NOTE TO USERS

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES
ET POSTDOCTORALES

FACULTY OF GRADUATE AND
POSTDOCTORAL STUDIES

Hamane BÉCHA

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Distributed Mobile Agents : Classification of Models trough the Construction of Topological Information

P. Flocchini

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

A. Nayak

N. Santoro

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES
SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE
AND POSTDOCTORAL STUDIES

Distributed Mobile Agents: Classification of Models through the Construction of Topological Information

by

Hanane Becha

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Supervisor: **Dr. Paola Flocchini**

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 0-494-01413-X
Our file Notre référence
ISBN: 0-494-01413-X

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

Distributed Mobile Agents: Classification of Models through the Construction of Topological Information

submitted by

Hanane Becha

In partial fulfilment of
the requirements for the degree of
Master in Computer Science

Paola Flocchini, Ph.D.

(Thesis Supervisor)

Ottawa University

Abstract

The goal of this thesis is to make a classification of the existing models for mobile agents and to study the relationship between them.

To do that, we consider the problem of collaboratively constructing some topological information and we study this problem under a variety of assumptions on the capabilities of the agents and on the network structure.

Distributed models for mobile agents differ from one another on a large number of parameters which can be grouped into two interdependent categories: structure surrounding the agents (e.g. structure of the topology, communication methods, etc.) and the capabilities of the agents (e.g. its a-priori knowledge, etc.). These parameters discern the common traits of the distributed environment problems and we use them to systematically classify the models used in the literature.

More precisely, we study the problem of determining the size of the network in some common topologies (rings, meshes, tori) and, through this example, we compare the power of the different models.

Acknowledgments

I was particularly fortunate to have Dr. Paola Flocchini as the supervisor of my Master thesis. I have benefited tremendously from her invaluable support and patient guidance at every stage of my master. Therefore, I would like to take this opportunity to express my sincere gratitude and esteem to her. I also thank professor Nicola Santoro that provided me with background, stimulating discussions, and inspiration for my research area.

I am grateful to Dr. Pelc Andrzej for his help and advice.

I would like to express my deepest feeling to all my family, I am grateful to all of them without exception, and especially my parents Ahmad Becha and Mongia Geuldiche, who believe in me, trust me and help me unconditionally.

As well, I thank my uncle Bilel Jamoussi who is my role model who helped me through my career to get this opportunity.

Finally I would like to thank my colleagues and friends for being around me and cheering me up. Thank you very much all those who were able to add magical touch to my life and create an unbreakable links between them and me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The daily rapid growth of networks requires new tools to help the users to manage the increasing distributed dynamic and heterogeneous resources. Intelligent software agent platforms represent a promising instrument to deal more efficiently and elegantly with this open environment and to undertake work in the network on behalf of the user.
There is no unique definition for an intelligent software agent. Based on the specific needs, it exists in the literature a variety of definitions. At the highest level, agents can be categorized as shown in figure 1.1:

The main difference between hardware and software agents is the under layer describing their

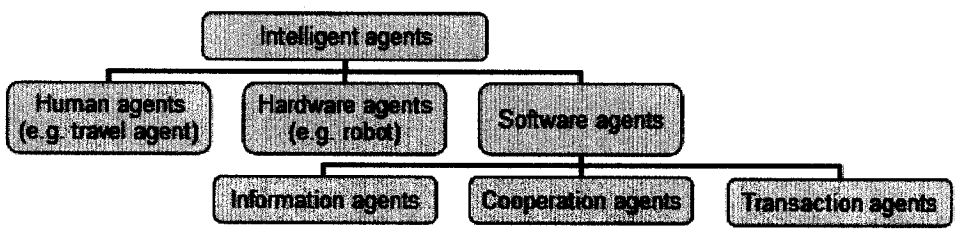

Figure 1.1: Classification of intelligent agents, [11], p19

moves. Hardware agents have limited surface as an under layer for their moves, while software agents have graphs that limit their movements only on edges. In this thesis we consider only software agents.

Depending on the needs of the user, software agents can have different specializations. Figure 1.1 classifies the software agents as: information agents, transaction agents, and cooperation agents; this classification is based on the area of application. An information agent is deployed for information retrieval in distributed systems. Using its knowledge about all the available resources, the information agent has to locate the information, extract it, filter it based on the interest of the user and return the result in an appropriate form. A transaction agent is used in sensitive areas such as database environments and electronic commerce. Its focus is processing and monitoring the requests. It reduces the network load and overcomes its latencies. Trustworthiness and confidentiality are major concerns for the transaction agent. Cooperation agent cooperates within a goal oriented group. It works out problems exceeding the capabilities of a single agent or it permits faster and better solutions than a single agent system.
To meet given requirements of an application, the deployed agent can belong at the same time to more than one category. In spite of the categories, commonly, an intelligent agent exploits its environment, collects and computes information, makes decisions on its own and reacts to achieve its goal. Common basic characteristics of agents are: autonomy, learning, reactivity, pro-activity, and computing capability.

- Autonomy: an agent is autonomous since it decides on its own what action to perform; it can adapt to new circumstances, without receiving commands or approval from its environment or from other agents.

- Learning: an agent changes its behavior based on its previous experience.

- Reactivity: an agent has adaptive behavior; i.e. has the ability to observe its environment and to make decisions when changes occur, to adjust to new circumstances.

- Pro-activity: an agent does not only adjust to the changes occur in its environment, but it takes the initiative to act under specific situations having a complex goal to achieve.

- Computation: the agent reasons based on its learning capacity and its internal knowledge to get closer to the problem overall solving.

Some applications require multiple agents that collaborate to improve the fulfillment of the predetermined goals. This necessitates that the agents have extra communication tools since the cooperation between agents occurs through interaction. Agents require communication capabilities to interact not only with other intelligent agents, but also with their environment, users, and information sources. The creator of the agents can empower them by mobility, which is the ability to migrate in order to perform particular tasks. An agent is called mobile when it is able to travel within the network, migrate from one site to another where it can make direct contacts with other agents and passing information directly. A stationary agent that is compelled to a specific site interacts with its environment through messages to remote objects. Almost all the tasks can be performed remotely by stationary agents. Mobility offers more flexibility, but leads to higher demand on the technical infrastructure and causes major security problems. The most robust areas of agent technology revolves around autonomy and mobility.

In this thesis, we consider cooperative software mobile agents. They collaborate asynchronously to achieve a given goal in a distributed computing environment. Despite the lack of global control, their limited capabilities, the decentralization of data and the asynchronous computation, the agents can achieve plenty of useful tasks. However, their workload orchestration and distribution introduce new levels of complexity.

According to N. Minar et.al [33] "As agents interact with one another - and so have the potential to become LIKE their peers by learning from them-preserving behavioral diversity becomes a major challenge. Efficient division of labor in the absence of centralized control is subtle, important problem". Agents have similar visions of steps to do to accomplish their common goal. They have to interact in order to divide the responsibilities and organize their behavior to avoid the duplication of their efforts. When the deployed agents are anonymous and initially located in symmetric placement on a symmetric anonymous topology, we call this model *symmetric*. In some scenarios, the symmetry of the model and agents do not only affect the efficiency of the agents, but make the problem unsolvable. The behavior diversity breaks the similarities of the agents. In the literature, breaking symmetry is achieved in different ways such as: introduction of randomization, use of asymmetric strategies with a priori agreement on the roles of the agents,

etc. Breaking the symmetry of the model is not only the most important factor to maximize the efficiency of the agents, but also a crucial factor for the solvability of the problem.
"Much of the behavior of agents results from autonomous decisions made in the context of organizational guidelines supported by explicit coordination of short-term activities." [13]

In this thesis, we define some of the extensively studied problems in the mobile agents distributed environments, and we discuss their relationships. The existing models of distributed mobile agents differ from one another on a large number of parameters that mostly deal with the knowledge available to the agents and and their capabilities. These parameters discern the common traits of the distributed environment problems, thus we use them to systematically classify the models used in the literature and to cover all the different possibilities when we intend to solve a particular problem.
The metric information available to the agents (e.g. its size, number of edges, maximum degrees, diameter, grith, etc.) and the topological properties (e.g. the fact that the topology of the network is a regular graphs, a Cayley graph, symmetric graph), differ from one model to an other and empower the deployed agents to solve problems with lower complexities.
When both the topology class and its size are known, the agents have a full map of the network and can exploit this information to solve more complex problems. Therefore, we study the problem of finding the size for ring, mesh, and torus topologies under different assumptions, when it is assumed that the class of the underlying topology is a common a-priori knowledge for all the deployed agents.
We present solutions for finding the size of a ring, a mesh and a torus for several models. We take advantage of the topology characteristics to avoid visiting all the edges and all the nodes. In particular, solving the size of a mesh or a torus by mobile agents that do not have common orientation (i.e. the notion of: south, north, east, and west) is not trivial. We make the agents travelling in a consistent direction without going through a global orientation construction. The complexity of the solutions is $O(\sqrt{n})$ instead of the trivial bound of $O(n)$, where n is the number of nodes in the network.

This thesis is organized as follows: Chapter 2 is dedicated to define the factors and parameters that identify the models of distributed mobile environments; to describe some typical problems such as: leader election, rendezvous, intruder capture, exploration and finding the size of a network; to identify the relationships of these problems and classify their possible solutions. The chapter also includes the recapitulative table of the models studied in the literature. Moreover, chapter 2 enumerates some observations and tools to deterministically break the symmetry of some models: breaking the symmetry of the model is not only an important factor to maximize the efficiency of the agents, but also a crucial factor for the solvability of the problem. Often in the literature, randomization and asymmetric strategies based on a priori agreement on the roles of the agents, are used to break the symmetry.

Chapters 3, 4 and 5 investigate the possible solutions and describe the algorithms to find the size of a network respectively in a ring, mesh, and torus, using one or two agents, under different assumptions on the power of the agents, on the means of communication, and on the availability of common orientation.

Chapter 2

Models and Solutions

There are numerous models for distributed mobile environments, differing from one another on a large number of important factors and parameters. This chapter defines different models for mobile agents, describes some typical problems, and classifies the possible solutions in distributed mobile environments.

2.1 Models

An accurate distributed solution starts with a precise description of the distributed model. However, the area of mobile agents seems to consist of a collection of distinct results in different models without fundamental formal concepts underlying the distributed mobile agents computing. Even so, informal assumptions that seem to be important are made. So as to successfully analyze, classify and compare different approaches and algorithms solving particular problem, we have to state clearly the taxonomy of the assumptions of the models. In this chapter, we classify the models founded on the values of these following constituents: topology, labelling of the edges and the nodes, identities of the agents, a-priori knowledge of the agents, memory space of the agents, communication methods, meeting conditions, synchronous versus asynchronous models, heterogeneous agents, and coordinator agent, etc. Even though some extent of fault tolerance is required in most real distributed systems, in this thesis we study distributed algorithms that are not fault tolerant, considering that other mechanisms will handle failures in case of interruption of the algorithms.

These are the definitions of all the constituents taken into consideration for the description of the models:

- *Topology:* For software mobile agents, the topology is a graph modelled as a set of vertices and edges. Agents are limited to move only on the edges. In a node, edges are assumed to be labelled so to be locally distinguishable. Edges can be unidirectional or bi-directional. Bi-directional edges can be traversed in both directions; while unidirectional edges restrict the agents to move only in one direction. A vertex can host as many agents as the network contains. We assume that the topology does not change dynamically during the computations. The network can be symmetric and look the same for all the vertices e.g. ring, complete graph, complete bipartite graph, etc.; but most of the topologies are inherently asymmetric.

- *Labelling:* In the model, there are three different components that can or cannot be labelled: the edges, the nodes, and the agents themselves.

    1. *Edge Labelling:* The edges are labelled if two (possibly different) labels, known also as ports numbers, are associated to both ends of each edge in the graph. The labels have to be locally distinct (at a given node all the incident edges have distinct labels) in order for an agent to take advantage of the labelling. Edges can be labelled globally in a certain fashion that offer a sense of direction or an orientation for the agents [19, 20]. If there is a sense of direction, that means it is possible to understand, from the labels associated to the edges, whenever different walks from a given node end in the same node or in different nodes. This is the case, for example, of a mesh labelled with {North, South, East, and West}.

    2. *Node Labelling:* Nodes are anonymous if they are not uniquely labelled. Due to protocol and naming convention mismatch, machine faults or even the possible presence of hostile agents, in general, nodes are assumed anonymous. A network is called anonymous when the nodes do not have distinct identifiers, even if the edges have a local labelling.

3. *Agents Identities:* Agents can be anonymous, or labelled by distinct identifiers. In the second case, identifiers can be comparable as integers or incomparable as colors or labels written in different alphabets.

- *Communication Topology Knowledge:* As defined in [19], topology knowledge is part of the structural knowledge that the agent might have of the system and refers to five categories as below:

  1. *Metric Information:* Knowledge of some information about the network; e.g. its size, number of edges, maximum degrees, diameter, grith, etc.
  2. *Topological Properties:* Knowledge of some properties of the topology; e.g. the fact that the topology of the network is a regular graphs, a Cayley graph, symmetric graph, etc.
  3. *Topological Classes:* Knowledge of the "class" to which the topology belongs; for example, the fact that the topology is a ring, a tree, a mesh, a tori, etc.
  4. *Topological Awareness:* Knowledge of the adjacent matrix that describes the topology (i.e. class of the topology and its size).
  5. *Complete Knowledge of the Topology:* Knowledge of the adjacent matrix and its position in it.

- *A-priori knowledge of the agents:*

  1. *Blind Agents:* Agents are dispersed in the network without any a-priori structural knowledge of type "communication topology".
  2. *Informed Agents:* Agents are dispersed in the network having some structural knowledge, usually metric information.

- *Communication Methods:* Mobile agents mainly dispose of four methods to communicate with each other: boards, messages exchange, direct exchange (known also as face-to-face communication) [22], and tokens.

1. *Boards:* The most usual mean of communication among these three methods is the use of boards. In this case; the agents write down the information that they want to share. There are two kinds of boards: blackboards and whiteboards. The blackboard is a unique central common board accessible, in mutual exclusion fashion, by the whole group. Whiteboards are local storage area. Each whiteboard resides in a node of the graph and it is accessible only by the agents that are present at that node.

2. *Messages:* The second mean of communication is sending messages. If the agents are mobile then the network looks like a wireless network model: every agent can communicate with any other agent by sending it a direct message (like using cell phones). Messages have to be minimized so they do not congest the network.

3. *Face-to-Face:* The third communication method is the direct exchange of information which is sometimes called "Free Communication". Meetings are the only information sources; no central control or whiteboards can be used as means of communication. Therefore, agents are mobile and able to meet face-to-face on the node; they can exchange what they learned during their trip only directly when they are at the same node. In this case, communication is instantaneous.

4. *Tokens:* Each agent has a unique marker available, called token, that can be left at a node. The node has enough memory to store the token.

- *Memory:*

1. *Memory of the Agents:* Agents could be oblivious, which means that they have no memory. They could have limited memory as $O(log(n))$, or as little as just one bit $O(1)$. They also could have unbounded memory.

2. *Memory of the Nodes:* Each node has to be able to host all the existing agents and tokens in the model, the whiteboard in case of whiteboards communication model, the labels of the incident edges, and the label of the node in the case of a non-anonymous network.

- *Meeting Conditions:* There are two possible scenarios as meeting conditions for the mobile agents. In the first scenario, two agents can meet only when they both reside in the same node, which means agents become computationally active only at nodes. In the second scenario, two agents can meet either when they are residing in the same node or when they are crossing a given edge.

- *Synchronous versus Asynchronous Models:* In the synchronous model, the agents consume the same amount of time to cross any edge (e.g. one unit of time). In the asynchronous models, it takes for an agent an unpredictable, but finite, amount of time to cross an edge. That time may differs from an agent to an other one or depends on the edge itself. If the computation is synchronous, then the startup time could be simultaneous (i.e. the agents start computing at the same moment) or could be non-simultaneous.

- *Heterogeneous Agents:* The distributed environment contains collaborative agents but it can contain also some neutral agents that are not necessarily directly involved in the computation or contain some harmful agents (sometimes called intruders) known also as enemies. The number of cooperative agents is usually known. The existence or the non-existence of intruders, is part of the assumptions defining the model. The agents that are not directly involved in the computation (let us call then Strangers) might be present because they are involved in other computations and they do have information about the model that collaborative agents ignore. Collaborative agents may interact with strangers to try to get better view of the model. The correctness of the answers of the strangers can be a probabilistic function.

- *Coordinator Agent:* A coordinator agent is about the presence or the absence of a particular agent given special rights to perform particular tasks or coordinate the work of the whole group. If the coordinator agent exists then it is unique, all the other agents are in the same state and the system is called *System with leader.* In this thesis, only decentralized models that contain no coordination software agent are considered. Models and solutions are completely decentralized so in order to get a coordinator agent we have to solve in a

decentralized fashion the "Leader Election" problem.

- *Taxonomy of Solutions:* finally, in all possible variations of the model, the complexity of a solution will be measured using two parameters: time units and number of moves. The algorithms can be randomized or deterministic. The strategies can be symmetric or asymmetric.

    1. *Complexity of the Solutions:* There are two complexity measures for distributed algorithms executed by mobile agents: move and time complexity. The move complexity consists of the total number of moves performed by the agents. The time complexity is the time needed both for the moves of the mobile agents and for computation within the processes. However, computations performed by agents are traditionally ignored, only the time needed for the moves being counted. Tradeoffs between time and moves complexity are often possible.

