is transferred up the tree towards the root, deposited there and never used. Finally, a recursive procedure transferring the broadcast packet from the root r is called. This procedure performs the agents' transfer according to the optimal M-flow computed before. Interestingly, the algorithm performs four traversals of tree T, which are alternately bottom-up (1st and 3rd) and top-down (2nd and 4th). Each of them may be given as a recursive procedure, but, for clarity, we chose only the last one to be recursive. Before giving our algorithm we describe below its idea in more details. Because of the lack of space the proof of correctness of the algorithm is deferred to the Appendix.

In the first step of the algorithm we call procedure TEST-BROADCAST-FROM-ROOT(T) computing for each node v its table $\mathcal{B}_v$.

The second step performs a top-down traversal of tree T and, using the tables $\mathcal{B}$ compute an optimal distribution of agent flows between any given node and its children. Note that, when a node v is of type (b) for any already computed optimal flow i_v through the exit edge of tree T_v, we obtain feasible optimal flows i_{u_1} and i_{u_2}. This step computes the entire M-flow of tree T, which results in achieving the amount of $\mathcal{B}_r[0]$ energy deposited at the root. The rest of the algorithm refers to this M-flow.

The agents' moves forming an optimal broadcasting schedule are generated in the third and the fourth step of the algorithm. The third step generates the moves corresponding to the energy transfer edges T_e of the multigraph T_M. This step makes a bottom-up traversal of T and identifies the subtrees containing excessive energy, i.e. the energy which is not needed to perform the local broadcast inside the subtree. Such amounts of energy are moved up the tree by mobile agents and

the amount of $\mathcal{B}_r[0]$ energy is eventually accumulated at the root. Observe that a bottom-up traversal ensures that, whenever two or more different agents arrive to the same node, they wait until the last such agent appear at the node. Then a single agent collects the energy of all other agents and alone continues its upward walk. Note that, it may happen that some energy of the system may be lost (cf. energy of agent H from Fig. 6.3) because transferring it up the tree results in a bigger cost than the amount of energy to transport. Moreover, the set of edges T_e may be disconnected, hence some energy that is moved up might not reach the root. The edges used to transfer energy are marked, so that in the rest of the algorithm the number of agents traversing them according to the computed M-flow is diminished by 1.

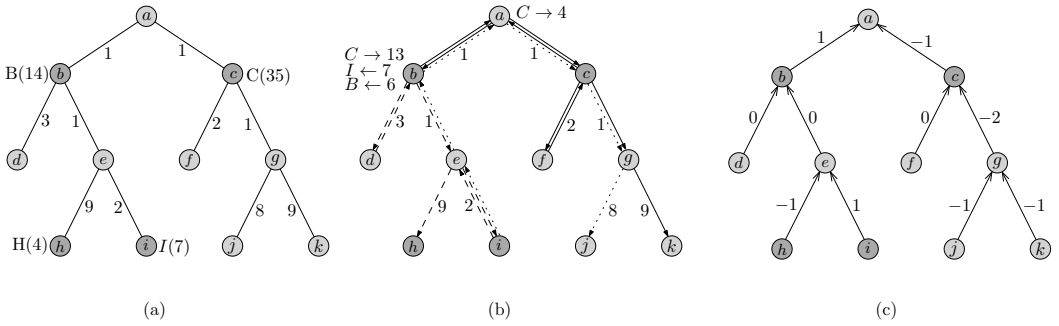

Figure 6.3: Example of a broadcast on a tree T from node a. (a) original tree with agents B, C, H, I and their initial energy levels, (b) trajectories of agents and energy transfers: the trajectory of agent C is given by normal arrows, while the paths of agents B and I are represented by dashed and dotted arrows, respectively. Agent H never moves; formula $X \to E$, attached to a node, means that agent X deposits an amount E of energy at this node, while $X \leftarrow E$ signifies that agent X picks up energy E when arriving at the node. Agent moves are set so that energy deposit and pick-up are synchronized (c) agents migration flow.

The final, fourth step starts when one of the agents walking up the tree reaches the root and deposits there the amount of $\mathcal{B}_r[0]$ energy. This agent starts the procedure of distributing it down the tree. The diffusion is performed by the

recursive procedure INFORM. When a recursive call of INFORM is made to a node with two children, a child with a larger flow is visited first. Indeed, it might be necessary to bring superfluous agents exiting from this child in order to use them in its sibling sub-tree.

ALGORITHM BROADCAST-IN-TREE(T);

STEP 1: TEST-BROADCAST-FROM-ROOT(T);

STEP 2: $r.Mindex := 0$;

for each edge $u \to v$ of T taken in top-down order **do**

$u.Mindex :=$ the value used to obtain $v.Mindex$ when computing $\mathcal{B}_v$ in TEST-BROADCAST-FROM-ROOT(T)

STEP 3: **for** each edge $u \to v$ of T taken in an bottom-up order **do**

if $\mathcal{B}_u[u.Mindex] > 0$ **then** MOVE-EXTRA-ENERGY-UP(u)

STEP 4: REMOVE-ENERGY($r, \mathcal{B}_r[0]$);

INFORM(r);

Theorem 10. *The optimal broadcasting on a tree of n nodes having k agents can be computed in $O(nk^2)$ time. The size of the optimal schedule is $O(nk)$.*

Proof. Observe that for any given tree of n nodes its converted version (cf. Fig. 6.4) is of size $\Theta(n)$ and it may be obtained in $O(n)$ time. The time complexity of algorithm TEST-BROADCAST-FROM-ROOT is dominated by the call of BROADCAST-IN-TREE(T), which calls function COMPUTE $\mathcal{B}$ $O(n)$ times. The function COMPUTE $\mathcal{B}$ consists of a for-loop executed $O(k)$ times. The most expensive case is when an iteration of the loop executes the max operation at line 4 (the case of node de type (b)) which takes $O(k)$ time, resulting in the overall time of $O(nk^2)$.

