Multi-Agent Exploration of Unknown Areas

Ettore Ferranti
New College

Computing Laboratory
University of Oxford

Trinity 2009

Submitted in Partial Fulfilment of the Requirements
for the degree of Doctor of Philosophy

Ettore Ferranti
New College

D.Phil. Thesis
Trinity 2009

Multi-Agent Exploration of Unknown Areas

Abstract

This work focuses on the autonomous exploration of unknown areas by a swarm of mobile robots, referred to as *agents*. When an emergency happens within a building, it is dangerous to send human responders to search the area for hazards and victims. This motivates the need for autonomous agents that are able to coordinate with each other to explore the area as fast as possible. We investigate this problem from an algorithmic, rather than a robotics point of view, and thus abstract away from practical problems, such as obstacle detection and navigation over rough terrain. Our focus is on distributed algorithms that can cope with the following challenges: the topology of the area is typically unknown, communication between agents is intermittent and unreliable, and agents are not aware of their location in indoor environments. In order to address these challenges, we adopt the *stigmergy* approach, that is, we assume that the area is instrumented with small inexpensive sensors (called *tags*) and agents coordinate indirectly with each other by reading and updating the state of local tags.

We propose three novel distributed algorithms that allow agents to explore unknown areas by coordinating indirectly through a tag-instrumented environment. In addition, we propose two mechanisms for discovering evacuation routes from critical points in the area to emergency exits. Agents are able to combine the tasks of area exploration and evacuation route discovery in a seamless manner. We study the proposed algorithms analytically, and evaluate them empirically in a custom-built simulation environment in a variety of scenarios. We then build a

real testbed of agents and tags, and investigate practical mechanisms that allow agents to detect and localise nearby tags, and navigate toward them. Using the real testbed, we derive realistic models of detection, localisation and navigation errors, and investigate how they impact the performance of the proposed exploration algorithms. Finally, we design fault-tolerant exploration algorithms that are robust to these errors and evaluate them extensively in a simulation environment.

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3003. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

To Elisa, my muse.

Acknowledgements

γνῶθι σεαυτόν, know yourself. Only if you start looking for what is inside you, you can really start comprehending reality. Because what is reality, if not the world reflected into ourselves?

This is the spirit by which I have followed the research path that eventually led to this dissertation, and this is how I would like to thank the person that more than anyone helped me to do it: my supervisor and friend Niki Trigoni. ευχαριστώ πολύ, for everything she has done during these three years, for all the support she offered, and especially for having challenged me constantly, pushing me to become a better researcher. I hope she managed to do that, at least a bit. I really cannot say anything more, except how proud I am to have been one of her first students. Keep it up Niki, and I wish you best of luck for your life and career (in this order).

And this only reminds me (as if it was necessary), that one's personal life should always come before one's career, and, therefore, I would like to thank Elisa, without whom my life would be meaningless, and to whom this thesis is dedicated. Thank you for being, there is not really much else to say.

Special thanks to everyone that supported and helped me during my research in both London and Oxford. I would like to start with the Birkbeck crew, and, of course, with Mark Levene, who has always been there with his experience to give the most precious advices, but also thanks to George Roussos, for whom I am also honoured to have been a teaching assistant, and then to Giovanna, Sven, George and Nigel. I am also very happy to have joined the Assent group from the beginning, and to have seen it growing to its current level and move from Birkbeck to Oxford, always under Niki's guidance. Among the members of the group, I would like to thank especially Alexandre Guitton, who taught me how to do research before anyone else, and Antonios Skordylis, who always offered help, especially in marking my English mistakes. I am really glad to be friends with both of them. At UCL, I do not really know with whom to start, since every single person there has given me so much. Thanks to Steve Hailes, who despite being so awfully busy, always

found time to talk about my research, and thanks to Licia Capra (and Luca), really wonderful friends, who were always there for a chat or for advice, and finally to Wolfgang Emmerich, especially for his outstanding skills at playing table football in the UCL common room. On the Cambridge side, thanks to Cecilia Mascolo, whom I am happy to have seen promoted so many times now that I lost the count, and to whom I wish all the best in the future. On the Oxford side, thanks very much indeed to all the students, staff and academics of the Computing Laboratory, and in particular to my examiners Irina Voiculescu and Stephen Cameron, and to my colleagues in the Sensor Networks group, of which I am a proud member. Finally, thanks to my college, the New College (of St Mary), for having accepted me in the first place and having offered me its resources during my D.Phil. (especially the punts).

From a more personal point of view, thanks to my parents, Andrea and Claudia, to my grandparents, Dino, Agnese, Vittorio, Jole, to my sister, Elena, and to Riccardo, who always believed in me since the beginning, and supported me in every possible way. Thank you guys!