    2. *Randomized versus Deterministic Solutions:* An algorithm is called deterministic if its behavior does not contain randomized elements (e.g. coins flips). Notice that in distributed models, the asynchronicity aspect of the agents may induce different results for different execution of the deterministic algorithm. An example of a random algorithm in the case of the exploration problem is the following: as agents move along an unknown connected graph to explore or gather, they decide randomly on their own the next node to move to.

    3. *Symmetric versus Asymmetric Strategies:* When identical agents run the same deterministic algorithm, they are acting similarly and the strategy is called symmetric. Otherwise the strategy is called asymmetric; agents play different roles and run different deterministic algorithms. They may have gotten the chance to agree in advance on their roles in a combined strategy or started their computations by electing a leader among them to assign different roles for the agents. As example of using different roles the "Wait for Mummy" technique when two agents want to meet. One player, known as "child", remains stationary while the other agent, known as "mummy", looks for

it. In this case we are using the mummy and child roles to break the symmetry.

2.2 Typical Problems and related works

In the following subsections, we will define the problems of: leader election, rendezvous, exploration, and finding the size of network in the mobile distributed environment.

2.2.1 Leader Election

Leader election is a classical problem in the message passing system [37]. In the mobile agents environment, it is the problem of choosing a single agent among a group of symmetric agents and giving it a particular role to play different than the roles of the other agents. The leader is unique and it does not matter which agent becomes a leader. This task consists of moving the system from an initial configuration where all the agents are in the same state (usually called available), into a final configuration where all the entities are in the same state (traditionally called follower) except one which is in a different state (traditionally called leader). The goal of the elected agent is to act temporarily as a central controller to coordinate the execution of a particular task by the whole agents. Electing a leader in a distributed mobile agents environment is important when some centralized coordination is required in the environment, because there is no known decentralized solution, or because a centralized approach provides a more efficient solution. By solving the leader election problem, the symmetry among the agents in a distributed environment is broken. As we will see also later, breaking the symmetry is at the basis of most control and coordination processes (e.g. mutual exclusion, synchronization, concurrency control, etc.) employed in distributed systems. Also, the leader election problem is closely related to other basic computations (e.g. minimum finding, spanning-tree construction, traversal). If the agents are identified with distinct values, then a possible strategy to solve the problem is to elect the agent having the smallest identifier.

In fully anonymous systems, the conditions for the solvability of the leader election problem among a group of k anonymous asynchronous mobile agents dispersed on an arbitrary anonymous network of size n is examined in [7]. The authors studied the impact of the availability of a

sense of direction and show how o elect a leader when there is sense of direction and $gcd(k, n) = 1$. Unlike all the known leader election protocols that assume the comparability of the agents identifiers, the research conducted in [6] does not rely on the existence of distinct and pairwise comparable identifiers, nor on any topological asymmetry of the network. The leader election problem is studied in its weakest setting: in anonymous highly symmetric graphs (i.e. Cayley graphs) and in qualitative models (e.g. entities are given distinct labels which are however mutually incomparable). The authors determine if the problem is solvable or not and give a conditionally effective protocol that performs election of one agent among any set of agents in any network.

2.2.2 Rendezvous

Rendezvous is a coordinated meeting of distributed mobile agents.

This problem is part of the human beings daily activities; we gather in order to negotiate, to celebrate or to get informed. Through communications by phone, e-mail, fax, and other means, decisions are made on the location and the time of meeting. Since the involved parties are willing to meet, they will eventually make it. In a perfect world, the mobile agents would achieve rendezvous in the same fashion: meeting when they want, in a given address e.g. meet on the node with the lowest identification number as fast as possible. However, in reality the case is not that trivial; faulty nodes, incompatible naming conventions and the refusal to provide the labels must be taken into account.

The solution of the rendezvous problem is a description of a strategy that allows two, or more, mobile agents (also known as trajectory), willing to meet, to meet in an unknown connected graph minimizing the time required to meet. The number of edges traversed can be used as a measurement of the algorithm complexity. As in any computer science problem, the memory required is important variable to take into consideration. The rendezvous itself differs from one model to another. Some authors, as in [31], consider that the rendezvous occurs when the mobile agents either meet on a network node or simultaneously cross the same network link while moving in opposite directions. The traditional search problem is a one side search (i.e. the search is

not mutual: an agent searches for a stationary object or for an another mobile agent that may not be aware about this search, or aware but hiding). The searcher wants to minimize the time it takes it to find a stationary or mobile object. In this case, the only resource measured is the time, in spite of the mobile agents rendezvous problem that considers the used memory. Even though all the involved agents, in the distributed model, are willing to meet still is it not trivial to make it considering the case when agents do not have any way to communicate remotely, do not agree on address for the physical locations, or do not have any orientation. These may seem to be unrealistic scenarios. However, in real life, this situation may occur with a parent and child visiting a crowded place and get separated. Another similar scenario in the military field, consider the case of two soldiers that are parachuted over enemy territory and have to meet quickly in order to accomplish their goals in the minimum time interval. Researchers are interested by the rendezvous to study and the prediction of some natural behaviors to understand and forecast the displacement of some species of birds that gather at certain time of year in order to mate.
In many distributed computing algorithms, authors rely on infinite bandwidth communication between agents at all times in order to achieve promised performance levels. Nonetheless, there are many physical limitations that allow communication only over short distances as underwater agents. Rendezvous problem is formulated in various settings based on different topologies, such as plane (finite set of points), infinite line, ring and mesh etc. Asymmetric rendezvous on the plane is the original setting of the problem. In [4] E. J. Anderson and S. P. Fekete referred to the agents as a player and they considered two scenarios for the rendezvous problem in plane which is the original setting of the problem. The first scenario is defined as follow: two agents have the knowledge that they are distance d apart but they do not know the direction in which they should travel in order to meet. The second scenario is based on an asymmetric knowledge of agents. Only one of the two agents knows the position of the other agent that knows only the initial distance d.
Rendezvous problem is analyzed in [14] considering anonymous ring, trees, and arbitrary connected topologies. Two dispersed mobile agents with distinct identifiers have to meet under two scenarios: simultaneous startup and arbitrary startup. The agents communicate face-to-face and

know the class of the topology, and their identifiers.

Rendezvous in trees relies on efficient network exploration but cannot be used in graphs with cycles. The authors contribute a general feasibility results on arbitrary connected graphs, even with arbitrary startup. They give a generic algorithm for all connected graphs when a simultaneous startup is assumed. In [31], the rendezvous problem is analyzed in the ring topology. Two identical anonymous dispersed agents have to break their symmetry and meet as fast as possible. Authors investigate the use of identical tokens to find a deterministic solution to solve the problem. They drive the explicit conditions under which the problem is solvable. They suggest a tradeoff between the memory of the mobile agents and the time complexity, which is limited.

In [22], the authors consider the rendezvous problem in anonymous ring using $k \geq 2$ dispersed anonymous agents. The described solutions use identical tokens to break the symmetry of the model. The authors derive the lower bounds on the memory required to solve this problem. They discuss the relationship between rendezvous and leader elections problems.

The rendezvous problem is studied in [17] in a anonymous ring of size n, when there is a blackhole (i.e. Blackholes are unsafe nodes that contain a stationary process that destroys any incoming agent without leaving a trace). The research concern is to determine the number of agents that can gather and under what conditions. For k dispersed anonymous asynchronous mobile agents aware of the existence of the danger but not of its location and using whiteboards, the time complexity of the solution is $O(n)$, which is a significant improvement over the $O(n \log n)$ complexity of the existence protocols for the same case.

Also in [21], the authors consider the rendezvous problem of $k \geq 2$ dispersed anonymous agents in anonymous ring using identical tokens and markers to break the symmetry of the models. Running the same deterministic algorithm, the authors study the effects of tokens failure on the required time for the rendezvous with different type of knowledge (knowledge of n, or k, or both).

In [7], the authors investigate the existence of deterministic generic solutions for the rendezvous problem of k anonymous dispersed agents having access to the whiteboards in an arbitrary anonymous network of size n. A generic solution will work regardless the of the network topology and

the initial placement of the agents. The results of this investigation: rendezvous is not possible if $\gcd(n, k) > 1$. On the other hand, the availability of sense of direction is sufficient to solve the problem when $\gcd(n, k) = 1$. The initial placement of the agents creates topological asymmetries that could be exploited to solve the problem.

2.2.3 Exploration

There are two variations of the exploration problem depending on the knowledge and the goals of the agents. The deployed agents might be exploring the graph searching for something or exploring to discover the topology of the network. The performance of their solution is measured by the completion time complexity, by the number of edges traversals or the memory size of the agent.

The exploration problem can be solved using a single agent or by a group of cooperative mobile agents: two or more agents work independently and then exchange the learned information. By exchanging information, an individual agent can acquire knowledge about parts of the network that it has never visited.

Explore to retrieve:

Known also as Navigation problem, the goal of this exploration is to touch all the nodes of the connected graph. It is possible that the deployed mobile agents know the topology of the network before starting but they have to explore it collaboratively to touch all the nodes for object retrieval. Then the output of the process is to return the location of a given information in the graph. The information that the agents may be looking for, could harmful as blackhole and intruders.

Explore to discover:

The goal of this exploration is to overtake the lack of information. the deployed mobile agents ignore the topology of the connected graph, and they have to explore it collaboratively. Then

the output of the process is the construction of a complete map of the environment. The agents ignore the environment but still they can have partial information as the total number of nodes, the order of the graph, the maximum number of the incident edges of node or the existence of a particular node as a central node or a node with a maximum degree, etc.

There are three versions of algorithm of the exploration termination: exploration with return, with stop or perpetual. In the exploration with return the agent has return to its starting position after performing the exploration phase. In the exploration with stop, the agent has to stop when the exploration phase is done. In perpetual exploration, as in [25], a team of agents has to traverse all the edges of the graph but it is not required to stop. That is the existence of a final state is not required.

In [34], the authors have studied the exploration problem in an arbitrary anonymous undirected graph using a single agent without any a-priori knowledge and having an unbounded memory. The agent task is to visit all the nodes and the edges in order to contract a complete map of that graph. The authors consider any move exceeding the actual number of edges in graph as a penalty and they describe a linear algorithm having $O(|V(G)|)$ as a penalty, where V in the number of nodes in the explored graph. The authors explained and compared the efficiency of three algorithms: greedy, DFS, and linear. The exploration techniques depend on the strategies used by agents to choose the next node to visit and the on involved information factored to make that decision. These solutions differ in which edge to use to leave an explored node (e.g. all the incident edges of the current node are already visited). Commonly, these solutions have three phases. The first one is a common phase in which a mobile agent uses unexplored edges as long as possible. The second phase imposes more tactics; it starts when there are no more unexplored edges so that a node is called saturated and the agent is considered stuck. The third phase is the most critical phase, reflects the strategy of picking the following node to move to in order to continue the exploration. In the greedy solution, the agent chooses the local best, while in the DFS, the agent moves to the most recently visited free node (e.g. still some unvisited incident edges of the node) following the shortest path in the explored part of the graph. In the linear algorithm, the agent constructs, dynamically, a spanning tree and will not get distracted by free

visited nodes that are close enough the current position of the agent and far away from the still unexplored part of the graph as in the greedy solution; neither will it be attracted by far away free node as in DFS algorithm that takes into account only the partial information of the order of the visited nodes.

The linear algorithm maximizes the information used by the agent to make the movement decision by keeping track of the bridges and dynamic tree and is therefore the most efficient of the three solutions presented above.

Also in [33], the exploration problem is refereed as the mapping task and it is studied in the same kind of graph as in [34], but using k blind anonymous dispersed agents instead of a single one. These agents have an unbounded memory and can communicate only face-to-face. The authors investigate the effectiveness of agents cooperation and the techniques to achieve an efficient labor division. They applied three different types of agent algorithms governing the movements of the agents: random, conscientious, and super conscientious agents. These algorithms differ in the way that the agents choose the next node to move to. In the first algorithm, the agents move simply to a random adjacent nodes, while in the second one, similarly to a depth-first search of the network, they move to the adjacent never visited node or to the least recently visited node using only their knowledge that they learned by themselves. In the last algorithm, the agents move to nodes that have not been explored and in contrast with the second algorithm, they can use all their knowledge, including the data learned from their peers.

Also in [25] the exploration problem is studied, but with a different perspective. Considering the same type of topology: arbitrary anonymous undirected graph, but with k blind mobile agents modelled as k finite automata (HYDRA). Due to this configuration, the mobile agents are globally cooperative, they can exchange all information available to them at each step of the exploration as if there as using a blackboard. The authors compare this model with the local model (i.e. face-to-face model), and prove that the global one is much stronger. In the global model, a universal exploration team, is composed of at least four agents.

Particulary, the exploration of unknown anonymous trees is analyzed in [27]. The authors deployed k collocated distinctly identified agents having an unbounded memory. They considered

three communication scenarios: whiteboards, blackboard, or no mean of communication. The complexity of the solutions is influenced by the possible communication between the agents. While in [15], the problem of anonymous tree exploration using a single blind agent is considered. Using a single agent, the paper [23] analyzed the problem of exploration of a directed anonymous digraphs with n nodes. The agent knows the class of the graph and its size. It has a constant memory and uses whiteboards. The size of it memory and the whiteboards depend on on the maximum out-degree of the nodes.

In [36], the authors use the rendezvous as preliminary phase of the exploration to overtake the impossibility of communication between agents. So they can merge the partial maps that they generated and collaborate to accelerate the exploration phase. The authors propose two kinds of solutions: deterministic and probabilistic. These solve the gathering problem by visiting a predefined number of points. Results show that the distributed exploration is faster than the single agent system even with the constraint of performing rendezvous to allow communication and the fact that the maps merging is a problem in itself because agents must share a common reference point to be able to merge their maps.

In [36] a deterministic algorithm is proposed based on sequential visits of the landmark set (most particular nodes, i.e. the nodes with the smallest degree, the central nodes, the corners, etc.): one agent picks a node from that set, waits there for the other agent, who is visiting all the nodes on the list. If the rendezvous does not occur after a given time, predicted for n cycles, the first agent moves to the next node/landmark. Where N is the size of the landmark set. This solution is similar to "Wait for Mommy" where a variation of sequential algorithm is proposed and is called Smart-Sequential. Agents, each on its side, assign a "goodness" value to each pair wise combination of landmarks. The list of the landmarks of each agent is sorted relatively similar, but not identical, depending on the given rankings.

2.2.4 Finding the size of a network

The problem of finding the size of a network consists of counting the number of its nodes n. We assume that the agents have the knowledge of the class to which the topology belongs (i.e. agents know that they are in a ring, a mesh or a tori) and their task is to determine the number of nodes of the network. The problem is solved when all the cooperating agents know the number of the nodes without having to go back to their home bases. Another variant of problem when the agents have to go back to their respective home bases in order to terminate. This problem has not been previously studied by itself, it was always subproblem of map construction (i.e the problem of drawing a map to show the exact shape of a given graph).

2.2.5 Intruder Capture

The intruder capture problem considers a team of mobile software agents deployed to capture a (possibly hostile) intruder in a network (i.e. virus). All agents, including the intruder move along the network links; the intruder could be arbitrarily fast, and aware of the positions of all the agents. The solution of thus problem is to design moving strategy for the agents in order to capture the intruder that can move arbitrarily fast in the network. To surround the intruder, the agents will be scanning all the nodes to clean the network. In [5], the authors determine the smallest number of searchers required to "clear" a tree.

2.3 Relationships of the Problems

The above defined problems emerge into two distinct classes of distributed problems: contention and cooperation.

2.3.1 Leader Election and Rendezvous

The problems of leader election and rendezvous are equivalent in most models. If the leader election problem is solved then the solution for rendezvous problem is easily derivable and verse-versa. If the leader election problem is solved, then the leader becomes stationary while the other agents explore the graph to reach its node as a meeting point. Equivalently, if we solve

the rendezvous problem, then all the agents reside in the same node and they can compare their identifiers locally (based of face to face communication only which we assume always possible) and elect a leader; If the agents are anonymous they still can elect a leader based on the first access to the shared memory space (by writing on the whiteboard, the blackboard, or releasing the token). These problems are not equivalent is when the agents are anonymous and communicate only face to face. In this case, in fact there are some situations when they still can meet but they cannot elect a leader. In [22], the author investigate the use of identical tokens to solve the leader election and the rendezvous problems. It turns out that the relationship between these two problems depends on whether the network has sense of direction. Thus they are equivalent if there is a sense of direction in the topology; however the leader election problem is strictly more complex than the rendezvous problem.

2.3.2 Exploration and Finding the Size of a Topology

These two problems are similar but not always equivalent. In order to explore a graph, the agents have, all the time, to touch all its nodes and edges but if they need only to count the number of nodes, they do not have to visit all the nodes; they can use the characteristics of the topology to reduce the number of nodes to visit. Example of useful characteristics of the topology: in a mesh, the agents have to count only the nodes on the width and the length and multiply these two values to compute the size of a mesh. In a hypercube, the agents need to know the degree of a node and the compute the size of a network. While in a ring or in an arbitrary topology, regardless the goal of the computations, exploration or finding the size of the topology, the agents have to visit all the nodes and the edges of the network.

2.3.3 Rendezvous and Exploration

These problems have different aspects. Rendezvous is an atomic goal (gather), unlike the exploration problem involves repeated tasks (explore, draw partial map, exchange maps and merge maps, etc.).

In the rendezvous problem, the involved dispersed agents in a distributed network collaboratively

make efforts to meet together on the same site. While in the exploration problem, they collaboratively construct a map of an unknown connected network by drawing partial ones and merging them as long as the map is not complete. Solvability of these problems is not guaranteed; it depends on the models and the scenarios. Depending on the assumptions, these two problems can be equivalent. Sometimes in order to gather, mobile agents have to independently explore the unknown connected graph and consequently the rendezvous problem is reduced to the graph exploration problem. In other situations, exploration is not a preliminary exercise to gather. On the contrary, the exploration problem gains substantial speed by having the rendezvous as a key step for the collaborative exploration.