The complexity of for-loops from lines 2 and 4 of the algorithm TEST-BROADCAST-FROM-ROOT equal $O(n)$ as each iteration is carried in constant time. Finally, the recursive call to function INFORM in STEP 4 takes $O(nk)$ time. Indeed, there are

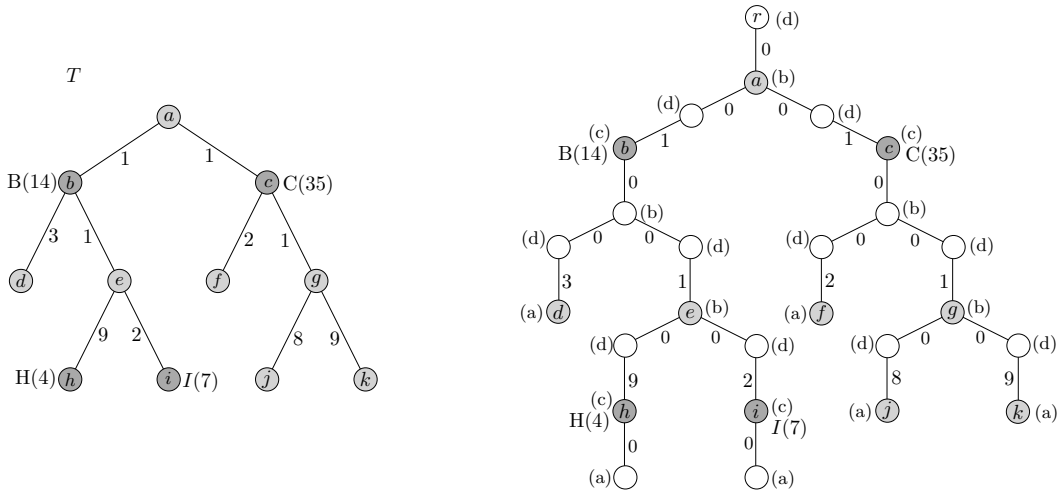

Figure 6.4: Tree T and its converted version. White nodes are added. At nodes b, c, h, i are present agents B, C, H, I having the initial amounts of energy $14, 35, 4, 7$, respectively. Nodes are marked according to their types. Corresponding weighted path lengths (between non-white nodes) are the same.

$O(n)$ total number of calls of function Inform. Every call to Inform executes functions Move-Up and Move-Down, each one containing a loop executed the number of times equal to the flow of the corresponding parameter. As each flow is bound by k, we have $O(nk)$ complexity of the function Inform and the same complexity for the number of agent moves generated by our algorithm. □

We pose as open questions the design of polynomial-time algorithms for the problem of broadcasting from a set of many source nodes and for the gossiping problem for trees (in energy exchange setting).

6.4 Proof of correctness of the algorithm BROADCAST-IN-TREE(T)

Procedure INFORM generates the sequence of agents' moves resulting in moving the packet from the root down to all other nodes. It also controls the migration of the agents according to the optimal M-flow computed earlier.

PROCEDURE INFORM(v);

1. **case** type of node v **of**
2. (a) **Exit** ;
3. (b) **if** $u_1.Mindex \geq u_2.Mindex$ **then** $f_1 := u_1, f_2 := u_2$
   else ($f_1 := u_2, f_2 := u_1$);
4. MOVE-DOWN(u_1); INFORM(u_1); MOVE-UP(u_1);
5. MOVE-DOWN(u_2); INFORM(u_2); MOVE-UP(u_2);
6. (c), (d) MOVE-DOWN(u); INFORM(u); MOVE-UP(u);

Procedure INFORM calls two procedures MOVE-DOWN(u) and MOVE-UP(u) to execute the travel according to the flow of the exit edge $u \to v$, for $v = parent(u)$. These procedures call the function MOVE, whose purpose is to generate the schedule of the agents' moves.

Procedure Procedure MOVE-EXTRA-ENERGY-UP(u) tests whether the excessive energy available in T_u is sufficiently large so that the transferring it along edge $u \to v$ is not too costly.

PROCEDURE MOVE-EXTRA-ENERGY-UP(u);

1. $v := parent(u)$; $w := weight(u, v)$; $i_u := u.Mindex$;
2. **if** $i_u < 0 \wedge \mathcal{B}_u(i_u) > 2w) \vee \mathcal{B}_u(i_u) > w$ **then**
3. MOVE-ENERGY(u); MARK-EDGE($u \to v$);

Procedures MOVE-DOWN(u) and MOVE-UP(u) generate the moves of the groups of agents, respectively, up and down a given edge $u \to parent(u)$. We assume that, when the sequence of such moves is made from the function MOVE-UP(u), or from the function MOVE-DOWN(u) for which u is an only child, they carry the entire energy available. Otherwise, if u has a sibling (i.e. it's parent is a node of type (b)), the energy is split according to the calls to MOVE-DOWN(u_1) and MOVE-DOWN(u_2). Procedure MOVE-ENERGY(u) creates a move of a single agent up the tree from node u, carrying entire energy (brought to u by all agents present there).

PROCEDURE MOVE-DOWN(u);

1. $v := parent(u)$; $i_u := u.Mindex$;
2. MOVE($v \to u$); {packet transfer move}
3. **if** edge $u \to v$ marked **then** $e := -i_u$ **else** $e := -(i_u + 1)$;
4. **for** $i := 1$ **to** e **do** MOVE($v \to u$); {moves of agents beside the packet-transferring one}

PROCEDURE MOVE-UP(u);

1. $v := parent(u)$; $i_u := u.Mindex$;
2. **if** edge $u \to v$ marked **then** $e := i_u - 1$ **else** $e := i_u$;
   {e agents to exit T_u; if edge marked, an energy-transferring agent moved earlier}
3. **for** $i := 1$ **to** e **do** MOVE($u \to v$); {agents exiting according to M-flow}

Lemma 12. *The excessive energy of $\mathcal{B}_r[0]$ is accumulated at the root r at the completion of STEP 2 of algorithm* BROADCAST-IN-TREE.

Proof. We prove that, after each iteration of the for-loop from STEP 2, the maximal energy $\mathcal{B}_x[x.Mindex]$, available inside T_x but not used for the broadcast from x into T_x, is present at node x. Moreover, if $\mathcal{B}_x[x.Mindex] > 0$ at least one agent

is present at x. The prove goes by induction on the height of T_x. The statement is clearly true for x being a leaf of T. Suppose that the statement of the lemma is true for trees of height h. Take edge $x \to y$ leading to node y of height $h+1$. If $\mathcal{B}_x[x.Mindex] \leq w = weight(x,y)$ then no energy may be transferred to node y, as the cost of this transfer equals w. Otherwise, if $i = x.Mindex < 0$, two cases are possible: either i agents enter T_x and no agent exits it, so no energy transfer along edge $x \to y$ is made, or, one agent exits T_x transporting excessive energy along edge $x \to y$ and $i+1$ agents enter T_x. The total cost of all traversals of edge $x \to y$ is then iw in the former case and $(i+2)w$ in the latter one. The transfer of such energy is then profitable only in the case when it exceeds $2w$, otherwise it is lost and never used in the schedule. This is exactly what is done in lines 2 and 3 of procedure MOVE-EXTRA-ENERGY-UP. Observe that, as $\mathcal{B}_x[x.Mindex] > 0$, by inductive assumption, the energy required for the transfer is already available at x as well as an agent necessary to perform the transfer is present at x. □

Lemma 13. *All agent actions generated by the calls of procedure* INFORM(y) *are feasible, i.e. when an agent move is generated along an edge $x \to y$ (or along an edge $y \to x$) then, there is an agent available at node x (or y) and there is at least $weight(x,y)$ energy available at x (or y), so that such move may be successfully completed.*

Proof. All agents' moves generated from a call to INFORM(x) are the moves down or up the tree T. We show first that all moves down the tree T, i.e. those generated through the procedure MOVE-DOWN, are feasible. The proof goes by induction on the depth of x. The claim of the lemma is clearly true for x being the root r as it has an agent (which brought excessive energy to it) and the weight of the edge incident to it is zero (by construction). Consider now x being any other node. By the inductive hypothesis, the claim of the lemma is true for node y, the parent of x, therefore there is an agent present at x and enough energy to reach y.

We show now that all moves up the tree T (i.e. those generated through the procedure MOVE-UP(x)) are feasible. The proof goes by induction on the height of x. The claim of the lemma is clearly true when x is a leaf. Consider now x being any non-leaf node and suppose, by the inductive hypothesis, that the claim of the lemma is true for node x. Denote $y = parent(x)$. The claim of the lemma is clearly true for y if no agent needs to go along the edge $x \to y$ according to the M-flow used, or if the only agent traversing edge $x \to y$ was the energy-transferring agent. Otherwise, since the energy available at node y at the moment of the call to INFORM(y) was sufficient to broadcast in T_y according to the M-flow computed, all the moves in procedure MOVE-UP are feasible. □

Theorem 11. *If $\mathcal{B}_r[0] \geq 0$, the algorithm* BROADCAST-IN-TREE *produces a correct broadcasting schedule, which deposits $\mathcal{B}_r[0]$ energy at the root of T.*

Proof. By Lemma 12 the excessive energy $\mathcal{B}_r[0]$ is indeed accumulated at the root r at the end of the for-loop from STEP 2. This energy is removed by the function REMOVE-ENERGY.

The remaining part of the algorithm is done by the function INFORM. By Lemma 13, all agent moves generated from the main call of INFORM(r) are feasible. Observe, that the sequence of produced agent moves contains a subsequence which results in transferring the information from the root to all other nodes. Indeed, procedure INFORM is called for the root r and then recursively for all other nodes of the tree in the top-down order. Each call of INFORM to any node v contains calls to the children of v, at the same time generating agent moves along edges leading to these children. This results in the transfer of the broadcast information to all nodes. This completes the proof. □

Chapter 7

Final remarks

This thesis was interested in communication protocols realized in tree networks with aid of energy-exchanging mobile agents. The considered communication protocols were *data delivery* (realizing one-to-one communication), *convergecast* (i.e. many-to-one communication) and *broadcast* (one-to-many communication). We designed efficient algorithms verifying if an input configuration of agents with their initial energy levels permits to realize a given communication protocol. In each case, such a decision problem has been realized by conversion to an optimization problem, in which our goal was to deposit a maximal amount of energy in the root of the tree.