2.4 Breaking the Symmetry of the models and Behavioral Diversity

In this section we make some observations about tools and techniques that could be employed to break the symmetry in the system of mobile agents.

2.4.1 Symmetry

Symmetry arises when anonymous agents are initially located in symmetric placement on a symmetric anonymous topology. Then the agents can be placed equidistantly when $gcd(k, n) > 1$. As proved in [7], it is not sufficient to have $gcd(k, n) = 1$ (e.g. k number of agents and n is the number of nodes in the network) to solve the symmetry breaking problem; it is sufficient, however when combined with a sense of direction.

In other words, an algorithm has been designed for breaking the symmetry among a group of k anonymous asynchronous mobile agents dispersed on an arbitrary anonymous network of size n, when *sense of direction* is available and $gcd(k, n) = 1$.

Therefore, the relation between the number of the nodes n in the anonymous symmetric topology and the number k of the deployed identical mobile agents impacts the symmetry breaking solvability.

This is the case, for example, of agents placed equidistantly on a ring. In such situations, most problems are proven to be unsolvable since any synchronous execution of a solution would lead the agents to perform exactly the same actions without being able to break the symmetry. Most of the problems of the mobile agents (e.g. rendezvous, leader election) require the agents to break the symmetry so to react in different fashion. As in [31, 17], under the above assumptions, gathering problem of two mobile agents in a ring is unsolvable. Agents are not able to meet since they are going in same direction at same speed and at same time and consequently they always preserve the initial distance between them. Even when there is a sense of direction known to the agents, there is no generic protocol that always terminates in finite time that breaks the symmetry of the agents and makes them move in opposite directions.

The symmetry breaking can be achieved by solving the leader election problem, introducing a coordinator software agent, called also "facilitator", that represents the layer of diversity in the model. Therefore, the symmetry breaking problem is very similar to the leader election problem. The coordinator might play a very important role in order to solve the rendezvous problem or to perform the exploration and can be used to handle the work distribution of the agents.

Randomization is another typical solution to break the symmetry. To avoid random solutions, asymmetric strategies can be used. Therefore there is a need to create distinguish roles to be able to distribute the strategies, which is not evident when the agents are identical and nodes are similar. In some models, it is possible that the agents meet before the algorithm begins to decide which role each one will take beforehand, then they can use different deterministic strategies. The possibility of gathering to negotiate beforehand, make plan and share the roles to be able to use asymmetric strategies is a strong assumption and it is not always possible. Rather than using randomized algorithms or different deterministic algorithms for various mobile agents, some papers propose the utilization of identical tokens [22, 21] to break the symmetry under certain conditions. Other authors [14, 39] assume that the mobile agents have different identifiers and, based on their labels, can break the symmetry by constructing different input for the executed algorithm. In [33], agents react based on different set of data and then make different decisions and end-up in different nodes. The limitation of the allowed amount of communication between

the collaborative agents and its frequency is used to break the symmetry and it introduces behavioral diversity that ensures a better distribution of the workload between agents. Thereby, there is a link between the symmetry breaking that is necessary to solve some distributed problems and the behavioral diversity that maximizes the collaboration between the agents. In the next sections, we present ideas and examples for symmetry breaking by solving the problem of leader election in the model defined as follow:

- The network is anonymous.
- Edges are locally labelled.
- Agents know the class to which the topology belongs.
- The number of agents k is known by all the agents.
- Each agent has at least $O(logn)$ bits memory.
- All the agents follow the same deterministic algorithm.
- The model is asynchronous.
- Agents start from different home bases.
- Agents can be anonymous or having distinct identifiers.
- Agents communicate using whiteboards.

2.4.2 Tools for Symmetry Breaking

△ **Asymmetry in Topology:** Topologies can refer to a general topology or be specified as ring, mesh, torus, or hypercube, etc. Some topologies are more symmetric than others. For example, a ring topology is completely symmetric while an arbitrary graph is inherently more asymmetric than symmetric. For asymmetric topologies, it is trivial to determine a unique node known to every agents, such as the node with a unique degree, the unique articulation node, etc.

Example of electing a leader in the star: Assuming that the network is a star, then the first agent reaching the central node (the node with the highest degree) and succeeds written its identifiers on the whiteboard is elect as a leader.

△ **Asymmetry of the initial placements of the agents:** Even in symmetric topologies as the ring, if the distances between the initial placements of the agents in the network, called home bases, are not equidistant then it creates topological asymmetries that can be exploited to break the symmetry.

Example of electing a leader in the ring: In this example, we assume that there are two anonymous agents and the size of the ring n is odd. The initial placements of the two deployed agents are asymmetric since the distance d that separates them is smaller than the half of the total number of nodes of the ring (i.e. $d < n/2$). When the agents wake-up, mark down their home bases by writing 1 on their respective whiteboards. The agents count all the nodes between the home bases. Then they go to the unique point, which is the middle of the interval containing an odd number of nodes. The first agent that reaches the unique node and succeeds to mark 1 on the whiteboard becomes the leader.

△ **Asymmetry of the identities of the mobile agents:** If the agents have different identifiers, then we can break their symmetry based on that fact.

Example of electing a leader among agents having distinct identifiers: The agents write down their identifiers on their respective whiteboards. All the agents navigate in the network, comparing the values on the whiteboards and choose the agent with the smallest identifier as the leader. In [31], the authors use the identifiers for a more sophisticated procedure is called extend-labels to break the symmetry in a synchronous environment.

△ **Asymmetry of the past success stories of the agents:** In the beginning all the agents are identical. As they navigate in the network, they learn new information and can act differently based on their different experiences while progressing towards the global goal.

Example of electing a leader among agents based on their past success stories: Assuming that two identical agents are visiting and counting the nodes in the ring with an

odd size n. The agent that has visited the greater number of the unexplored nodes that it is visiting and marking their respective whiteboards for the first time becomes the leader.

△ **Breaking the symmetry using a blackboard:** The problem of breaking the symmetry becomes trivial when a blackboard is available regardless if the agents reside in the same home base or in different ones.

Example of electing a leader using the blackboard: When a blackboard is handy, the first agent succeeds to access the mutual exclusive blackboard becomes the leader and the problem is solved.

2.5 Studied Models:

This following list of abbreviations is used for the understanding the table classifying the models reviewed in the literature:

Abbreviation	Meaning
Pb	Problem
Sol	Solution
RDV	Rendezvous Problem
Exp	Exploration Problem
BH	Blackhole Location Problem
LE	Leader Election Problem
Synch.	Synchronous network
Asynch.	Asynchronous network
Arbit.	Arbitrary
Anony.	Anonymous
Direc.	Directed
Undirec.	Undirected
WB	Whiteboards
BB	Blackboard
FtF	Face-to-Face
SD	Sense of Direction
Rand.	Random Solution
Det.	Detrministic Solution
Sym.	Symmetric Solution
Asy.	Asymmetric Solution
D.	graph's diameter
AAU	Arbitrary Anonymous Undirected Graph

2.6 Our Setting and Results

In chapters 3, 4, and 5, we consider the problem of finding the size of a ring, a mesh and a torus under different assumptions. Some basic assumptions are common to all the models (enumerated below); while other assumptions differ from one model to another as the power of the agents, the availability of common orientation (left and right directions in the ring; north, south, west and east in the mesh and the torus), and the way agents communicate. The problem is solved when all the cooperating agents know the number of the nodes. We assume that the agents do not have to go back to their home bases.

The different models are formulated by the possible combinations of the different parameters (the agents start from the same home base or from different ones (dispersed), the identities of the agents: distinct or anonymous, the available method of communication: whiteboards, blackboards, face-to-face or tokens, the knowledge of common orientation). By combining all the possible variables of the four listed parameters, we have 16 different models. For some models we use the same solution algorithm. This means that, even though the models are different, their respective solutions are equivalent. Avoiding the redundancy, we grouped the equivalent solutions and then we have reduced the number of the models. In all the models considered below, these are the common assumptions:

- All the deployed agents have the same processing capabilities, and move in the graph from node to neighboring node.
- All the deployed agents know the class to which the topology belongs but they ignore the number of nodes.
- The network is anonymous.
- Edges are locally labelled; otherwise an agent can get stuck infinitely navigating back and forth on same edge.
- The number of agents k is known by all the agents.

- Each agent has at least *logn* bits memory.
- All the deployed agents follow the same algorithm.
- All the computations, including the moves, are asynchronous and if several agents are deployed then the startup times may be different with regard to the worst case.
- Computations are assumed synchronous only when computing the ideal time complexity.
- We consider two measures of complexity: the number of moves made by the cooperating agents and the time consumed by the slowest agent.
- Different communication methods are used depending on the model, agents could be having, blackboards, whiteboards, tokens or only face-to-face communication.
- If tokens are available, then they are identical.
- Regardless the available communication method, agents are always able to communicate face-to-face when they reside in the same node.
- The different parameters under consideration are:

    1. Agents reside in the same home based or dispersed in the network.
    2. The identities of the agents: distinctly labelled or anonymous.
    3. The communication methods available for the agents: whiteboards, blackboard, face-to-face, or tokens.
    4. The availability of common orientation for all the agents.

For the clarity of the models definition, their respective parameters will be recapitulated in tables. Little stars will superscript the parameter that is not affecting the correctness of the proposed algorithm, if there is any. Such as, if an algorithm is used to solve the problem in both cases of the identities of the agents (anonymous and distinctly identified), then the weakest model is captured in the table, which is the anonymous agents in this example, and

a little start superscript the anonymity indicating that if the agents are distinctly identified then the identifiers will be simply not usefully.

We present deterministic solutions for the finding the network size in a ring, mesh and torus in different models. We compare the solutions for the different models and we draw some conclusions on the importance of common orientation in the model, the availability of blackboard to communicate, the nature of the topologies, and the symmetry breaking possibilities.

Ref.	Pb	Topology	# Agents	Agent's Identity	Agent's Memory	Communication	Agent's Knowledge	Time-Ass.
[34]	Exp	AAU	Single	N.A.	Unbounded	N.A.	Blind	N.A.
[14]	RDV	Anony. Ring, Tree, Arbit.	2 Dispersed	Distinct	N.A.	FtF	Class:Ring, Tree, Arbit. Possibly Id and n	Both
[33]	Exp	Arbit. Undirec.	k Dispersed	Anony.	Unbounded	FtF	Blind	Synch.
[31]	RDV	Ring	2 Dispersed	Anony.	Depends on knowledge	Tokens	Possibly:N,d and SD	Synch.
[17]	RDV	Anony. Ring	k Dispersed	Anony.	N.A.	WB	Existence of BH and SD	Asynch.
[22]	RDV	Anony. Ring	k Dispersed	Anony.	Depends on knowledge	Tokens	Possibly SD	Synch.
[6]	LE	Cayley	k dispersed	In-Comp.	WB	None		
[7]	LE + RDV	Anony. Undirec.	k Dispersed	Anony.	N.A.	WB	$\gcd(n,k) = 1$ and Possibly SD	Asynch.
[25]	Exp	AAU	k	Distinct	k states (HYDRA)	WB, BB, FF	Existence of BH, possibly SD	N.A.
[16]	BH	Anony. Ring	k	Anony.	N.A.	WB $O(\log n)$	Possibly n, SD	Asynch.
[21]	RDV	Anony. Ring	k Dispersed	Anony.	O(k $\log n$)	Tokens	Possibly n,k	Synch.
[18]	BH	Anony.	k co-located	Distinct	O($\log n$)	WB	Full	Asynch.
[27]	Exp	Anony. Tree	k collocated	Distinct	Unbounded	WB, BB, none	IDs	Synch.
[23]	Exp	Direc Anony. Digraph	Single	N.A.	Constant	WB $O(\log d)$	Graph and its size	N.A.
[15]	Exp	Anony. Tree	Single	Anony.	$O(\log max(d))$	None	Blind	N.A
[39]	Exp	Anony. Graph	Single	Anony.	$O(D \log d)$	None	Blind	N.A

Chapter 3

Finding the size of a Ring

In order to find the size n of a ring, the agents have to visit, cooperatively, all the nodes of the network. Then finding n in the ring is equivalent to the navigation problem. Since agents know the topology class, which is the ring in this chapter, the problem is also equivalent to the exploration problem.

In the ring topology, when the size of of the ring is known and the edges are consistently labelled with left an right, then the agent has a sense of direction. However, only the edges are consistently labelled with left an right, but the number of nodes is unknown, then the agent does not have sense of direction since it cannot tell for example whenever it gets back to its home base or not; in this case the agent has an orientation. Notice that the sense of direction is more powerful compare to the orientation. For the ring topology, we will present deterministic algorithms to solve the problem of finding the size of the network using one agent and two cooperative agents in all the different models.

3.1 Algorithm for one agent in the blackboard model:

With *one agent*, no cooperation is possible; and the following algorithm is used to find the number of nodes in a ring in all our variations of the model, except when the agent is not able to mark and recognize its own home base which is the case of *the model with only the blackboard*, using only the blackboard, the agent is not able to mark its home base and recognize it, because the

nodes are anonymous. There is no way to make a link between a particular node and a global memory space: blackboard.

1. When the agent wakes-up, it marks its home base (using its token or the whiteboard).
2. The agent starts walking in an arbitrary direction counting the nodes until it arrives back to its home base.
3. When the agent gets back to its home base, it recognisees it, finishes the counting of the nodes and terminates.

Theorem 1. *The agent performs n moves and consumes n time units in order to find the size of a ring.*

PROOF The agent visits all the nodes in order to find the size of a ring, then n moves are required. For the ideal time, one time unit is needed for each move, therefore to make n moves, n unit time are needed. ∎

Global Results	Worst Case	Best Case
Number of moves	n	n
Time	n	n

Table 3.1: Finding the size of a ring with one agent.

In all the following algorithms, 2 *agents* are deployed to find the size of a ring.

3.2 Algorithms for two agents

3.2.1 Model 1: Starting from the same home base

In model 1, we consider the following assumptions:

1. Agents start from the same home base.
2. The agents can be anonymous if they are using whiteboards or blackboard.

3. The agents are distinctly identified if they are using tokens or face-to-face only.

4. Regardless if the agents do have common orientation or do not.

Parameters	Ring - Model 1.1	Ring - Model 1.2
Communication Methods	Tokens or Face-to-face	Blackboard or Whiteboards
Identifiers	Distinct	Anonymous*
Orientation	No*	No*

Table 3.2: Parameters of model 1 - Ring.

We remind that a star indicates the weakest choice for this parameters. For example, anonymous identifiers indicates that our solutions work regardless the availability of distinct identifiers.
In this model, the presence of common orientation will not make any difference, since both agents reside in the same home base and can go in opposite directions without specifying which agent has to go in which direction. Thus, algorithm 1 is used regardless the presence of common orientation. When the whiteboards or the blackboard are available then the availability of distinct identifiers does not help since agents can break their symmetry based of the first access of the boards, while using tokens or face-to-face communication method, the agents have to be distinctly labelled so they can exchange their labels and elect an agent which has the smallest identifier to be able to go in opposite directions to be able to cooperate. When the agents are anonymous and have only face-to-face communication, they will not be able to break their symmetry since the identifiers are the only information that they can exchange and based on it they can go in opposite direction. In this case, the agents cannot mark their home base or break their symmetry, so they will not be able to count the number of nodes and terminate.

Algorithm 1:

1. When an agent wakes-up, it makes sure locally that the other agent is awake or wakes-it up.

2. The agents break their symmetry by electing a leader if they are distinctly identified or by accessing the whiteboard or the blackboard, then they start walking in opposite directions

counting the nodes until they meet again and then terminate. Notice that the meeting can occur on a node or an edge, if it occurs on an edge than an extra move is performed.

Theorem 2. *In order to find the size of a ring, the agents perform n moves and consumes $n/2$ time units in the best case, while they perform $n+1$ moves and consumes $n/2+1$ time units in the worst case.*

PROOF The agents cooperatively traverse all the ring, then n moves are required. Since we are considering ideal time and the agents start simultaneously, they will meet after $n/2+1$ time units. In the worst case, the meeting occurs on an edge and causes an extra moves and time unit. ∎

Global Results	Worst Case	Best Case
Number of moves	$n+1$	n
Time	$n/2+1$	$n/2$

Table 3.3: Finding the size of a Ring with 2 agents starting from the same home base.

Remark: When anonymous agents have only face-to-face communication, the problem of counting the number of nodes in a ring is deterministically unsolvable. Since if the agent are initially in a symmetric configuration then they will not be able to break their symmetry to go in opposite directions in order to be able to figure out when they have to terminate the counting. For all the remaining models, we consider that *the agents are starting from different home bases.*

3.2.2 Model 2: Blackboard

1. Regardless of the identities of the agents (distinct or anonymous).
2. Communication method: blackboards.
3. Regardless if the agents do have common orientation or do not.

In the case of communication trough blackboard, the availability of distinct identifiers as well as the presence of common orientation could not be exploited. Distinct identifiers are not needed since the symmetry is broken based on the first access to the blackboard. Common orientation

Parameters	Ring - Model 2
Communication Methods	Blackboard
Identifiers	Anonymous*
Orientation	No*

Table 3.4: Parameters of model 2 - Ring.

will not make any difference because there is no need to specify which agent has to go in which direction. Thus, algorithm 2 is used regardless the identities of the agents and the availability of common orientation.