Similar problems, studied in operations research (particularly in vehicle routing), sometimes assume limited capacities of robots and quantities of product to be transported. In our case, the amount of product to be carried by a robot is irrelevant. Consequently, similarly to [35], we categorize our problem as *data communication* using mobile agents.

Below we attempt to observe that some slight modifications of the settings analyzed in this thesis may lead to obvious positive results, i.e. relatively elementary algorithms. On the other hand, some other slight settings modifications imply

negative results, which is the NP-hardness of the problem. We also discuss some variations of the related problems that may lead to interesting open questions.

It is somewhat remarkable that, without energy exchange, even the simplest problem of data delivery is NP-complete in the simplest environment of the line, (cf. [35]), while, as we have shown in Chapter 4, considered communication problems with energy exchange are solvable in linear time even for tree networks. On the other hand, it is not surprising that energy exchange in general graphs does not help and the problems are NP-complete.

The broadcasting algorithm, presented in Chapter 5, permits to generate an efficient data dissemination in a tree network using mobile agents. In the case when the source node (containing the packet to be broadcast) coincides with the root of the tree (at which the agents are initially placed) our problem is equivalent to the tree exploration problem. Surprisingly this version of the tree exploration, which needs to optimize the total energy used, has not been investigated in the past.

Observe that, if the agents exploring the tree, starting from the same node, are required to return to their initial position, the optimal-energy solution would be obtained trivially by using only one agent performing a standard depth-first search of the tree. Indeed, each tree edge has to be traversed at least twice (at least once in each direction), hence single-agent DFS solution is optimal. The technique presented in Chapter 5 exploits the fact that our agents need not return to their home base. Consequently, every agent used in the exploration process may finish its trajectory at some other node of the tree. The main difficulty was to find a subset of tree nodes at which the agents may end their routes. Sometimes such set of nodes turn up to be of the size smaller than the number of available agents and some agents are not to be activated.

An interested reader may observe that, in the case of general graphs rather than trees, the exploration problem optimizing total energy used is NP-hard already for a single agent case by reduction from the Hamiltonian path problem. Indeed, the decision question whether an n-node graph having unit-weight edges may be explored by an agent using $n-1$ units of energy is equivalent to the existence of Hamiltonian path starting from the source node.

If the case when the number of available agents is at least equal to the number of leaves of the tree our algorithm has $O(n)$ time complexity for an n-node tree network. This is clearly optimal as each tree node needs to be taken into consideration by the algorithm. In case when the number of agents is smaller than the number of tree leaves, our algorithm works in $O(n \log n)$ time. We leave it as an open problem whether this time complexity may be improved over the obvious lower bound of $\Omega(n)$. We conjecture that, when $k < o(n)$ the above lower and upper bounds (as functions of parameters n and k) may be both improved.

The main question asked in our research for energy-exchanging agents was to decide, whether the agents can schedule their moves and data transfers resulting in transporting data packets successfully. Nevertheless, in the model studied in Chapter 5, the scheduling aspect of the agents' moves is not really relevant, as the agents can move independently. Indeed, the agents should share the available energy before the start of their movements so that the subsequent broadcast is possible and then each agent continues its walk without any synchronization with the movements of other agents. It is worth noting that the same problem for agents that cannot share energy is hard. In particular, it has been shown in [35] that if a collection of agents is distributed along a line, each of them having some amount of energy that it cannot share with other agents, the decision problem whether the agents can collectively transport a data packet between two given points of the network, is proven to be NP-hard.

There are several other open questions related to the communication problems and data delivery for mobile agents. An interesting open problem is to design a schedule for energy-exchanging agents resulting in gossiping, i.e. when data packets initially present at the tree nodes need to be disseminated so they reach all other nodes of the tree. Another variant of this problem is to design an algorithm generating the schedule for energy-exchanging agents, when data packets initially present at some *source* nodes need to reach some specific *target* nodes.

When the initial amounts of energy are a priori assigned and the agents cannot share their energy, most communication problems for mobile agents are shown to be NP-complete (cf. [6, 35]). However, when the assignment of energy levels to the agents is left to the algorithm, minimization of total energy used for the communication problems remains open.

Another possibility for open problems arise when we can use non-homogenous mobile agents. However, when different maximal speeds are considered, problems concerning mobile agents often turn up to be hard and the research interest corresponding to them is relatively limited. Moreover, where agents' speeds are taken into consideration, the researchers seek to optimize time rather than energy used. As for many such problems, it is necessary to divide the tasks between the agents that often leads to NP-hardness (by reduction from partition) even for same-speed agents, e.g., see [78]. However it may be interesting to try to extend the results of the present paper to the case when the agents do not have the same rate of energy consumption. Such extension may be easier to obtain for the case of agents starting from the same node (the case of the present paper) rather than for agents initially distributed over the tree nodes (cf. [44]).

Our approach from Chapter 6 may be extended to compute within the same time bound all tree nodes from which the broadcast may be performed. We may show the following

Theorem 12. *In $O(nk^2)$ time we can find all nodes from which we can perform a successful broadcasting in a given tree T of n nodes with k agents. Moreover, for each node x of T we can compute the maximal amount of energy which may be left at x while performing a broadcast from it.*

Proof. (sketch). Consider a converted version of T. The algorithm Broadcast-in-Tree(T) computes the optimal energy, which can be deposited by the successful broadcast at its root r. The algorithm considers the set of directed edges $u \to v$, for all nodes u, v such that the path from u to r in T contains v. The algorithm computes tables $\mathcal{B}_u$ performing a bottom-up traversal of this set of edges. Observe that we can consider the set of all directed edges of the tree (in both directions), which is only twice larger. It is easy to see, that this set may be ordered so that for any edge $u \to v$, each edge $y \to u$, for $u \neq y$ is earlier in this order. We can run slightly modified our algorithm Broadcast-in-Tree for such set of edges. Consequently, for any node x of the tree, we will have available the tables $\mathcal{B}_u$ for u being children of x (x has at most two children for the converted version of T). Having them, in constant time we can compute the largest energy which may be delivered to x, from which the broadcast needs to be performed. □

The approach presented in Chapter 6 is the most involved among all the techniques studied in this thesis. Consequently, we believe that one of the most interesting remaining open problems is to improve the $O(nk^2)$ time complexity of the algorithm generating the broadcasting schedule for energy-exchanging agents, initially dispersed in the different nodes of the tree network.

The optimisation criterion studied in the present thesis was to maximize the total amount of energy, which can be deposited at the root of the tree, while the studied communication protocol is still successfully terminated. The time at which the protocol might be concluded (which equals to maximal energy used by the activated agents) is not studied. An interesting open problem is to try to

minimize the time of the generated schedule, while still staying within the smallest possible total amount of energy.

The above mentioned problem may be possibly solved by scheduling the agents' movements so they walk in parallel, while they still meet to perform the necessary energy transfers. This may be possibly done by "merging" the moves of the agents by adding minimal waiting times for some of them, which allows the meetings necessary for the energy transfers. More interesting open problem concerning minimization of the schedule time may arise if the total energy used is not to be optimized. For example, consider the case when the initial energy levels of available agents are sufficient to realize a communication protocol while some (surplus) amount of energy may be deposited at the root. How can such surplus of energy can be used in order to minimize the time of the schedule.

Several variants of data delivery problems have been studied in the PhD thesis of Bärtschi (see [15]), where the proposed algorithms concerned agents having possibly different data-carrying capacities, different rates of energy consumption and they were analyzed under various optimization criteria. In all scenarios investigated in [15] the agents could not exchange energy between them. Possibly some of these problems might be also studied under the energy-exchanging scenarios.

Bibliography

[1] S. Albers: Energy-efficient algorithms. *Comm. ACM 53(5)*, 86-96 (2010).

[2] S. Albers and M. R. Henzinger: Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164-1188, (2000).

[3] S. Alpern and S. Gal: *The theory of search games and rendezvous.* Springer, (2003).

[4] C. Ambühl, L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang: Tree exploration with logarithmic memory. *ACM Trans. Algorithms 7(4)*, 1–21 (2011).

[5] J. Anaya, J. Chalopin, J. Czyzowicz, A. Labourel, A. Pelc, and Y. Vaxès: Collecting information by power-aware mobile agents. In: *Aguilera, M.K. (ed.) DISC 2012. LNCS*, vol. 7611, pp. 46–60. Springer, Heidelberg (2012).

[6] J. Anaya, J. Chalopin, J. Czyzowicz, A. Labourel, A. Pelc, and Y. Vaxès: Convergecast and Broadcast by Power-Aware Mobile Agents. *Volume 74, Issue 1*, pp 117-155, January (2016).

[7] V. Annamalai, S.K.S. Gupta, and L. Schwiebert: On tree-based convergecasting in wireless sensor networks. *IEEE Wirel. Commun. Netw. 3*, 1942-1947 (2003).

[8] I. Averbakh, and O. Berman: A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree. *Discr. Appl. Math. 68*, 17–32 (1996).

[9] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish: A trade-off between information and communication in broadcast protocols. *J. ACM 37(2)*, 238-256 (1990).

[10] R. Baeza Yates and J. Culberson, and G.Rawlins: Searching in the plane. *Information and Computation*, 106(2):234-252, (1993).

[11] R. Baeza Yates and R. Schott: Parallel searching in the plane. *Computational Geometry*, 5(3):143-154, (1995).

[12] E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, and R. Klasing: Beachcombing on strips and islands. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, Pages 155-168. Springer, (2015).

[13] E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pajak: Linear search by a pair of distinct-speed robots. In *SIROCCO*, pages 195-211. LNCS, (2016).

[14] E. Bampas, S. Das, D. Dereniowski, and C. Karousatou: Collaborative Delivery by Energy-Sharing Low-Power Mobile Robots. *ALGOSENSORS,* 1-12 (2017).

[15] A. Bärtschi: Efficient Delivery with Mobile Agents *(Phd Thesis), ETH Zurich,* (2017).

[16] A. Bärtschi, J. Chalopin, S. Das, Y. Disser, B. Geissmann, D. Graf, A. Labourel, and M. Mihalák: Collaborative delivery with energy-constrained

mobile robots. In: *Suomela, J. (ed.) SIROCCO 2016. LNCS*, vol. 9988, pp. 258–274. Springer, Cham (2016).