Algorithm 2:

1. An agent tries to access the blackboard, if it is the first that succeeds to write 1 on the blackboard, then becomes a leader. Otherwise, it starts walking in any arbitrary direction
2. The leader waits for the defeated agent.
3. When the agents meet they have to start walking in opposite direction. We already broke the symmetry of the agents based on the access of the blackboard. Then the elected agent picks a direction and the defeated agent follows the opposite direction. Agents will be going in opposite directions counting the nodes on the network.
4. When the agents meet for the second time, they sum the total of nodes and terminate.

Theorem 3. *In order to find the size of a ring, in the best case, the agents perform* $n+1$ *moves and consume* $\frac{n}{2}+1$ *time units, while in the worst case, they perform* $2n-1$ *moves and consume* $\frac{3n}{2}-1$ *time units.*

PROOF With only blackboard, agents are not able to mark their respective home bases and recognize it since the nodes are anonymous. There is no way to make a link between a particular node and a global memory space as the blackboard, then rendezvous is a key step to find the size of the ring. In the best case, the agents perform only one move to meet; while in the worst case, they perform $n-1$ moves. After meeting as in the model 1, they have to visit all the nodes in order to find the size of a ring, then n moves are required in both best and worst cases. That makes a total of $n+1$ moves in the best case and $2n-1$ moves in the worst case. In terms

of time units, the meeting phase requires as many time units as moves, since only one agent is moving while the second one is waiting, for the second phase, both the agents are computing at the same time, then together, they make 2 moves at every time unit which make $n/2$ unit time to perform n moves. In total, $n/2+1$ time units in the best case, and $(n-1)+(n)/2 = 3n/2-1$ time units in the best case. ∎

Global Results	Worst Case	Best Case
Number of moves	$2n-1$	$n+1$
Time	$3n/2-1$	$n/2+1$

Table 3.5: Finding the size of a Ring with 2 agents starting from different home bases, using blackboard.

3.2.3 Model 3: Whiteboards

1. Regardless of the identities of the agents (distinct or anonymous).
2. Agents communicate via whiteboards.
3. Regardless if the agents do have common orientation or do not.

Parameters	Ring - Model 3
Communication Methods	Whiteboards
Identifiers	Anonymous*
Orientation	No*

Table 3.6: Parameters of model 3 - Ring.

In this model, the availability of distinct identifiers as well as the presence of common orientation could not be exploited. Distinct identifiers are not needed since the agents are dispersed in different home bases. Common orientation will not make any difference because there is no need to specify which agent has to go in which direction. Thus, algorithm 3 is used regardless the identities of the agents identifiers and the knowledge of common orientation

Algorithm 3:

1. When an agent wakes-up, it starts walking in an arbitrary direction; counts each node and marks the partial sum on the corresponding whiteboard (the agent marks 1 on the whiteboard of its home base, marks 2 on the following whiteboard of the node and so on).

2. While the agent is walking and counting the nodes, two possible cases can occur:

   - Case 1: In the beginning, both agents took opposite directions and they will eventually meet on a node or an edge. The meeting for the first time carries the information that the nodes in the first interval are counted and still there is a second interval with uncounted nodes. The fact that the agents meet on a node or an edge does not affect the solution, the agents that met on the edge, will continue their moves, get to two different nodes, check the values written on the whiteboards, compare the values of the counted nodes so far, follow the path of the minimum value, if the values are identical then agents can follow an arbitrary direction (could be the same or opposite). Since the agents have to go back to check how many nodes are left uncounted between the two home bases in the second interval. Preferability, the agents go back to the second interval by reaching the home base of the agent that counted less nodes, that means using the shortest path to go count the nodes between their home bases in the other side and terminate.

   - Case 2: In the beginning, the agents took the same direction. Then they will not meet before passing by the home base that may contain an agent that is still asleep or has been already counted by the other agent (that left to count the nodes) and marked by 1. This means that the agent that arrived to the home base of the other agent, has already counted all the nodes in one side of the ring, (To differentiate the agents and facilitate the description of the following step, let us call this last agent "agent 1", and the other agent "agent 2"). Then, agent 1 reverses its direction and goes to help agent 2 counting the nodes in the other side of the ring. Agent 1 keeps walking until it gets to the node just after its home base, checks if its corresponding whiteboard is unmarked (the node is uncounted). It counts it and marks the total of the nodes that

counted on the whiteboards as long as it is advancing. The agent 1 keeps walking and counting the nodes, writing down the partial total on whiteboards until it encounters the first counted node by the other agent. Each agent sums the partial total written on the whiteboard with the number of nodes that it counted and terminates.

Theorem 4. *To find the size of a ring , in the best case, the agents perform $n+1$ moves and consume n time units, while in the worst case, they perform $2n+1$ moves and consume $n+1$ time units.*

PROOF Since the two agents do not reside in the same home base, then they are separated by the distance d_1 on one side of the ring and by the distance $d_2 = n - d_1$ on the other side. In case 1, the agents start walking in opposite directions then they will meet on a node or an edge in the interval with d_1 nodes. Let d_{11} be the distance traversed an agent and d_{12} be the distance traversed by the other agent, then $d_1 = d_{11} + d_{12}$ (w.l.g. let us assume that $d_{11} \leq d_{22}$). The first agent makes d_{11} moves and the second agent makes d_{12} moves. If the rendezvous happened on an edge, then an extra move is needed. When the agents meet, they continue walking and counting the nodes together. They will make $2(min(d_{11}, d_{22}) + d_2)$ extra moves (min $(d_{11}, d_{22}) + d_2$ moves by each) to count all the nodes and agents terminate. The total number of moves is $d_1 + 2d_{11} + 2d_2 = n + d2 + 2d_{11}$. The best case or the worst case depends if $d_2 + 2d_{11}$ is at its minimum or its maximum. In the best case $d_2 = 1$ and $d_{11} = 0$, then the number of moves is equal to $n+1$. In the worst case $d_2 = n-1$ and $d_{11} = 0$, and the rendezvous occurs on an edge, then the number of moves is equal to $n + (n-1) + 2 = 2n + 1$. In terms of time complexity, n time units are required in the best case and $n+1$ time units in the worst case. For case 2, the first agent makes $2d_1$ moves to get to the home base of the second agent and to go back to its own home base. The agents may meet somewhere in the d_2 interval; so both of them made an extra d_2 moves to finish the counting and terminate. Or the second agent makes $2d_2$ to reach the home base of the first agent and come back to its own with meeting the agent itself. The total number of moves is $2d_1 + d_2 = n + d_1$ in the first case or $2d_1 + 2d_2 = 2n$ in the second case. The best case occurs in the first situation when $d_1 = 1$ for a total of $n+1$ moves. The worst case occurs in the second situation, then the total is $2n$ moves. In terms of time complexity, $n+1$ time units are required in the best case and n time units in the worst case. For the overall best

case, $n+1$ moves and n time unit are sufficients.While for the overall worst case, there is a need to $2n+1$ moves and $n+1$ time units. ∎

Remark: Every whiteboard will be used once to write the partial sum.

Case 1	Worst Case	Best Case
Number of moves	$2n+1$	$n+1$
Time	$n+1$	n

Table 3.7: Finding the size of a Ring for case 1 with 2 agents starting from different home bases, using whiteboards.

Case 2	Worst Case	Best Case
Number of moves	$2n$	$n+1$
Time	$n+1$	n

Table 3.8: Finding the size of a Ring for case 2 with 2 agents starting from different home bases, using whiteboards.

Global Results	Worst Case	Best Case
Number of moves	$2n+1$	$n+1$
Time	$n+1$	n

Table 3.9: Finding the size of a Ring with 2 agents starting from different home bases, using whiteboards.

3.2.4 Model 4: Tokens

1. Regardless of the identities of the agents (distinct or anonymous).

2. Communication method: tokens.

3. Regardless if the agents do have common orientation or do not.

In this model, there is no need for distinct identifiers since the agents do not have to break the symmetry to meet. The availability of common orientation could not be exploited too. It

Parameters	Ring - Model 4
Communication Methods	Tokens
Identifiers	Anonymous*
Orientation	No*

Table 3.10: Parameters of model 4 - Ring.

does not make any difference because there is no need to specify which agent has to go in which direction. Thus, algorithm 4 is used regardless the used identifiers and the knowledge of common orientation.

Algorithm 4:

1. When an agent wakes-up, it releases its token in its home base and starts walking in an arbitrary direction while counting the nodes.
2. Whenever, an agent passes by an agent that it is still asleep, then it wakes-it up.
3. When the agent passes by a token for the first time, it ignores it.
4. when the agent passes by a token for the second time, it means it is back to its home base, it counted all the node and terminate.

Theorem 5. *In order to find the size of a ring, the best and the worst cases are the same, the agents perform $2n$ moves, n moves by each agent, and consume n time units.*

PROOF In the algorithm 4, the agents did not cooperate to count the number of nodes in the ring. Each of them count n by itself on his side. Therefore, each agent makes n moves. There is 2 agents in the model then $2n$ moves are made to solve the problem. Assuming that the agents are computing at the same time, then they consume only n time units. ∎

Global Results	Worst Case	Best Case
Number of moves	$2n$	$2n$
Time	n	n

Table 3.11: Finding the size of a Ring with 2 agents, starting from different home bases, using tokens.

3.3 Finding the size of a ring in presence of an intruder

In the ring topology, the problem of counting the number of nodes is equivalent to the navigation problem. All the nodes have to be visited. There are two possible ways to make two agents visiting all the nodes; in the first strategy, the agents start counting the number of nodes separately as soon as they wake up from different home bases, while in the second strategy, the agents sweep the network starting from the same home base and going in opposite directions, or by first performing a rendezvous if they are not starting from the same home base. The second strategy does not only count the number of nodes in the ring but it also allows the agents to capture the intruder (this is also the case for the solutions described for Model 1 and Model 2). Then there is a link between the problem of finding the size of the network and the intruder capture problem. In both cases, agents need to visit all the nodes. However, these two problems are not equivalent since in order to catch an intruder we have to explore but not every exploration strategy will allow us to catch the intruder, as it is the solutions for the Model 3 and the model 4. Using two agents, we have given already the solutions for the homogenous systems and we now study the situation when an intruder is present in the network (Model 5 and Model 6).
When an intruder is present, if the agents do not have the same home base, then the first step of the algorithm is a rendezvous before proceeding to count the number of nodes.

3.3.1 Model 5: Whiteboards and distinct identifiers

1. The agents have distinct identifiers.
2. Agents communicate via whiteboards.
3. Regardless if the agents do have common orientation or do not.

In this model, the availability of common orientation could not be exploited while distinct identifiers are needed. In order to capture the intruder, if it exists, distinct identifiers are used to break the symmetry and make the agents meet. Common orientation will not make any difference because there is no need to specify which agent has to go in which direction. Thus, algorithm 5 is

Parameters	Ring - Model 5
Communication Methods	Whiteboards
Identifiers	Distinct Identifiers
Orientation	No*

Table 3.12: Parameters of model 5 - Ring.

used regardless the knowledge of common orientation.

Algorithm 5:

1. When an agent wakes-up, it marks down its identifier on the whiteboard of its home base and it starts walking in an arbitrary direction. Then two cases are possible in order for their meeting to occur:

   - Case 1: In the beginning, the agents took opposite directions, then eventually they will meet on a node or an edge.

   - Case 2: In the beginning, the agents took the same direction, then they will not meet until at least one of them passes by the home base of the other agent. Either they meet in one of the home bases (i.e. an agent arrives at the home base of the other agent while it is still asleep) or they do not meet on either home base (i.e. both agents woke-up and started already the computations), if they do meet at the home base, then they start the next step; otherwise each agent compares its own identifier with the other identifier written on the whiteboard and elects a leader (the agent with the smaller identifier). The leader will be waiting while the other agent reverses its direction and walks until they meet.

2. The agents reside in the same node, they walk in opposite directions (the leader picks its direction and the defeated follow the opposite direction) and count the nodes while cleaning the network until they meet again then terminate.

Theorem 6. *In order to find the size of a ring and capture the intruder, in the best case, the agents perform* $n+1$ *moves and consume* $n/2+1$ *time units, while in the worst case, they perform* $3n-1$ *moves and consume* $5n/2-1$ *time units.*

PROOF Since the two agents do not reside in the same home base, let us assume that they are separated by the distance d_1 on one side and by the distance $d_2 = n - d_1$ on the other side. For case 1, w.l.g. d_1 moves have to be made for the rendezvous to occur. Then agents have to make n moves to count the nodes of the network. That makes a total of $d_1 + n$ moves. In the best case $d_1 = 1$ so it takes only 1 move to meet, plus the n extra moves to clean, then a total of $n + 1$ moves to terminate; while in the worst case, $d_1 = n - 1$ so it takes $n - 1$ moves to meet, plus the n extra moves to clean, then a total of $2n - 1$ moves to terminate.

In terms of time, in the best case, the agents meet after only $n/2 + 1$ units, while in the worst case, they meet after $(n - 1)/2 + n/2 = n - 1$ moves. For the case 2, if the agents meet at the home base of an agent that is still asleep (w.l.g. let d_1 be the distance traversed by the moving agent), then d_1 moves are performed for the rendezvous to occur. On the other hand, if they both started to move, each of them will reach the home base of the other agent making a total of n moves, and based on the comparison of the identifiers marked on the whiteboards of the home bases, the agent with the higher identifiers will be performing an extra d_2 in order the rendezvous occurs. Then the agents will meet after $d_1 + 2d_2 = n + d_2$ moves where, w.l.g. d_2 is the distance corresponding to the portion of the ring traversed by the agent with the higher identifiers. Next, in the both possibilities, the agents have to make n to count the number of nodes of the network. For the first possibility, the total of moves is $d_1 + n$. This total at its maximum for $d_1 = n - 1$ which makes $2n - 1$ moves; and it is at its minimum for $d_1 = 1$ which makes $n + 1$ moves. For the second possibility, the total of moves is $2n + d_2$.This total at its maximum for $d_2 = n - 1$ which makes $3n - 1$ moves; and it is at its minimum for $d_2 = 1$ which makes $2n + 1$ moves. In terms of time, $2n - 2 + n/2 = 5n/2 - 2$ time units are required for the worst case. While in the best case, it takes $n/2 + 1$ time units. ∎

Case 1	Worst Case	Best Case
Number of moves	$2n - 1$	$n + 1$
Time	$n - 1$	$n/2 + 1$

Table 3.13: Finding the size of a Ring and intruder capturing for case 1 with 2 agents with distinct identifiers, starting from different home bases, using whiteboards.

Case 2	Worst Case	Best Case
Number of moves	$3n-1$	$n+1$
Time	$5n/2-2$	$n/2+1$

Table 3.14: Finding the size of a Ring and intruder capturing for case 2 with 2 agents with distinct identifiers, starting from different home bases, using whiteboards.

Global Results	Worst Case	Best Case
Number of moves	$3n-1$	$n+1$
Time	$5n/2-2$	$n/2+1$

Table 3.15: Finding the size of a Ring and intruder capturing with 2 agents with distinct identifiers, starting from different home bases, using whiteboards.

3.3.2 Model 6: Whiteboards but no distinct identifiers

1. The agents are anonymous.
2. Agents communicate via whiteboards or by using tokens.
3. Regardless if the agents do have common orientation or do not.
4. With intruder.

Parameters	Ring - Model 6
Communication Methods	Whiteboards
Identifiers	Anonymous
Orientation	No*

Table 3.16: Parameters of model 6 - Ring.

In this model, the availability of common orientation could not be exploited. Common orientation will not make any difference because there is no need to specify which agent has to go in which direction. Thus, algorithm 6 is used regardless the availability of common orientation.

Remark: In this model, rendezvous of the agents is a necessary step to capture the intruder. Since the agents are anonymous, then we cannot rely on the identifiers to break the symmetry and we have to find other techniques to do so. There exists initial symmetric configurations that

cannot be broken making the problem of finding n unsolvable. For example, if the anonymous agents are placed equidistantly in the ring, a synchronous schedule will force them to behave in the same way without allowing them to meet. In the following, we consider some particular asymmetric conditions under which the problem becomes solvable. There are two strategies to do so. In the first strategy, we can use their initial placements in the network to determine the meeting node or edge. The feasibility depends on the distance d that separates initially the agents, it has to be smaller than the half of the total number of nodes of the ring (i.e. $d < n/2$). If the distance $d = n/2$, then strategy 1 will not solve the problem, then a second strategy could be tried, the agents could take advantage of the fact that the model is asynchronous and meet on the home base of the agent that explored more nodes for the first time during the navigation. Algorithm 5 uses strategy 1 and solve the problem only when $d < n/2$. If the agents meet on an edge and they have common orientation, then they will be able to elect a node based on that. Otherwise the agents will not be able to break the symmetry without having a whiteboard to write on since there is no way to manage to meeting while they are passing by an edge.

Algorithm 6:

1. When an agent wakes-up, it marks down its respective home base by writing 1 or by releasing its token (the marks of the two agents are identical, since the agents are anonymous), starts walking in any direction and counts the nodes. Then two cases are possible in order they meet.

    - Case 1: In the beginning, the agents follow opposite directions, then eventually they will meet on a node or an edge.
    - Case 2: In the beginning, the agents follow the same direction. Maybe one agent catches the other agent asleep; but it is also possible that both of agents woke-up and started already the computations, then they will not meet until they break their symmetry. The agents count all the nodes in the ring and go to the middle of the smallest interval that separates them, wait for each other until they get together. During this step, if the agent meet they do not have to go to the computed node and execute the next step.