[17] A. Bärtschi, J. Chalopin, S. Das, Y. Disser, D. Graf, J. Hackfeld, and P. Penna: Energy-efficient delivery by heterogeneous mobile agents. In: *Proceedings of STACS*, pp. 10:1–10:14 (2017).

[18] R. Bar-Yehuda, O. Goldreich, A. Itai: On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. *J. Comput. Syst. Sci. 45(1)*, 104-126 (1992).

[19] A. Bärtschi, D. Graf, and M. Mihalák: Collective fast delivery by energy-efficient agents. *Unpublished manuscript,* (2017).

[20] A. Bärtschi, and T. Tscharger: Energy-Efficient Fast Delivery by Mobile Agents. In *21st International Symposium on Fundamentals of Computation Theory FCT' (2017), pages 82-95, (2017).*

[21] J. Beauquier, J. Burman, J. Clement, and S. Kutten: On utilizing speed in networks of mobile agents. In *Proceeding of the 29th* ACM SIGACT-SIGOPS *Symposium on Principles of Distributed Computing,* pages 305-314. ACM, (2010).

[22] A. Beck. On the linear search problem: *Israel Journal of Mathematics*, 2(4):221-228, (1964).

[23] R. Bellman: An optimal search. *SIAM Review,* 5(3):274-274, (1963).

[24] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan: The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the thirtieth annual* ACM *symposium on Theory of computing*, pages 269-278. ACM, (1998).

[25] M. A. Bender and D. K. Slonim: The power of team exploration. Two robots can learn unlabeled directed graphs. In *Foundations of computer science, Proc. 35th annual symposium on*, pages 75-85. IEEE, (1994).

[26] A. Bonato and R. Nowakowski: *The game of cops and robbers on graphs.* AMS, (2011).

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

[28] S. Brandt, F. Laufenberg, Y. Lv, D. Stolz, and R. Wattenhofer: Collaboration without communication: Evacuating two robots from a disk. In *Algorithms and Complexity - 10 th International Conference*, CIAC *2017, Athens, Greece, May 24-26, 2017. Proceedings*, (2017).

[29] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao: *Multirobot tree and graph exploration. IEEE Transactions on Robotics 27(4)*, 707-717, (2011).

[30] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun: Collaborative multirobots exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, pages 476-481. IEEE, (2000).

[31] W. Burgard, M Moors, and F. Schneider. Collaborative exploration of unknown environments with team of mobile robots. In *Advances in plan-based control of robotics agents*, pages 52-70, Springer, (2002).

[32] G. Cabri, L. Leonardi, and F. Zambonelli: Mobile-agent coordination models for internet applications. *Computer (Volume: 33, Issue: 2)*, pages 82-89, Feb (2000).

[33] J. Chalopin, S. Das, and A. Kosowski: Constructing a map of an anonymous graph: Applications of universal sequences. In *International Conference On Principles Of Distributed Systems*, pages 119-134. Springer, (2010).

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

[35] J. Chalopin, R. Jacob, M. Mihalák, and P. Widmayer: Data delivery by energy-constrained mobile agents on a line. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) *ICALP 2014*, Part II. LNCS, vol. 8573, pp. 423–434. Springer, Heidelberg (2014).

[36] M. Chrobak, L. Gasieniec, Gorry T., and R. Martin: Group search on the line. In *Proceedings of SOFSEM 2015, LNCS 8939*, pages 164-176. Springer, (2015).

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

[38] R. Cohen and D. Peleg: Convergence of autonomous mobile robots with inaccurate sensors and movements. *SIAM Journal on Computing*, 38(1):276-302, (2008).

[39] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, R. Martin, and O. Morales Ponce: Optimal patrolling of fragmented boundaries. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 241-250. ACM, (2013).

[40] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, and A. Labourel: Tell me where I am so I can meet you sooner. In *International Colloquium on Automata, Languages, and Programming*, pages 502-514. Springer, (2010).

[41] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, and R. Martin: Synchronous rendezvous for location-aware agents. In *International Symposium on Distributed Computing*, pages 447-459, Springer, (2011).

[42] J. Czyzowicz, K. Diks, J. Moussi, and W. Rytter: Communication problems for mobile agents exchanging energy. In: *Suomela, J. (ed.) SIROCCO 2016. LNCS*, vol. 9988, pp. 275–288. Springer, (2016).

[43] J. Czyzowicz , K. Diks, J. Moussi , and W. Rytter: Energy-Optimal Broadcast in a tree with Mobile Agents. *ALGOSENSORS*, pp. 98-113 (2017).

[44] J. Czyzowicz , K. Diks, J. Moussi , and W. Rytter: Broadcast with Energy-Exchanging Mobile Agents Distributed on a Tree. *SIROCCO*, pp. 209-225 (2018).

[45] J. Czyzowicz, S.Dobrev, K. Georgiou, E. Kranakis, and F. MacQuarrie: Evacuating two robots from multiple unknown exits in a circle. *Theoretical Computer Science 709:*, pages 20-30. (2018).

[46] J. Czyzowicz, L. Gasieniec, K. Georgiou, E. Kranakis, and F. MacQuarrie: The beachcomber's problem: walking and searching with mobile robots. *Theoretical Computer Science*, 608:201-218, (2015).

[47] J. Czyzowicz, L. Gasieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak: Evacuation robots from an unknown exit located on the perimeter of a disc. In *DISC 2014*. Springer, Austin, Texas, (2014).