2. The agents reside in the same node, break their symmetry based on the access of the whiteboard (if the agents are using tokens, then to break their symmetry, they will try to elect a leader as the first agent arriving to the meeting point, or the agent that made more moves, or the agent with a home base closer to the meeting point. If none of these ideas brook the symmetry then terminate without solving the problem of intruder capturing) in order to be able to walk in opposite directions cleaning the network until they meet again then terminate.

Theorem 7. *In order to find the size of a ring and capture the intruder, in the best case, the agents perform* $n+1$ *moves and consume* $n/2+1$ *time units, while in the worst case, they perform* $(7n-1)/2$ *moves and consume* $(7n-1)/4$ *time units.*

PROOF The two agents do not reside in the same home base, then let us call the distances that separated them on both sides of the ring d_1 and d_2 as $(d_2+d_1=n)$ and let us call d the distance between the agents and the node at the middle middle of the smallest interval that separates them. In case 1, w.l.g. the agents meet after d_1 moves and make n extra moves to clean and terminate. That makes a total of $n+d_1$ moves. So, in the best case, $d_1=1$ and the total number of moves is $n+1$, while in the worst case, $d_1=n-1$ and the total number of moves is $2n-1$. In terms of time, in best case, only $n/2+1$ units is enough, while $(n-1)/2+n/2=n$ in the worst case. For case 2, each agent makes $d_1+d_2=n$ moves to compute the meeting nodes and makes d moves (at most $n-1/4$) moves to reach it. Then the agents make n moves together to meet again and terminate. The total of moves is $2d_1+2d_2+2d+n$. In the best case, $d_1=1$, the agent makes one move and gets to the home base of the second agent while it is still asleep (it took only 1 move to meet, which equivalent to 1 time unit). Adding n extra moves to count, clean and terminate in $n/2$ time units, the total is $1+n$ moves in $n/2+1$ time units in the best case. While in the worst case, both of the agents get up and start computing. Each of them make n moves to compute the meeting node and make an extra $(n-1)/4$ moves to get there. That makes a partial total of $2*(n+(n-1)/4)=(5n-1)/2$ moves which corresponds to $n+(n-1)/4=(5n-1)/4$ time units. Adding n extra moves to count, clean and terminate in $n/2$ time units, the total is $(7n-1)/2$ moves in $(7n-1)/4$ time units in the worst case. In order the two agents meet, in the overall best case, 1 moves and 1 time unit are sufficient; while

for the overall worst case, there is a need for $(5n-1)/2$ moves and $(5n-1)/4$ time units. An extra n moves and $n/2$ time units are needed for the last step in all the different cases to clean the network. Then for this model, in order to find the size of the ring and clean it, in the best case, it takes $n+1$ moves and $n/2+1$ time units; while in the worst case, it takes $(7n-1)/2$ moves and $(7n-1)/4$ time units. ∎

Case 1	Worst Case	Best Case
Number of moves	$2n-1$	$n+1$
Time	n	$n/2+1$

Table 3.17: Finding the size of a Ring and intruder capturing for case 1 with 2 anonymous agents, starting from different home bases, using whiteboards.

Case 2	Worst Case	Best Case
Number of moves	$(7n-1)/2$	$n+1$
Time	$(7n-1)/4$	$n/2+1$

Table 3.18: Finding the size of a Ring and intruder capturing for case 2 with 2 anonymous agents, starting from different home bases, using whiteboards.

Global Results	Worst Case	Best Case
Number of moves	$(7n-1)/2$	$n+1$
Time	$(7n-1)/4$	$n/2+1$

Table 3.19: Finding the size of a Ring and intruder capturing with 2 anonymous agents, starting from different home bases, using whiteboards.

3.4 Remarks and observations about finding the size of a ring

- **The need of face-to-face communication method:** Face-to-Face is always possible regardless the availability of other communication methods. This assumption is necessary to make the agents cooperate when they are using only tokens, then they can perform

partial nodes counting, each on its side, and cooperate by making the sum of their nodes to have the total nodes.

- **Limits of blackboards models:** The availability of a blackboard is equivalent to a central communication control. The availability of a blackboard might seem to make the model always stronger than whiteboard models, however it is not always true. In fact, when the agents and the nodes are anonymous, the agents will not be able to mark and recognize their own home base in the blackboard model. In this case, one agent is not only insufficient to capture an intruder but also not able to explore a ring and count the nodes. A second agent is needed to cooperate at least by playing the role of a token in order for the other agent to terminate the computation. Therefore, even if no intruder is involved, a meeting is necessary as a key step in order to explore the ring.

- **Limits of one agent models:** To find the size of the ring, it is necessary to visit all the nodes, thus at least n moves are required. This could be accomplished by using a single agent, however the intruder capture problem can not be solved.

- **Performance of 2 agents models:** Deploying 2 agents from the same home base, consumes the same number of moves as deploying a single agent to count the nodes but consumes only half of the time compared to the solution with one agent. Furthermore, it can solve the intruder capture problem if there is any.

- **Importance of starting from the same home base:** Having all the deployed agents starting from the same home base facilitates the solution of the problem. In this case, there is no need to rendezvous as a preliminary step when there is an intruder in the model or when the blackboard is used for communication. In addition, only n moves and $n/2$ time units are needed, which is optimal when 2 agents are deployed.

- **Effect of common orientation on the ring exploration:** The fact that agents have or have not common orientation did not make any difference for the complexity of solution of the ring exploration. There are two reasons for this observation depending whether the

agents started for the same home base or not.

- △ When the agents start from the same home base, they can simply go in opposite directions without a need to know if the this direction is the left side or the right side of the ring. The local labelling it is all what the agents use to keep going in same direction.
- △ Based on the assumption that there is no beforehand agreement to break the roles between agents, when they start computing by executing the same algorithm from different home bases, using their common orientation, they will always start walking in the same direction if they have common orientation. While it is more convenient to stat walking in opposite direction for faster meeting, which could be possible when they do not have common orientation or they simply do not use it.

- **Effect of having more than 2 agents on the ring exploration:** Definitely deploying more than 2 agents will not help to minimize the number of moves. Moreover, it causes a penalty to inform the other agents that were waiting in the home base about n if there is no blackboard. When 2 agents are deployed, in the same home base, to find the size of the ring and clean it, they need n moves which is the same number of moves needed by one agent; and it is the lower bound to solve the problem for finding the size of the ring that requires to visit all the nodes. While a single agent consumes n time units, 2 agents consume only half of that amount $n/2$ time units. When k agents are deployed to find the size of a ring, using the blackboard to communicate and there are placed in $k/2$ different home bases, two agents per home base, then they will perform n moves but they will consume n/k time units when they are able to break their symmetry to go in the opposite home base, and computing synchronously.

- **When the Ring Exploration problem is unsolvable deterministically?:** The ring exploration problem is unsolvable deterministically when:

  - △ Under our assumptions, regardless the number of agents that are deployed, if they reside in different home bases and can communicate only face-to-face (no blackboard

or whiteboards are available). In the case that only one agent is deployed then it cannot mark its own home base and will keep walking without a termination condition. If more agents are deployed they will not be able to meet and use one out of them as a marker, even if they have common orientation unless we assume the possibility of breaking the roles based on beforehand agreement; in that case, the problem becomes solvable.

△ Agents having only tokens (whiteboards + anonymous + synchronous, etc) they cannot solve the problem of the intruder capture when the distance d that separates them is equal to $n/2$ since they cannot break their symmetry and meet as a prerequisite step for solving intruder capture problem.

△ If one agent only is deployed and it has only a blackboard, it will not be able to mark its own home base and therefore it is incapable to solve the exploration ring problem. Once again it will walk without a termination condition. For the blackboard model, two agents are necessarily and sufficient to solve this problem.

Chapter 4

Finding the size of a Mesh

Mesh topology is a matrix of nodes, each with connections to its nearest neighbors.

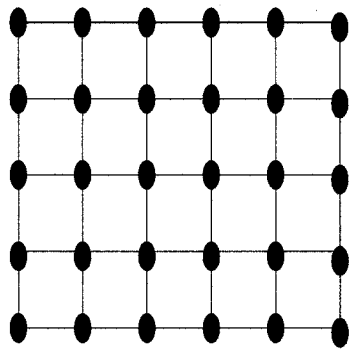

Figure 4.1: Mesh Topology

It derives from the definition of the topology that every interior node V is directly connected to 4 nodes, in other words, it has 4 neighbors at distance 1, referred in this thesis as 1-neighborhood nodes.

Also, there is a total of 8 nodes that are at distance 2 (the 1-neighborhood nodes of the 1-neighborhood node of V), and connected to V through its 1-neighborhood nodes, referred as 2-neighborhood.

The problem of finding the size of a mesh corresponds to the problem of counting the number of the nodes on its width i and its length j and then compute $n = i * j$.

Since there is no need to visit all the nodes in the network to solve the primary problem, then

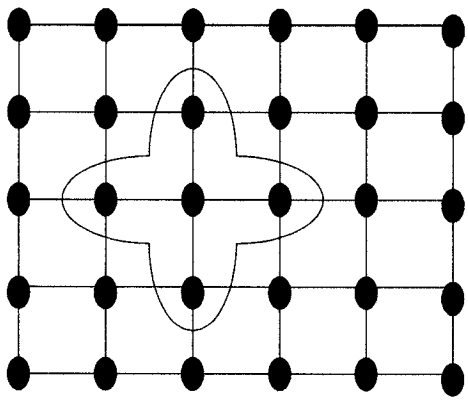

Figure 4.2: 1-neighborhood nodes

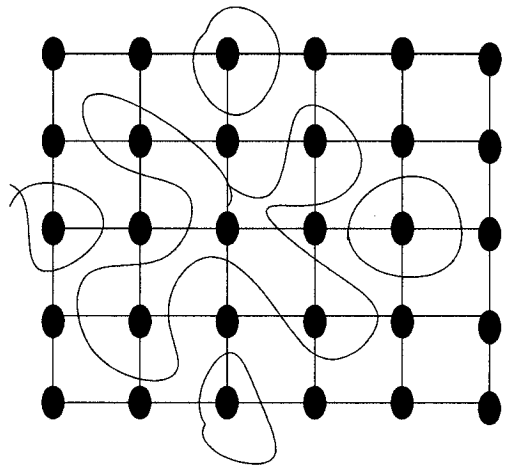

Figure 4.3: 2-neighborhood nodes

the problem of capturing an intruder will not be considered in the subsequent models. In other words, in the following we assume that the system is homogenous.

4.1 Algorithms for a single agent

4.1.1 Model 1: Single agent having orientation

1. Regardless of the identifiers of the agent (distinct or anonymous).
2. Regardless the mean of communication.
3. Agent does have orientation (North, South, East, and West).

Parameters	Mesh - Model 1
Communication Methods	Face-to-face*
Identifiers	Anonymous*
Orientation	Yes

Table 4.1: Parameters of model 1 - Mesh.

In this model, there is no need for distinct identifiers since a single agent is deployed. The communication method was not exploited. It does not make any difference since the agent

is using its own memory to count the node and it is not sharing that information with any other agent. Thus, algorithm 1 is used regardless the identifiers and the available method of communication.

Algorithm 1:

1. When the agent wakes-up, it walks toward the north direction until it gets to a node of type border (to the node with 2 or 3 incident edges).

2. Then the agent walks toward the south direction while counting the nodes (counting j) until it gets to a node of type the border.

3. Next,the agent walks toward the east direction until it gets to another node of type border (to the node with 2 or 3 incident edges).

4. after that, the agent walks to the west direction while counting the nodes (counting i) until it gets to a node of type border.

5. The agent computes n by multiplying $i * j$ and terminates.

Theorem 8. *In order to find the size of the mesh, in the best case, the agent performs* $i + j$ *moves and consumes* $i + j$ *time units, while in the worst case, the agent performs* $2i + 2j - 2$ *moves and consumes* $2i + 2j - 2$ *time units.*

PROOF In order to find the size of the mesh, first, the agent has to get to a node of type border. Let us assume that the agent performs x moves to get there, then performs i moves to get the nodes on the width of the mesh. Second it performs y moves to get to the other border, then performs j moves counting the nodes and the length of the mesh and terminates. That make a total of $x + y + i + j$ moves. In the best case, $x + y = 0$, which is the case of the agent starts computing from a corner (a node with two incident edges), then performs only $i + j$ moves and consumes $i + j$ time units. However, in the worst case, $x + y$ is at its maximum. Then $x = i - 1$ and $y = j - 1$. Thus, the agent starts computing from the node having the coordinates (1, 1), considering the corner as the origin of the referential system. The agent performs $2i + 2j - 2$ moves and consumes $2i + 2j - 2$ time units. ∎

Remark: The fact that the agent starts the computations by counting i or j first, does not affect the correctness nor change the complexity of the algorithm.

Global Results	Worst Case	Best Case
Number of moves	$2i + 2j - 2$	$i + j$
Time	$2i + 2j - 2$	$i + j$

Table 4.2: Finding the size of a rectangular mesh with a single agent having orientation.

4.1.2 Model 2: Single agent without orientation

1. Regardless of the identifier of the agent (distinct or anonymous).
2. Communication method: Whiteboards .
3. The agent does not have orientation.

Parameters	Mesh - Model 2
Communication Methods	Whiteboards
Identifiers	Anonymous*
Orientation	No

Table 4.3: Parameters of model 2 - Mesh.

In this model, there is no need for distinct identifiers since a single agent is deployed.
In case of lack of common orientation, we consider the following assumptions:

- The nodes with two or three incident edges are considered as nodes of type border.
- The nodes with four incident edges are considered as nodes of type interior.
- The nodes with two incident edges are considered as nodes of type corner.

We describe the algorithm using whiteboards, however the solution can be adapted to be used for blackboard, tokens and face-to-face communications methods.
Without common orientation, a trivial solution to find the size of a mesh is as follow: the mobile

agent visits all the nodes and traverses all the edges. The complexity of this solution is $O(n)$. The solution described by the algorithm 2, that we present below, works in $O(\sqrt{n})$. The preliminary step to count the number of nodes in an undirected mesh is to get to a node of type border, to do so in the following algorithm, the agent will construct a diagonal to be able to walk in a consistent direction. Then, the agent has to walk in a special way, moving on the nodes of type border until it gets to a node of type corner. At this stage, the agent starts counting the nodes on the length and the width of the mesh using similar walk to the one employed to walk on the diagonal of the mesh.

Algorithm 2: When the agent wakes-up, it checks its home base to see if it is a node of type border. If it is the case then it starts *GetToCorner* phase (defined later), otherwise starts the *Diagonal Construction* phase, described below.

Diagonal Construction Phase: The goal of this phase is to get the agent in a node of type border with as few moves as possible. Although without orientation, the agent will be able to move in an arbitrary consistent direction. The agent will compute the diagonal of the mesh, starting from its home base, it keeps walking on a consistent direction without going trough a global orientation construction.

The home base of the agent is considered as the first node of the diagonal. Notice that, given a node x in the diagonal, the next node of the diagonal is a node at distance 2 from x, at the intersection between the 2-neighbors of x. Following this observation, the agent performs two steps. The first step is meant to mark the nodes that are candidates to be part of the diagonal and the second step is to compute the next particular node to be added to the diagonal.

The termination condition of the `Diagonal Construction` phase is the following: Whenever the agent visits a node for the first time, it tests if it is a node of type *Border* or not, if it is the case, then it terminates the *Diagonal Construction* phase and starts the *GetToCorner* phase, otherwise it continues the *Diagonal Construction* phase .

Step 1: Starting from the last node of the diagonal (it is the home base of the agent in the first round), the agent picks an incident unexplored edge, gets to the node at its end, and explores all its unexplored incident edges at to reach the 3 nodes at distance 2 from the last node of the

diagonal, one by one as follow: the agent picks an edge, gets to the node in its end, checks if it is type border if yes then starts the *GetToCorner* phase otherwise it marks the $Round_k$ number on its corresponding whiteboard (one of these nodes will become part of the diagonal the next time it is visited), gets back to in order to check the next edge until all the incident edges are explored.

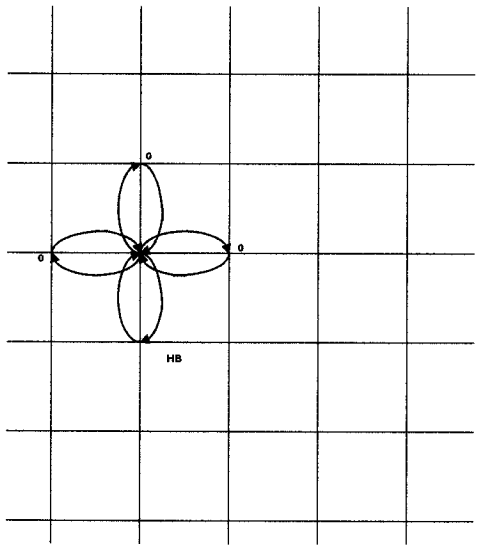

Figure 4.4: Step 1 starting from the home base

Step 2: The agent gets back to the node on the diagonal, it picks another unexplored edge, and visits all the nodes at distance two from the node on the diagonal as it did in step 1, but instead of marking the whiteboards it checks which node is already visited in step 1. When the agent finds that node, then it adds it to the diagonal, increments the round number: $Round_k = Round_{k+1}$ and restarts the computations from there until the termination condition is satisfied.

REMARK: When the agent starts the computation at its home base, then it is possible that it has to perform the step 2 twice in order to find the first node to add to the diagonal. Since the home base has 4 incident edges that are all unexplored in the beginning, so maybe the agent picks in step 2 a parallel edge to the one that it picked in step 1. For the the subsequent added nodes to the diagonal, each of them got only two unexplored edges since the node itself is computed by

exploring two of its incident edges.