[48] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis: Boundary patrolling by mobile agents with distinct maximal speeds. In *ESA*, pages 701-712, (2011).

[49] J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, and N. Taleb: When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. In *Algorithms and Computation - Proc. 26th international Symposium, ISAAC 2015*, pages 343-354, (2015).

[50] J. Czyzowicz, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny and S. Shende: Search on a line by byzantine robots. In *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, (2016), Sydney, Australia*, pages 27:1-27:12, (2016).

[51] J. Czyzowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny and B.Vogtenhuber: Evacuation robots from a disc using face to face communication. In *Proceeding of CIAC 2015, LNCS volume 9079*, pages 140-152. Springer, Paris, France, (2015).

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

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

[54] J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, and J. Opatrny: Search on a line with faulty robots. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, (2016)*, pages 405-414, (2016).

[55] J. Czyzowicz, E. Kranakis, D. Pajak, and N. Taleb: Patrolling by robots equipped with visibility. In *International Colloquium on Structural Information and Communication Complexity*, pages 224-234. Springer, (2014).

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

[57] S. Das, D. Dereniowski, C. Karousatou: Collaborative exploration by energy-constrained mobile robots. In: *Proceedings of SIROCCO*, pages 357-369 (2015).

[58] S. Das, D. Dereniowski, and C. Karousatou: Collaborative Exploration of trees by Energy-Constrained Mobile Robots. *Theory Comput. Syst.* 62(5): 1223-1240 (2018).

[59] S. Das, D. Dereniowski, and P. Uznanski: Brief Announcement: Energy Constrained Depth First Search. *ICALP:* 165:1-165:5 (2018).

[60] S. Das, P. Flocchini, A. Nayak, and N. Santoro: Distributed exploration of an unknown graph. In *International Colloquium on Structural Information and Communication Complexity*, pages 99-114. Springer, (2005).

[61] S. Das, P. Flocchini, N. Santoro, and M. Yamashita: Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing* 28(2): 131-145 (2015).

[62] E. D. Demaine, S. P. Fekete, and S. Gal: Online searching with turn cost. *Theoretical Computer Science*, 361(2):342-355, (2006).

[63] X. Deng, T. Kameda, and C. Papadimitriou: How to learn an unknown environment. In *Proceedings of FOCS*, pages 298-303. IEEE, (1991).

[64] X. Deng, and C.H. Papadimitriou: Exploring an unknown graph. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 355-361. IEEE, (1990).

[65] D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznanski: Fast collaborative graph exploration. *Information and Computation*, 243:37-49, (2015).

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

[67] Y. Dieudonné, A. Pelc. and D. Peleg: Gathering despite mischief. *ACM Transactions on Algorithms (TALG)*, 11(1), (2014).

[68] Y. Dieudonné, A. Pelc. and V. Villain: How to meet asynchronously at polynomial cost. *SIAM Journal on Computing*, 44(3):844-867, (2015).

[69] M. Dynia, M. Korzeniowski, and C. Schindelhauer: Power-aware collective tree exploration. In: *Grass, W., Sick, B., Waldschmidt, K. (eds.) ARCS 2006. LNCS*, vol. 3894, pp. 341–351. Springer, Heidelberg (2006).

[70] M. Dynia, J. Kutylowski, F. Meyer auf der Heide, and C. Schindelhauer: Smart robot teams exploring sparse trees. In: Královič, R., Urzyczyn, *P. (eds) MFCS 2006. LNCS,* vol 4162, pp. 327-338. Springer, Heidelberg (2006).

[71] M. Dynia, J. Lopuszanski, and C. Schindelhauer: Why robots need maps. In: G. Prencipe, S. Zaks (eds.), SIROCCO 2007. *LNCS, vol 4474,* pp. 41-50, Springer, Heidelberg (2007).

[72] P. Flocchini: Time-varying graphs and dynamic networks. In *2015 Summer Solstice: 7th International Conference on discrete Models of Complex Systems*, (2015).

[73] P. Flocchini, D. Ilinkas, A. Pelc, and N. Santoro: Computing without communicating: Ring exploration by asynchronous oblivious robots. In *International Conference On Principles Of Distributed Systems*, pages 105-118. Springer (2007).

[74] P. Flocchini, B. Mans, and N. Santoro: Exploration of periodically varying graphs. In *International Symposium on Algorithms and Computation*, pages 534-543, Springer (2009).

[75] P. Flocchini, G. Prencipe, N. Santoro, and G. Viglietta: Distributed computing by mobile robots: uniform circle formation. *Distributed Computing 30(6)*: 413-457 (2017).

[76] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer: Gathering of asynchronous robots with limited visibility. *Theretical Computer Science*, 337(1):147-168, (2005).

[77] F. V. Fomin and D.M. Thilikos: An annotated bibliography on guaranteed graph searching. *Theretical Computer Science*, 399(3):236-245, (2008).

[78] P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc. Collective tree exploration. *Networks* (2006), 48(3), pp. 166-177.

[79] A. Frieze, M. Krivelevich, and P. Loh: Variations on cops and robbers. *Journal of Graph Theory*, 69(4):383-402, (2012).

[80] R.H. Glitho, E. Olougouna, and S. Pierre: Mobile agents and their use for information retrieval: a brief overview and an elaborate case study - *IEEE Network, 16:1*, pages 34-41. (2002).