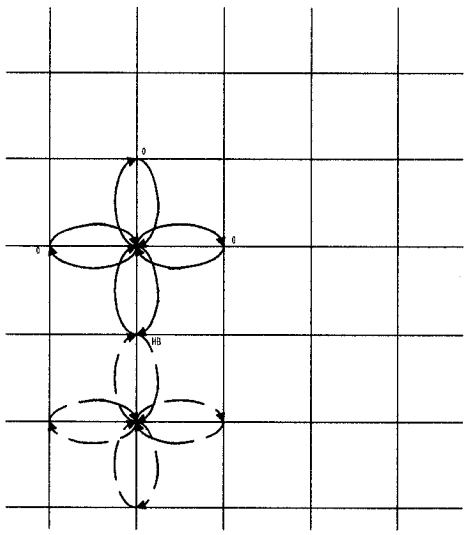

Figure 4.5: Step 2 starting from the home base using the non perpendicular edge

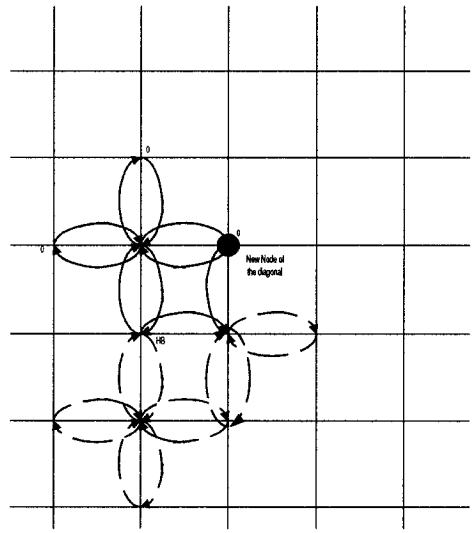

Figure 4.6: Repeating step 2 using a perpendicular edge

GetToCorner Phase: Starting from a node of type border, the goal of this phase is to get the agent to a node of type corner with as few moves as possible based on the degree of the nodes. The termination condition of the `GetToCorner` phase is the following: Whenever the agent visits a node for the first time, it tests if it is type of *Corner*, if it is the case, then it starts the *Counting* phase, otherwise it continues the `GetToCorner` phase.

The agent picks an incident edge to its current node, if it leads it to a node of type interior, then it backtracks using the same edge to get back to its last node, marks that on the whiteboard and tries the second edge that will leads him for sure to a node of type border, otherwise it continues its walk as long as it did not get in a node of type corner.

Counting Phase: The goal of this phase is to count the nodes on the width i and the length j of the mesh. The agent keeps using the same walk described in the *GetToCorner* phase, from the first corner to the second one (left behind) counting i, when it gets to the second corner start

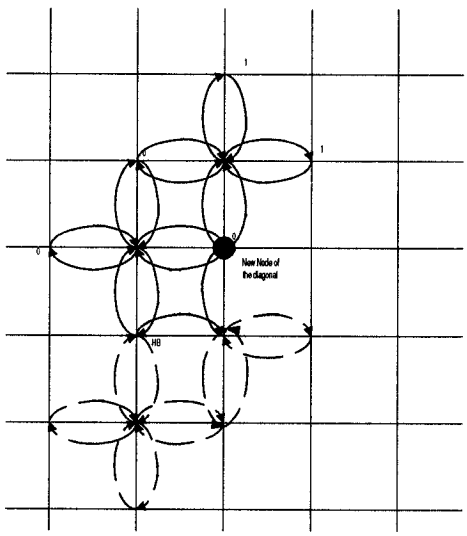

Figure 4.7: Step 1 from the second node of the diagonal

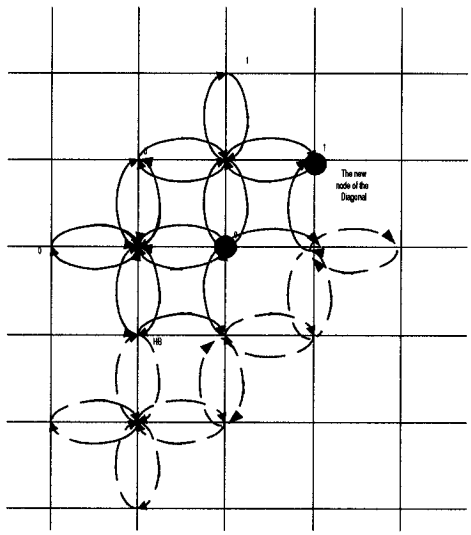

Figure 4.8: Step 2 from the second node of the diagonal

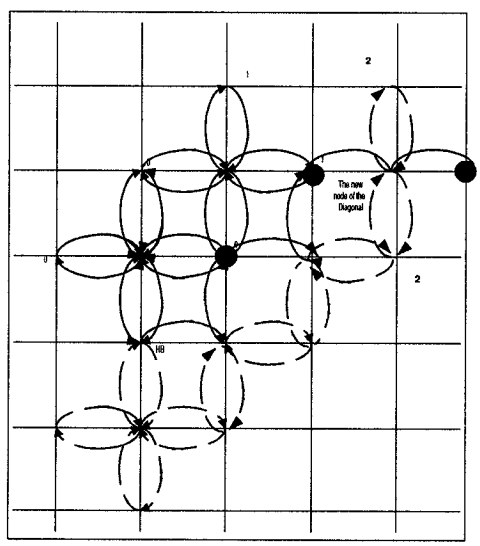

Figure 4.9: Step 1 - The agent reaches a node of type border

counting j until it gets to the third corner, then computes $n = i * j$ and terminates.

The parameters passed to the *Diagonal Construction* phase are *round_number*: the round number initialized with 0 and the last node of the diagonal which is marked as the home base of the agent.

Definition of Algorithm 2:

```
main()
{
  /*Declaration of the variables*/
  int number_node =0;/* Size of the mesh*/
  int i=0; /*Counter*/
  int j=0;/*Counter*/
  node diagonal_end = agent's home base; /*Starting node*/
  int round_number =0; /* to differentiate the rounds*/

  DiagonalConstruction(diagonal_end, round_number);
  GettoCorner(diagonal_end);
  Counting(border_end, i, j, round_number);
}
```

Definition of DiagonalConstruction Function:

```
/* The DiagonalConstruction function gets as an input two parameters: the variable diago-
nal_end is the last node of the diagonal,
round_number is the round number.
At the beginning, the variable diagonal_end is equivalent to the agent's home base and the vari-
able round_number is 0 */
DiagonalConstruction(diagonal_end, round_number)
```

```
{
  FindNext(diagonal_end);
  For each unexplored neighbor diagonal_end_neighbor of diagonal_end :
    /*An unexplored neighbor is the node at the end of unexplored incident edge*/
    Expand(diagonal_end, diagonal_end_neighbor, round_number);
}
```

Definition of Expand Function:

```
Expand(diagonal_end, diagonal_end_neighbor, round_number)
{
  Go to (diagonal_end_neighbor);
  VisitNeighbor(diagonal_end, diagonal_end_neighbor, round_number);
  Go back to (diagonal_end);
}
```

Definition of VisitNeighbor Function:

```
VisitNeighbor(diagonal_end, diagonal_end_neighbor, round_number)
{
  For (each unexplored neighbor diagonal_end_second_neighbor of diagonal_end_neighbor)
    Go to diagonal_end_second_neighbor
    {
    if (diagonal_end_second_neighbor is already marked with phase number round_number)
    then
    {
      round_number is round_number + 1;
      diagonal_end is diagonal_end_second_neighbor;
```

```
        TERMINATE;
      }
      else
      {
        mark diagonal_end_second_neighbor with phase number round_number;
        Go back to (diagonal_end_neighbor);      }
    }
}
```

Definition of GettoCorner Function:

```
GettoCorner(diagonal_end)
{
  Go to border_end, the unexplored neighbor of diagonal_end;
  if(border_end is of type Corner)
  then
  {
    TERMINATE;
  }
  else if (border_end is of type Border)
  {
    GettoCorner(border_end);
  }
  else
  {
    mark that leading edge as interior;
    Go back to (diagonal_end);
  }
```

}

Definition of Counting Function:

/* i and j are the counters used to count respectively nodes on the width and the length of the mesh. In the beginning of the counting, the variable border_end is a corner*/ **Counting(border_end, i, j, round_number)**

{

Go back to the last visited node; /*We use the partial path that we

explored already and had marked the edges leading to nodes of type interior in order to count the nodes. Thus, we save some unnecessary moves*/

i++;

while (current_node is not of type of corner) /* Counting i*/

{

i++;

while (the edges are identified interior or non-interior and current_node is not of type of corner)

{

Go to the next node of type border following the non-interior edge;

i++;

}

Go to border_end, the unexplored neighbor of current_node;

if (current_node is of type interior)

then

{

Go back to the last node coming from and explore the second unexplored edge;

i++;

}

```
  }
/*Exiting the last loop, the current_node is a corner, we have to reach the next untouched node
to count j */
  Go to border_end, the unexplored neighbor of current_node;
  while(current_node is not of type of corner)/*Counting j*/
  {
    Go to border_end, the unexplored neighbor of current_node;
    if(current_node is of type interior)
    then
    {
      Go back to the last node coming from and explore the second unexplored edge;
    }
    j++;
  }
number_node = i * j
}
```

Remark about the other communication methods: Whiteboards were specified as the communication method in this model, but the solution can be generalized to suit the other communication methods with some extra additional assumptions.

For the tokens models, the agent has to be able to generate as many tokens as it needs to mark the nodes that are marked in the algorithm described using the whiteboards in this section. It would be interesting to determine the minimum number of tokens needed for the tokens models in order to find the size of a mesh.

In the blackboard models, the nodes need to be distinctly labelled or must have enough memory space so the agent can mark them and distinguish them later. For face-to-face models, the memory space of the nodes is still needed and the agents need to have enough memory to store the

partial map to relocate the marked nodes. Thus, algorithm 2 is used regardless the identifiers and it can be adapted to any specific available method of communication.

Correctness and Complexity of Algorithm 2:

Theorem 9. *Using Algorithm 2, the agent picks the right node to move to in order to keep walking on the diagonal.*

PROOF We have to prove by induction these following items:

- There is a single 2-neighborhood intersection node between two 1-neighborhood nodes at the end of two incident perpendicular edges.

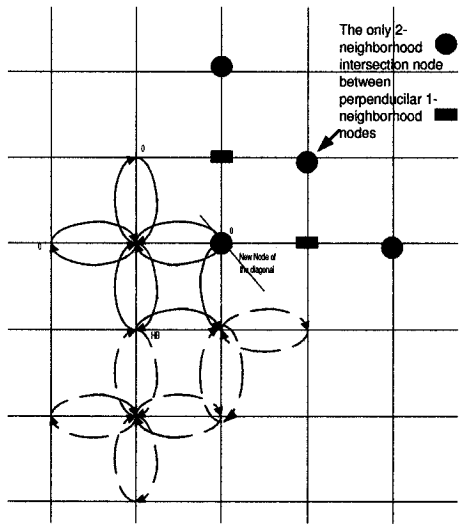

Figure 4.10: 2-neighborhood intersection

- Algorithm 2 finds this unique intersection node, by expanding the nodes at the end of two perpendicular edges.

Let us assume that the agent chooses the nodes to move to correctly up to $X_{i-1}{}^{th}$ node (i.e. up to choice i is correct). We have to prove that the next choice is correct (i.e. X_i is the right node to move to).

The fact that there is a single 2-neighborhood intersection node between two perpendicular incident edges of two 1-neighborhood nodes derives automatically from the nature of the mesh topology. Once we choose a direction at round 0 (at the home base), each of the subsequent nodes of the diagonal has only two unexplored edges which are perpendicular. These two edges are expanded by the algorithm and the unique intersection is determined as the next node of the diagonal.

Thus, the agent makes the correct choice of the next node to go in straight direction. The agent will be always able to locate this unique new node to add to the diagonal which directs it consistently toward the border. At round 0, this node is not unique, the agent has to pick the first intersection node to mark off a direction. The orientation is irrelevant. Later on, the last node of the diagonal has only two unexplored edges which are perpendicular, since the node itself is computed by exploring the two other perpendicular edges. For the subsequent node, the procedure *Expand* will be called twice on the two unexplored perpendicular edges. Then there will be always two expansions: one performed fully to mark all the nodes at 2-neighborhood, the second one visit the 2-neighborhood nodes until it get interrupted when it touches a 2-neighborhood node previously visited. Then Out of all the nodes, there is only one node at distance 2 from X_{i-1} This node is X_i and is the next node in the diagonal. ∎

Theorem 10. *In order to find the size of a square mesh (e.g. i=j), in the best case, the agent performs i moves and consumes i time units, while in the worst case, the agent performs* $17\sqrt{n}-29$ *(for* $n \geqslant 4$*) moves and consumes* $17\sqrt{n} - 29$ *time units.*

PROOF To find the size of a square mesh, first, the agent has to manage to get to a node of type border, then has to walk in a particular way to stay on the border until it gets to a corner and finally starts counting the nodes on the length and the width of the mesh.

To get to a border without having any orientation, the agent will compute the diagonal. Aiming to compute the second node of the diagonal, assuming that the home base as the first node of the diagonal, *Expand* will be called up to three times and only twice for the next nodes, it will be performed fully (the agent will perform 8 moves to reach the nodes at the distance 2 from the last node of the diagonal) and the second time, this function will be interrupted when the intersection

point is found. At this point, in the best case with only 2 extra moves it will compute the next node of the diagonal which is it happens to be the first visited node at distance 2; however in the worst case, the agent will make 8 moves without succeeding in visiting any node for 2 times since it started this 8 exploration from an edge that is in opposite direction of the previously explored edge. Consequently, the agent will make between 2 and 4 moves to get the node that will be part of the diagonal (in the worst case, the agent performs 4 moves). That makes a total of 20 moves to compute the first added node to the diagonal in the worst case.

To find the following nodes of the diagonal, expecting the last node, the agent performs 8 moves within the call of *Expand* marking the nodes that are candidates to be part of the diagonal and makes up to 6 moves in the worst case to compute the right node to add. To finish the `Diagonal Construction` phase, the agent makes between 1 (the first call of *Expand* is interrupted since the visited node is of type border) and 12 moves to compute the last node that is of type border. In the worst case the agent will be computing the longest possible diagonal, so it will start from the node with coordinates $(3, 3)$ in the referential system having the corner in the most south-west as origin (this is the reason behind considering $n \geqslant 4$ in the theorem), this because the *Expand* function touches the 2-neighborhood nodes, which are at the distance two from the home base, and in the worst case *Expand* function misses the closest border that is only 1 move away. Then the diagonal is at its maximum for $i - 4$, 4 nodes will not be computed, the 3 nodes left behind since the starting node is at $(3, 3)$ coordinates and the home base itself that by default part of the diagonal.

For the `DiagonalConstruction` phase, in the worst case, the agent will be performing $20 + 14 * (i - 4) + 12$ moves. If we consider $i = j = \sqrt{n}$ this equation is equal to $20 + 14 * (i - 4) + 12 = 20 + 14\sqrt{n} - 56 + 12 = 14\sqrt{n} - 24$; while in the best case the agent starts from a corner so it does not have to make any move to get to a border. With the `GetToCorner` phase, in the worst case the agent will be performing 3 moves to go from a border node to another border node, assuming that it will always follow first the edge that lead it to an internal node, then it backtracks to get back to its previous node and follows the third and the last unexplored edge to get to the next node of type border. The starting node that causes the worst case for the

`GetToCorner` phase is the node with the coordinates (i, j-1) in the same referential system. Then the agent needs a total of $3(j-2)+1 = 3\sqrt{n}-5$ moves to get done with the `GetToCorner` phase. The substraction of 2 moves is due to the fact that one move is already counted to get to the first node itself, and the second subtracted move resulting of the fact, two moves are leading the agent to two different nodes of type border but one of them is already the corner so in the worst case the agent will make the longest path, following the first picked edge ; while in the best case the agent starts from a corner so it does not have to make any move to get to a corner.
For the counting phase, the agent will not be trying all the incident edges since it marked the edges leading him to the interior nodes. Then in the worst case, counting the nodes on the length will cost the same number of moves as it the best case, only i moves. We do not need to count the nodes on the width since the mesh is square. The overall total of the moves in the worst case is $17\sqrt{n} - 29$ while in the best case is i moves. ∎

Global Results	Worst Case	Best Case
Number of moves	$17\sqrt{n} - 29$	i
Time	$17\sqrt{n} - 29$	i

Table 4.4: Finding the size of a square mesh with a single agent without orientation.

Remark: Because of the lack of orientation, the agent has to keep track of the visited nodes and the traversed edges. If the whiteboards are handy, then the agent will simply mark the visited nodes whiteboards, equivalently he can use tokens to differentiate the nodes visited for the first or the second time(more than 12 tokens are needed). Using a blackboard or its own local memory, the agent has to keep track of the visited nodes by constructing and maintaining a partial map of graph but it has to be able to attach a local but unique node' labelling for each visited node (e.g. the node has space to contain its label).

4.2 Algorithms for two agents

In all the following models, two agents are deployed to find the number of nodes of a mesh.

4.2.1 Model 3: Blackboard with orientation

1. Regardless of the identities of the agents (distinct or anonymous).
2. Agents communicate using a blackboard.
3. Agents do have common orientation.
4. Regardless if the agents reside in the same home base or in different ones.

Parameters	Mesh - Model 3
Communication Methods	Blackboard
Identifiers	Anonymous*
Orientation	Yes

Table 4.5: Parameters of model 3 - Mesh.

The fact that agents are distinctly identified cannot be used since they are only two agents and thus they can always break their symmetry based on the first access to the blackboard. Thus, algorithm 3 is used regardless of the availability of distinct identifiers.