[81] Y. Higashikawa, N. Katoh, S. Langerman, and S.-I. Tanigawa: Online graph exploration algorithms for cycles and trees by multiple searchers. *Journal of Combinatorial Optimization*, 28:2, pages 480-495 (2014).

[82] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel: The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577-600, (2001).

[83] J. Hromkovič, R. Klasing, B. Monien, and R. Peine: Dissemination of information in interconnection networks (broadcasting & gossiping). In *Combinatorial networks theory*, pages 125-212. Springer, (1996).

[84] S. Irani, S.K. Shukla, R. Gupta: Algorithms for power savings. *ACM Trans. Algorithms TALG 3(4)*, 41 (2007).

[85] A. Kawamura, and Y. Kobayashi: Fence patrolling by mobile agents with distinct speeds. In *Distributed Computing*, Volume 28:2, pages 147-154, (2015).

[86] A. Kesselman, and D.R. Kowalski: Fast distributed algorithm for convergecast in adhoc geometric radio networks. *J. Parallel Distrib. Comput. 66(4)*, 578-585 (2006).

[87] J. Kleinber: On-line search problem in a simple polygon. In *Proceesings of SODA*, pages 8-15. SIAM, (1994).

[88] A. Kupavskii, E. Welzl: Lower Bounds for Searching Robots, some Faulty. *PODC*, pages 447-453, (2018).

[89] I. Lamprou, R. Martin, and S. Schewe: Fast two-robot disk evacuation with wireless communication. In *International Symposium on Distributed Computing*, pages 1-15. Springer, (2016).

[90] A. Lemieux: Exploration d'un anneau par les robots mobiles: *MSc, Computer Sciences, UQO,* (2017).

[91] F. Lessard: Exploration optimale d'un segment de droite par deux agents mobiles: *MSc, Computer Sciences, UQO,* (2013).

[92] G. De Macro, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro: Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science*, 355(3):315-326, (2006).

[93] A. Mallios, P. Ridao, D. Ribas, F. Maurelli , and Y. Petillot: EKF-SLAM for AUV navigation under probabilistic sonar scan-matching, *IEEE/RSJ, International Conference on Intelligent Robots and Systems*, (2010).

[94] N. Megow, K. Mehlhorn, and P. Schweitzer: Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science*, 463:62-72, (2012).

[95] E.W.T. Ngaia, and A.Gunasekaranb: A review for mobile commerce research and applications. Decision Support Systems *Volume 43, Issue 1, February (2007), Pages 3-15.*

[96] A. Oliver, S. Kang, B. Wunsche and B. MacDonald: Using the Kinect as a navigation sensor for mobile robotics. *Proceedings of the 27th Conference on Image and Vision Computing New Zealand.* Pages 509-514, (2012).

[97] B. Orgun, and J. Vu: HL7 ontology and mobile agents for interoperability in heterogeneous medical information systems. *Computers in biology and medicine*, Vol. 36:7-8, pages 817-836. (2006).

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

[99] G. Prencipe: Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2-3):222-231, (2007).

[100] R. Rajagopalan, and P.K. Varshney: Data-aggregation techniques in sensor networks: a survey. Syracuse University SURFACE, Pages 1-30, (2006).

[101] O. Reingold: Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, (2008).

[102] A. Ta-Shma and U. Zwick: Deterministic rendezvous, treasure hunts and strongly universal exploration sequences. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 599-608. Society for Industrial and Applied Mathematics, (2007).

[103] P. Toth, and D. Vigo: Vehicle routing: problems, methods, and applications. *SIAM* (2014).

[104] P.M. Vieira-Marques, S. Robles, and J. Cucurull: Secure integration of distributed medical data using mobile agents. *IEEE Intelligent Systems* (Volume: 21, Issue: 6, Nov.-Dec. (2006)).

[105] T. Wall, and T. Monahan: Surveillance and violence from afar. *The politics of drones and liminal security-scapes*. Volume: 15 issue: 3, page(s): 239-254. Article first published online: August 17, (2011); Issue published: August 1, (2011).

[106] M. Yamashita and I. Suzuki: Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28):2433-2453, (2010).

[107] Y. Zhang, M. Qiu, and C.W. Tsai: Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data. *IEEE Systems Journal, volume: 11, Issue: 1, pages: 88 - 95*, March (2017).

[108] https://www.robotics.org/content-detail.cfm/industrial-robotics-industry-insights/robotics-in-security-and-military-applications/content_ id/3112. *Collaborative robots. Advanced vision technologies. Artificial intelligence.*

[109] https://www.techemergence.com/military-robotics-innovation/. *Military Robotics Innovation - Comparing the US to Other Major Powers.*

[110] https://en.wikipedia.org/wiki/DARPA. *Defense Advanced Research Projects Agency (DARPA)*

[111] https://futirism.com/nine-most-amazing-darpa-projects/. *DARPA Projects.*

[112] https://ieeexplore.ieee.org/abstract/document/4287130/. *Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty.*

[113] https://www.sciencedirect.com/science/article/pii/S1367578816300219. *Unmanned surface vehicles: An overview of developments and challenges.*

[114] https://ieeexplore.ieee.org/abstract/document/790796/. *Mobile agents for networked electronic trading.*

[115] https://www.igi-global.com/chapter/mobile-agents-commerce/41246. *Encyclopedia of E-Business Development and Management in the Global Economy, mobile-agents-commerce.*

[116] https://ieeexplore.ieee.org/abstract/document/980543/. *Mobile agents and their use for information retrieval.*