Algorithm 3:

1. When the agent wakes-up, it tries to access the blackboard; if it is the first one to succeed, then it will play role 1 otherwise role 2. If both agents reside in the same home base then the first agent that wakes up, makes sure that the other agent is awake too.
2. The agent playing role 1 walks in the north direction until it gets to the border (to the node with 2 or 3 incident edges). Then it walks to the south direction while counting the nodes (counting j) until it gets to the other border.
3. the agent playing role 1 writes down j and keeps checking the blackboard until it finds the value of i, written down by the other agent, computes n (i*j) and terminates.
4. The agent playing role 2 walks in the east direction until it gets to the border (to the node with 2 or 3 incident edges). Then it walks to the west direction while counting the nodes (counting i) until it gets to the other border.

5. The agent playing role 2 writes down i and keeps checking the blackboard until he finds the value of j, written down by the other agent, computes n (i*j) and terminates.

Theorem 11. *In order to find the size of a rectangular mesh, in the best case, the agents perform $i + j$ moves and consume $max(i, j)$ time units, while in the worst case, the agents perform $2i + 2j - 2$ moves and consume $max(2i - 1, 2j - 1)$ time units.*

PROOF In order to find the size of a rectangular mesh, in the best case, the agents start computing from the corner (a node with two incident edges), perform $i+j$ moves and consume $max(i, j)$ time units, while in the worst case, the agents start computing from the node having the coordinates (1, 1), considering the node in the most north/west as the origin of the referential system. The agents perform $2i + 2j - 2$ moves and consume $max(2i - 1, 2j - 1)$ time units. ∎

Global Results	Worst Case	Best Case
Number of moves	$2i + 2j - 2$	$i + j$
Time	$max(2i - 1, 2j - 1)$	$max(i, j)$

Table 4.6: Finding the size of a rectangular mesh with 2 agents having common orientation and using blackboard.

4.2.2 Model 4: With common orientation but no blackboard

1. The agents can be anonymous if they are using whiteboards or tokens.
2. The agents are distinctly identified if they are using face-to-face only.
3. Agents communicate using any method of communication, except a blackboard (whiteboards, tokens, or face-to-face).
4. Agents do have common orientation.
5. Regardless if the agents reside in the same home base or in different ones.

To break the symmetry, first, the agents have to be in the same node. The presence of common orientation solves the rendezvous problem that is needed to break the symmetry of the agents when they do not reside in the same home base. It is trivial for the agents to meet, such as

Parameters	Mesh - Model 4
Communication Methods	face-to-face*
Identifiers	Anonymous*
Orientation	Yes

Table 4.7: Parameters of model 4 - Mesh.

gathering on to the most north east node. When the agents have access to whiteboards or tokens then they break their symmetry based on the first access to the whiteboard or the placement of the token. However, in the case of face-to-face communication, in order the make the agents cooperate they have to be distinctly identified so they can compare the identifiers to break their symmetry. If the agents are anonymous and have only face to face communication then each agent will be computing on its side the number of node as it is in the algorithm 1 performed by a single agent having orientation. Thus, algorithm 4 recapitulates all the possibilities of the identities of the agents and corresponding communication method, except the availability of a blackboard.

Algorithm 4:

1. Agents that are anonymous and using face-to-face method of communication only, have to perform Algorithm 1 each on its side. They cannot switch back to this algorithm to cooperate unless if they meet by chance during their process.

2. If the agents do not reside in the same home base and none of them has a corner as home base, then they have to go to the node in the most north/east and wait for each other until they get together. (if they meet by chance before getting to the node in the most north, then they interrupt this step and go to the following step).

3. If one of the agents has a corner as a home base then it computes i or j while moving to the meeting node (the most north/east).

4. When the agents get together, they verify if one of them had computed partially or totally the lengths of i and j. If both of i and j are computed then agents exchange information and terminates. If i or j is computed, then the agents walk together to count the second

parameter. If none of them (i, j) was computed, then they break their symmetry (based on the first access to the whiteboards or token placement or using their distinct identifiers) to share the roles, then one of them counts the nodes on the width i and the other agent counts the nodes on the length j. If the agents are not able to break their symmetry (which is the case for anonymous agents using only face-to-face mean of communication) then each of them has to count by himself the nodes on the length and the width of the mesh i and j.

5. If the agents were able to break their symmetry and had different roles, then they have to go back to the most north/east and wait for each other until they get together to exchange the values of i and j, compute $i * j$ and terminate.

Theorem 12. *In order to find the size of a rectangular mesh, in the best case, the agents perform* $2i + 2j$ *moves and consume* $max(2i, 2j)$ *time units, while in the worst case,* $4i + 4j - 5$ *moves are needed which corresponds to* $i + j - 2 + max(2i, 2j)$ *time units.*

PROOF In order to find the size of a rectangular mesh, the best case occurs when the agents reside in the same home base, regardless of the position of this node in the network. In addition to that they have distinct identifiers or dispose of whiteboards or tokens. Notice that there are many other setting for the best case. Let us say that this node has coordinates (x, y) in the referential system having the most south/east node as an origin. Let us assume the agent 1 will be counting the nodes on the axis of i , in the same referential system defined above, has these coordinates (x_1, y_1). Then it will make $i - x_1$ moves to get to the border; then makes i moves to count i and finally x_1 moves to get back to its home base. That makes a total of $2i$ moves for agent 1. Similarly, agent 2 will make $2j$ moves. When the agents do not reside in the same home base. The worst case occurs when, agent 1 resides in the node with coordinates (1,1) and agent 2 resides in the node with coordinates (1,2), so in their way to the meeting node, none of them will pass by the node of type corner and arriving there, both of the agents did not count any parameters (i,j). In order to get to the most north/east node, agent 1 has to make $i + j - 2$ moves, while agent 2 has to make $i + j - 3$ moves. After getting to the meeting node and sharing the roles, an extra $i + j$ moves is needed to count i and j and an other $i + j$ moves to get that

meeting node to exchange the values of i and j. Therefore the total of moves needed in the worst case is $4i + 4j - 5$. Comparing these results, with the case of anonymous agents having only face-to-face communication method, it turns out that this last case is the worst case, the agents needs $4i + 4j - 5$ moves but only $i + j - 2 + max(2i, 2j)$ time units ∎

Remark: Trying to save some moves by making the 2 deployed agents cooperate, in the worst

Global Results	Worst Case	Best Case
Number of moves	$4i + 4j - 5$	$2i + 2j$
Time	$i + j + 1 + max(2i + 2j)$	$max(2i, 2j)$

Table 4.8: Finding the size of a rectangular mesh with 2 agents having common orientation but no blackboard to communicate.

case they end-up consuming more moves than if they have solved the problem each on its side. The agents consume $4i+4j-5$ moves, while they where able to solve the problem separately and consuming only $2i+2j$ moves. When the agents do not reside in the same home base, sometimes is not worth to make them meet in order to break their symmetry and cooperate to solve the problem.

4.3 Remarks and observations about finding the size of a mesh

- **Other models using two agents having no orientation:**

  1. Regardless of the identities of the agents (distinct or anonymous).
  2. Regardless of the mean of communication
  3. Agents do not have orientation.
  4. Regardless if the agents reside in the same home base or in different ones.

  If the agents are dispersed then their only chance to meet is going to be by chance. The fact that agents are anonymous and using only face-to-face communication then they are not able to break their symmetry and share the tasks even if they have a common home

Parameters	Mesh - extra Models
Communication Methods	face-to-face*
Identifiers	Anonymous*
Orientation	No

Table 4.9: Parameters ofextra Models - Mesh.

base. At the beginning, each agent has to start computing by itself. If they meet by chance during their process they might be able to exchange the learned information and help each other finishing the counting. Considering that we are in rectangular mesh, in the best case, the agents reside in opposite corners, and it happens that each move they make leads them to a node of type border, then the they perform only $i + j$ moves in max (i, j) time units and meet, exchange information, compute the number of the nodes, and terminate. If the agents reside in the same home base which is an internal node, then they have to be able to break the symmetry in such way the agent playing the role 1 will be computing the diagonal and the second agent will be passively following agent 1. in this case, agent 1 has to inform the other agent about the moves to make to get to the next node of the diagonal. This will cause a penalty that varies between 2 to 4 moves for each node computed node on the diagonal performed by the agent 2 or by both of them. In the worst case, each agent will be doing the whole computations by itself.

- **Importance of having common orientation:** The existence of common orientation is very important factor only to improve the complexity of the solution. In fact the problem is solvable even without orientation. Moreover, having common orientation in the mesh, allow the agents to gather even without communicating, which is not the case for example in the ring.

- **Effect of deploying more than one agent:** When the agents have common orientation and they are able to break their symmetry then they are able to share the workload and cooperate. Even though they can reduce the number of time units to the half of the number of moves consumed by one agent, they are not able to ameliorate the number of moves.

- **Symmetry breaking using boards:** If the agents are using whiteboards and they are in the same home base, then they will break the symmetry based on the first access to the board but they will perform extra moves to get together (let say in the most north corner) and exchange the information (i and j) then compute n and terminate.

- **Tokens:** Whiteboards were specified as the communication method for Algorithm 2, but the solution can be generalized to suit the other communication methods with some extra assumptions. Such as, for the token models, the agent has to be able to generate as multiple different tokens as it needs to mark the nodes that are marked, using the whiteboards, in the algorithm described in this section. It would be interesting to determine the minimum number of tokens needed for the tokens model in order to find the size of a mesh. In the blackboard models, the nodes need to be labelled or must have enough memory space so the agent can mark them and distinguish them later. For face-to-face models, the memory space of the nodes is still needed and the agents need to have enough memory to storage the partial map to relocate the marked nodes. In addition, we did not require distinct identifiers since a single agent was deployed.

- **Impact of the topology characteristics:** To solve the problem of finding the number of nodes in the mesh, we took advantage of the topology asymmetry. The degrees of the nodes differ depending on their positions on the grid. First the agent computed the diagonal to get to a node of type border and then walked in special fashion to get to a node of type corner and then counted the nodes on the width and the length of the mesh. However, the solvability of the problem does not depend on the asymmetry of the topology, since in the next chapter, we will describe a similar algorithm solving the same problem but in a symmetric topology: the torus.

Chapter 5

Finding the size of a Torus

The torus is a combination of the two topologies used in the two previous chapters: the ring and the mesh. It can be derived directly from the mesh topology. Only extra $(i + j)$ "wraparound" connections are added to the mesh to be converted into a torus, the nodes at the top of the mesh are linked to the nodes at the bottom of the mesh and the nodes at the left side of the mesh are linked to the nodes at its right side. Figure 6.1 represents the torus topology, visualizing the extra side-to-side links in a rectangular mesh. Unlike the mesh, the torus is a symmetric

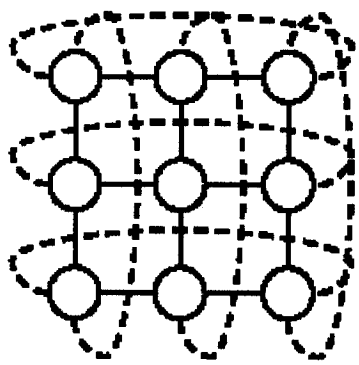

Figure 5.1: Relation between torus and mesh topologies

topology. By adding these extra links, all the nodes have the same degree (4 incident edges) and from each of them, the agents have the same view. Therefore, like in the case of the ring, we cannot use the topology to break the symmetry of the model. In order to compute the size of a

torus, we have to compute the number of nodes on its width and length, visualizing it as a mesh. Taking into consideration the extra edges, then the length and the width of the torus are actually two perpendicular rings. Therefore the problem of finding the size of a torus is equivalent of the problem of delimiting these two rings and counting the number of nodes of both of them.
When the deployed mobile agents know that they are in a torus and they do have common orientation then the solution of the problem of finding the size of the network is similar to what we already presented in Chapter 3, for finding the size of a ring. Only models without orientation are considered in this chapter.

5.1 Torus 1: Single agent without orientation

1. Regardless of the identifier of the agent (distinct or anonymous).
2. Communication method: Whiteboards .
3. Agent does not have common orientation.

Parameters	Torus - Model 1
Communication Methods	Whiteboards
Identifiers	Anonymous*
Orientation	No

Table 5.1: Parameters of model 1 - Torus.

In this model, there is no need for distinct identifiers since a single agent is deployed. Thus, algorithm 1 is used regardless the availability of distinct identifiers.

In case of lack of common orientation, a trivial solution to find the size of a torus is as follow: the mobile agent has to touch all the nodes and traverse all the edges. The complexity of this solution is $O(n)$. The solution described by Algorithm 1, that we present below, works in $O(\sqrt{n})$.

Although there is no common orientation, the agent will be able to move in a consistent direction.

The agent will initially choose an arbitrary direction, and will then continue to move toward that same direction, counting the nodes, until it comes back to its home base. The target constructed by the mobile agent is a ring and its home base is considered as the first node of it. To let the agent move on a ring, we use the same *expand* function (8 moves forming a flower) as in the algorithm deployed to find the size of the mesh, but we choose the intersecting node in a different fashion.

The Algorithm, has only one phase, the agent walks and counts in parallel. When the torus is not square but rectangular, the agent will have to do the same computations twice, once to compute and count the nodes in the ring defined by the button-top links and second for the ring defined by the side-to-side links.

Starting from its home base, the agent picks an adjacent arbitrary node and move on it. This node becomes part of the ring and marks off the direction of the agent. the orientation is irrelevant. From there, in order to find the subsequent nodes of the ring, the agent performs a full expansion visiting all the nodes that are at distance 2. The next edge to be included in the ring is the only one adjacent to the last node added to the ring (the home base in the round 0) and that does not have any intersection with the other edges visited during this stage.

Let X_1, X_2, X_3, ..., X_i be the first i nodes included in the ring.

In general, after including a node to the ring, let us refer to it as the node X_i, an expansion is performed from X_{i-1} (in the 3 directions not leading to X_{i-2}) and the only edge adjacent to X_i and that does not belong to the intersection between edges visited during this stage, is selected as X_{i+1}

Definition of Algorithm 1:

```
main()
{
  /*Declaration of the variables*/
  int number_node = 0, round_number = 0;/* Size of the Torus*/    node ring_end = home_base;
```

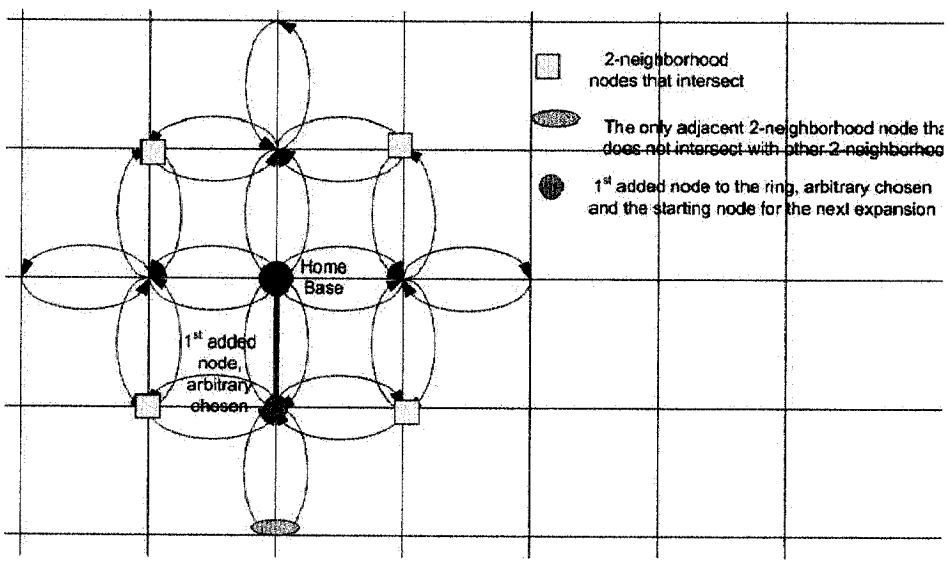

Figure 5.2: Finding the nodes of the ring

/*Starting node*/

RingConstruction(ring_end, round_number, number_node);

number_node = *number_node*^{2}; /*We consider square Torus*/ }

Definition of RingConstruction Function:

/* The RingConstruction function get as an input two parameters: the variable ring_end is the last node of the ring,

round_number is the round number.

At the beginning, the variable ring_end is equivalent to the agent's home base and the variable round_number is 0 */

RingConstruction(ring_end, round_number, number_node)

{

/* Variables */

Node nodeA, nodeB, nodetemp;

```
Edge edgeA, edgeB, edgetemp;
int round_number=0, number_node = 0;
Repeat until (ring_end = home_base and round_number ≠ 0)    {
  number_node ++;
  if(ring_end == home_base)       then       {
    /* The first node is chosen arbitrarily */
    nodetemp = ring_end;
    edgetemp = none;
    Follow any incident edge edgeA to ring_end to reach any 1-neighborhood node nodeA;
    ring_end = nodeA;
  }
    /* next instructions expand nodetemp and choose next of the ring*/
    For (Each incident edge of nodetemp except edgetemp)
    {
      Follow the incident edge of nodetemp to reach the 1-neighborhood node nodeB;
      For (Each incident edge of nodeB)
      Follow the edge to reach and mark with round_number the 1-neighborhood of the node
nodeB;
    }
    choose the adjacent edge edgeB to nodeA that ends at nodeB and does not belong to the
intersection between edges visited at this round;
    nodetemp = ring_end;
    ring_end= nodeB;
    edgetemp = edgeA;
    edgeA = edgeB;
    round_number ++;
  }
}
```

Correctness and Complexity of Algorithm 1:

Theorem 13. *Using Algorithm 1, the agent walks toward a straight direction.*

PROOF We proof by induction that following Algorithm 1, the is going straight.

- The first node can be chosen arbitrary (Determine North).

- Let us assume that the agent chooses the nodes to move to correctly up to $X_{i-1}{}^{th}$ expansion (i.e. up to choice i is correct).

- We have to prove that the next choice is correct (i.e. X_i is the right node to move to).

To reach the node X_i, by the definition of the Algorithm, we expand from the $i-1^{th}$ node in the column.
By the construction of the torus topology, each node, has eight 2-neighborhood nodes. four of these 2-neighborhood nodes can be reach following two different paths (i.e. intersection between edges). Out of the other four 2-neighborhood nodes, there is only one node at distance 2 from X_{i-1} This node is X_i and is the next node in the column.
Thus, the agent makes the correct choice of the next node to go in straight direction. ∎

Theorem 14. *In order to find the size of the torus, in the best case, the agent performs* $8+\sqrt{n}*23$ *moves and consumes* $8+\sqrt{n}*23$ *time units, while in the worst case, the agent performs* $35+i*25+j*25$ *moves and consumes* $35+i*25+j*25$ *time units.*

PROOF To find the size of the torus, at round 0, the agent picks an arbitrary node to move to. To do so, it makes 2 moves (back and forth). Then it expands its home base (visiting the 2-neighborhood). Four incident edges requires 4 expansions, each expansion costs 8 moves, that

makes a partial total of 32 moves. the total cost of adding the first node to the ring is 34 moves. For the subsequent nodes of the ring, the agent has to expand its three 1-neighborhood. If during these expansions, the agent finds the right node to add to the ring, it cannot interrupt the expansions and it has to finish visiting all its 2-neighborhood to get enough information to compute the next node of the ring in the next round. That results in performing $3 * 8 = 24$ moves, in addition to one more move to get back to the right node, which makes a total of 25 moves per new added node to the ring in the worst worst case. In the best case, the right node to add to the ring happens to be the last expanded node, then it takes only 7 moves to explore all its 1-neighborhood, no need for the the eighth moves to go back to the initial node neither for for the extra move to be back to the right node to add the to the ring. Which makes $8 + 8 + 7 = 23$ moves only

Let assume that the length of the torus is i while its width is j. In the best case $i = j = \sqrt{n}$ and the total number of moves to find the size of the torus is $34 + (\sqrt{n} - 1) * 23 = 8 + \sqrt{n} * 23$. In the worst case, the mesh is not square and the agent needs to compute and count the nodes on two perpendicular rings. The first ring will costs $34 + i * 25$ moves. While we reduce the moves of the first round of the second ring to a single one, since the the nodes are already marked in the first round of the first ring. The total number of moves for the second ring is $1 + j * 25$. That makes an overall total of $34 + i * 25 + 1 + j * 25 = 35 + i * 25 + j * 25$ in the worst case. ∎

∎

	Worst Case	Best Case
Number of moves	$35 + i * 25 + j * 25$	$8 + \sqrt{n} * 23$
Time	$35 + i * 25 + j * 25$	$8 + \sqrt{n} * 23$

Table 5.2: Finding the size of a torus with a single agent having no common orientation.

5.2 Remarks and Observations

Adaptation of the solution for the other methods of communication: Whiteboards were specified as the communication method for Algorithm 1, but the solution can be generalized to suit the other communication methods with some extra assumptions. Such as, for the token models, the agent has to be able to generate as multiple different tokens as it needs to mark the nodes that are marked, using the whiteboards, in the algorithm described in this section. It would be interesting to determine the minimum number of tokens needed for the tokens model in order to find the size of a torus. In the blackboard models, the nodes need to be labelled or must have enough memory space so the agent can mark them and distinguish them later. For face-to-face models, the memory space of the nodes is still needed and the agents need to have enough memory to storage the partial map to relocate the marked nodes. In addition, we did not require distinct identifiers since a single agent was deployed.

Thus, Algorithm 1 is used regardless the identifiers and it can be adapted to any specific available method of communication.

Deploying more agents:

If more than one agent are available, we could take advantage of them by using techniques similar to the ones described for the mesh topology. When the network is oriented, the work could be split among the agents. Thus, for square torus, the optimal number of agents to deploy in the same home base is two. However, when the torus is rectangle, four agents can be deployed in the same home base. Each pair will count the number of moves on each ring. In both cases, the agents have to break their symmetry and go in opposite directions. They will make the same number of moves as a single agent but consume less time units.

Chapter 6

Conclusion

In this thesis, we have identified the parameters that impact the distributed models of the mobile agents. Then we have used these parameters to systematically classify the models used in the literature and to cover all the different possibilities when we intend to solve a particular problem. We have defined some of the classical problems (i.e. rendezvous, exploration, leader election, and intruder capture, etc.) and investigated their relationships. We have studied the problem of finding the size of a network, under different assumptions. We have presented some solutions for this problem in the ring, the mesh and the torus. The following tables 6.1, 6.2, and 6.3 summarize the results of the proposed algorithms.

Particularly interesting has been the network size problem in a mesh or a torus topology by

Parameters	Model 1.1	Model 1.2	Model 2	Model 3	Model 4	Model 5	Model 6
Communication	Tokens, Ftf	BB or WB	BB	WB	Tokens	WB	WB
Identifiers	Dist.	Anony.*	Anony.*	Anony.*	Anony.*	Dist.	Anony.
Orientation	No*	No*	No*	No*	No*	No*	No*
Moves in wc	$n+1$	$n+1$	$2n-1$	$2n+1$	$2n$	$3n-1$	$(7n-1)/2$
Moves in bc	n	n	$n+1$	$n+1$	$2n$	$n+1$	$n+1$
Time in wc	$n/2+1$	$n/2+1$	$3n/2-1$	$n+1$	n	$5n/2-2$	$(7n-1)/4$
Time in bc	$n/2$	$n/2$	$n/2+1$	n	n	$n/2+1$	$n/2+1$

Table 6.1: Results for finding the size of a ring using two agents.

Parameters	Model 1	Model 2	Model 3	Model 4
Communication	FtF*	WB	BB	FtF*
Identifiers	Anony.*	Anony.*	Anony.*	Anony.*
Orientation	Yes	No	Yes	Yes
Moves in worst case	$2i + 2j - 2$	$17\sqrt{n} - 29$	$2i + 2j - 2$	$4i + 4j - 5$
Moves in best case	$i + j$	i	$i + j$	$2i + 2j$
Time units in wc	$2i + 2j - 2$	$17\sqrt{n} - 29$	$max(2i - 1, 2j - 1)$	$i + j + 1 + max(2i + 2j)$
Time units in bc	$i + j$	i	$max(i, j)$	$max(2i, 2j)$

Table 6.2: Results for finding the size of a mesh.

Parameters	Model 1
Communication Methods	Whiteboards
Identifiers	Anonymous*
Orientation	No
Moves in worst case	$35 + i * 25 + j * 25$
Moves in best case	$8 + \sqrt{n} * 23$
Time units in wc	$35 + i * 25 + j * 25$
Time units in bc	$8 + \sqrt{n} * 23$

Table 6.3: Finding the size of a torus.

mobile agents that did not have common orientation. In this situation, we have made the agents travelling in a consistent direction without going through a global orientation construction. The complexity of the solutions is $O(\sqrt{n})$ instead of the trivial bound of $O(n)$, where n is the actual number of nodes in the network. This walking technique could be used to solve other problems, like for example: exploration or search, etc.. When the agents know the class of the topology, then the problem of finding its size is equivalent to the problem of its map construction. When the agents have a full map of the network then can exploit it to solve complex problems.

By studying the problem, we have tried to discern the common traits of the distributed environment problems, when we consider the same assumptions and attempted to understand the hierarchy of the models. Common traits such as the fact of finding a solution in cooperative mobile agents involves breaking the symmetry of the model; which is not only an important factor to maximize the efficiency of the mobile agents but sometimes is the most crucial factor

to solve the problems under certain circumstances.

We have studied the different models, the three different topologies: ring, mesh and torus, and the problem of finding the size of the topology, and we have made various observations about the cooperation of the agents and the symmetry breaking possibilities. This work is a first step toward the creation of a hierarchy of the models and maybe standardization, toward a library creation that contains some common techniques to be applied to some classical problems: as the described walks in the mesh and the torus. However, we still need to study other problems to possibly be able to generalize our observations. Moreover as future work, it is interesting to adapt the single agent algorithms described for the whiteboards model to find the size of the mesh and the torus when there is no common orientation for the tokens model, and determine the exact number of tokens needed to solve the problem.

Some of our observations are the following:

Common orientation does not make any difference for the complexity of ring exploration solution, but it turns to be an important factor to improve the complexity of mesh exploration solution. Even with no common orientation, the problem of finding the size of a mesh is solvable; moreover, its availability allows the agents to gather even without communicating, which is not the case for example in the ring.

The availability of a blackboard is equivalent to a central communication control; then the availability of a blackboard seems to make the model always stronger than whiteboard models, however it is not always true. In fact, when the agents and the nodes are anonymous, the agents will not be able to mark and recognize their own home base in the blackboard model. Therefore, an agent is insufficient to find the size of the ring. A second agent is needed to cooperate at least by playing the role of a token in order for the other agent to terminate the computation.

In the mesh topology, the degrees of the nodes differ depending on their positions on the grid. This asymmetry is the key factor to solve the problem of finding the number of nodes. However, the extra edges in the torus transform it to a total symmetric topology; they do

not make the problem of finding the size of a torus unsolvable, and can be used efficiently to get back to the home base.

Finding a solution in cooperative mobile agents involves breaking the symmetry of the model. This turns out to be not only an important factor to maximize the efficiency of the mobile agents but sometimes it is the most crucial factor to solve the problems under certain circumstances. The availability of distinct identifiers for the agents allows a natural symmetry breaking of the models.

Bibliography

[1] S. Alpern, Asymmetric Rendezvous search on the Circle, *Dynamics and Control, vol. 10, pp. 33-45*, 1995.

[2] S. Alpern and V. J. Baston, Rendezvous Search on a Graph, *J. Applied Probability Trust 36, pp. 223-231*, 1999.

[3] E. J. Anderson and S. P. Fekete, Asymmetric Rendezvous on the Plane,. *Proceedings of the Fourteenth Annual Symposium on Computational Geometry, Minneapolis, Minnesota, USA, number 276, pp. 365-373* 1998.

[4] E. J.Anderson, and S. P. Fekete, Asymmetric Rendezvous on the Plane, *Proceedings of the 14th Annual ACM Symposium on Computational Geometry, pp. 365-373* 1998.

[5] L. Barrire, P. Flocchini, N. Santoro, and P. Fraigniaud, Capture of an intruder by mobile agents *Proceedings of the 14th annual ACM symposium on Parallel algorithms and architectures pp. 200 - 209*, 2002.

[6] L. Barriere, P. Flocchini, P. Fraigniaud and N. Santoro, Can we elect if we cannot compare?, *Proc. of 2003 Symposium on Parallel Algorithms and Architectures, pp. 324-332(SPAA 2003)*, 2003.

[7] L. Barriere, P. Flocchini, P. Fraigniaud and N. Santoro, Election and rendezvous in fully anonymous networks with sense of direction, *Proceedings of 9th International. Colloquium on Structural Information and Communication Complexity (SIROCCO 2003) pp. 17-32*, 2003.

[8] L. Barriere, P. Fraigniaud, N. Santoro and D. M. Thilikos, Searching is not Jumping, *The 29th Workshop on Graph Theoretic Concepts in Computer Science, pp. 34-45, WG 2003, Elspeet, the Netherlands, Springer Verlag*, 2003.

[9] M. A. Bender and A. Fernndez and D. Ron and A. Sahai and S. Vadhan, The Power of a Pebble: Exploring and Mapping Directed Graphs, *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC), pp. 269-278*,1998.

[10] M. A. Bender and D. K. Slonim, The Power of Team Exploration: Two Robots Can Learn Unlabeled Directed Graphs, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS) pp. 75-85*, 1994.

[11] W. Brenner, R. Zarnekow, and H. Wittig, Intelligent Software Agents: Foundations and Applications, *Publisher: Spring-Verlag*, 1998.

[12] M. Cielibak, P. Flocchini, G. Prencipe and N. Santoro, Solving the robots gathering problem, *In Proc. of 30th International Colloquium on Automata, Languages and Programming, pp. 1181-1196(ICALP20003)*,2003.

[13] D. D. Corkill and Susan E. Lander, Diversity in Agent Organizations, *Object Magazine, 8(4), pp. 41-47*, 1998.

[14] A. Dessmark, P. Fraigniaud and A. Pelc, Deterministic Rendezvous in Graphs, *Proc. 11th Annual European Symposium on Algorithms (ESA'2003), Budapest, Hungary, LNCS 2832* pp. 184-195, 2003.

[15] K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree exploration with little memory, *Proceedings of 13th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'2002), San Francisco, U.S.A pp. 588-597*, 2002.

[16] S. Dobrev, P. Flocchini, G. Prencipe and N. Santoro, Mobile agents searching for a black hole in an anonymous ring, *Proc. Of 15th International Symposium on DIStributed Computing, (DISC 2001), pp. 166-179*, 2001.

[17] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro, Multiple Agents Rendezvous in a Ring in Spite of a Black Hole, *Proceedings of 7th International Conference on Principles of Distributed Systems (OPODIS 2003), pp. 34-46*, 2003.

[18] S. Dobrev,P. Flocchini, R. Kralovic, G. Prencipe, P. Ruzicka and N. Santoro, Optimal search for a black hole in common interconnection networks, *Proc. of Symposium on Principles of Distributed Systems (OPODIS 2002), pp. 169-180*, 2002.

[19] P. Flocchini, B. Mans and N. Santoro, Sense of Direction in Distributed Computing, *Theoretical Computer Science, 291(1), pp. 29-53*, 2003.

[20] P. Flocchini, B. Mans and N. Santoro, Sense of Direction: definitions, properties, and classes, *Networks, 32(3), pp. 165-180*, 1998.

[21] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio, C. Sawchuk and N. Santoro, Mobile agents rendezvous in a Ring when token fail, *In Proc. of 10th Int. Symposium on Structural Information and Communication Complexity (SIROCCO 2004), pp. 161-172* , 2004.

[22] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk, Multiple Mobile Agent Rendezvous in a Ring, *Proceedings of 6th Latin American Symposium, Buenos Aires, Argentina, (LATIN 2004)vol 2976, pp.599-608* 2004.

[23] P. Fraigniaud and D. Ilcinkas, Digraphs Exploration with Little Memory, *In 21st Symposium on Theoretical Aspects of Computer Science (STACS), Montpellier, pp. 246-257*, 2004.

[24] P. Fraigniaud, A. Pelc, D. Peleg and S. Perennes, Assigning labels in unknown anonymous networks with a leader, *Distributed Computing, Vol. 14, No. 3, pp. 163-183*, 2001.

[25] P. Fraigniaud, D. Ilcinkas, E. Markou and A. Pelc, Power of communication in cooperative exploration of graphs, 2004.

[26] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg, Graph Exploration by a Finite Automaton, *Proceedings of 29th International Symposium on Mathematical Foundations of Computer Science (MFCS'2004), Pargue, Czech Republic, LNCS 13153, pp. 451-462*, 2004.

[27] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc, Collective tree exploration, *In 6th Latin American Theoretical Informatics Symposium (LATIN), Buenos Aires, pp. 141-151*, 2004.

[28] S. Franklin and A. Graesser, Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agent, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, published as Intelligent Agents III, Springer-Verlag, pp. 21-35* 1997.

[29] S. Gal, Rendezvous Search on the line, *Operations Research, 47, pp. 974-976*, 1999.

[30] J. Howard, Rendezvous Search on the Interval and Circle, *Operations Research, 47, No.4, pp. 550-558*, 1999.

[31] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Mobile agent Rendezvous in a Ring, *Proc. of 23rd IEEE International Conference on Distributed Computing Systems (ICDCS 2003), pp. 592*, 2003.

[32] B. Mans, Optimal Distributed Algorithms in Unlabelled Tori and Chordal Rings, *Journal of Parallel and Distributed Computing, 46(1), pp. 80-90*, 1997.

[33] N. Minar and K. H. Kramer and P. Maes, Cooperating Mobile Agents for Mapping Networks, *Proceedings of the First Hungarian National Conference on Agent Based Computation*, http://www.media.mit.edu/ nelson/research/routes-coopagents, 1999.

[34] P. Panaite and A. Pelc, Exploring Unknown Undirected Graphs, *Journal of Algorithms 33, pp. 281-295*, 1999.

[35] K. Rothermel and F. Mattern, Control Algorithms for Mobile Agents, *Institut fr Parallele und Verteilte Hchstleistungsrechner (IPVR) der Universitt Stuttgart*, 1999.

[36] N. Roy and G. Dudek, Collaborative Robot Exploration and Rendezvous, *Algorithms, Performance Bounds and Observations, Autnomous Robots, V.11 n.2, pp. 117-136*, 2001.

[37] N. Santoro, Design and Analysis of Distributed Algorithms, *To Appear*

[38] M. O. Sztainberg, Esther M. Arkin, Michael A. Bender, and Joseph S. B. Mitchell, Analysis of Heuristics for the Freeze-Tag Problem, *in Proceedings of the Scandinavian Workshop on Algorithms (SWAT), vol. 2368 of Springer-Verlag LNCS, 2002, pp. 270-279*

[39] X. Yu, Moti Yung, Agent Rendezvous: A Dynamic Symmetry-Breaking Problem *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, pp. 610 - 621*, 1996.