What is life without friends? I have been really lucky from this point of view, because I could not have met better ones here in London. I really need to start from Chris, a friend, the best man at my wedding, and also much much much more. Even if anything had gone bad during my stay in England, his friendship would have still made it worth it coming here. And thanks to Jasmin, of course, I am really glad she managed to come to London at last. And then, thanks to all the friends I made here, Adam, Afra, Aitor, Alan, Anastasia, Anders, Andrea, Andy, Bence, Beppe, Bruno, Claudio, Clovis, Costin, Cristina, Daniele (Borsaro), Daniele (Quercia), Daria, Dimitris, Endri, Enrico, Felipe (with special thanks to him and Sabine, also for the wonderful Heidelberg experience), Franco, Genaina, Ilias, James, Jidtima, Jo, Leticia, Liam, Manish, Matteo, Max, Maxamed (Mo), Michele, Mirco, Panteha, Pan Xueni, Rae, Salvo, Selma, Socrates, Sonia, Stefania, Stefano, Torsten, Valentina, Vito and Vladimir. I could really write a whole book only about them, and how great they all are, and it would be much longer than this thesis (and I am

glad about that).

Finally, I would like to thank all the anonymous reviewers of the papers I have published, who helped making them better, the people I have met at the conferences I attended (I am really glad to be part of this community) and EOARD, and especially Paul B. Losiewicz, because one needs to eat, and hence without them (and their financial support) this thesis would not have been written.

Quaerendo invenietis...

Contents

List of Figures

Chapter 1

Introduction

This thesis considers the problem of distributed exploration of an unknown area using a swarm of mobile robots. The focus is on two-dimensional indoor environments, and the main goal is to cover and sense the entire environment as quickly as possible.

The main class of applications that motivates this work includes emergency scenarios, such as flooding, biological contamination and earthquake events. When an emergency occurs within a building, it is crucial for the first responders to acquire as much information as possible on the ongoing situation, in order to identify and contain hazards and coordinate the rescue of victims. Initially, the area is off-limits and hazardous for anyone not wearing respiratory equipment, garments or barrier materials to protect themselves from exposure to biological, chemical, and radioactive hazards. This kind of suit can be very heavy and bulky, consequently limiting the first responders' movements, and reducing their sensing capacity (touch, vision, and hearing).

A group of autonomous robots, also referred to as *agents*, could therefore be deployed in the area to explore it in search of hazards or victims, without risking the lives of human responders. Agents could also deploy a network of sensors during the exploration process, which can then be used by humans to monitor how the emergency situation evolves over time (e.g. moving fires, collapsing corridors). Finally, agents could discover paths from points of interest in the area (e.g. places where victims or hazards are located) to exit points. The idea is to provide victims with visual signals pointing to the nearest exit, or to guide human responders from their location to a particular victim or hazard.

In pursuing these tasks, agents are typically faced with a number of limitations, such

as the possible lack of the area's map (the environment could in any case be substantially changed as the result of a disaster), the failure of previously established networks, and the short-range and often unreliable wireless communication in indoor environments. In addition, GPS receivers typically fail to pick up signals strong enough to be usable indoors and, thus, an agent cannot rely on knowledge of its exact location within a building.

Although the proposed work is inspired by this real scenario, this dissertation aims to address the problem of multi-agent exploration from an algorithmic rather than a robotics point of view. The focus is not on addressing technical challenges that arise in emergency scenarios, such as obstacle detection and avoidance (Foka and Trahanias, 2003), navigation over rough terrain (Palis et al., 2005) or deployment of artefacts on the ground (Kleiner et al., 2006). The underlying assumption of this work is that agents are capable of performing these tasks, and the main focus is on proposing distributed algorithms for the coordination of agents to rapidly explore unknown areas.

Several algorithms have been proposed for the exploration of an unknown area by a team of mobile agents (see Chapter 2 for a detailed description of them). Most of the existing approaches however make assumptions like flawless agent-to-agent communication, perfect localisation or the possibility of centralised planning, which are unrealistic in indoor emergency environments. In such scenarios, agent-to-agent communication is often intermittent and unreliable, and the indoor environment makes the use of GPS positioning impossible. A suitable approach to coordinating agents in such conditions is that of *stigmergy*: this is a nature-inspired approach observed in colonies of termites and other insects, which has been applied to various problems in the area of artificial intelligence. A description of how it works in nature is borrowed from Theraulaz and Bonabeau (Theraulaz and Bonabeau, 1999):

"The basic principle of stigmergy is extremely simple: Traces left and modifications made by individuals in their environment may feed back on them. The colony records its activity in part in the physical environment and uses this record to organise collective behaviour. Various forms of storage are used: gradients of pheromones, material structures (impregnated or not by chemical compounds), or spatial distribution of colony elements. Such structures materialise the dynamics of the colony's collective behaviour

and constrain the behaviour of individuals through a feedback loop."

In an emergency scenario as the one described above, agents can use stigmergy by instrumenting the environment with sensors and leaving traces of exploration activity on them. An existing algorithm that applies the stigmergy concept in the robotics exploration field is Ants (Koenig and Liu, 2001; Svennebring and Koenig, 2004). Ants agents use traces left on the environment to communicate and coordinate their exploration task (see Section 2.1.3 for a more detailed description of it). The application of the Ants algorithm in an emergency scenario is however difficult because its agents do not have a way of knowing when the exploration completes successfully, and they just explore the same area indefinitely until they run out of battery. Furthermore, the Ants algorithm is not very efficient in terms of exploration time, as explained in Chapter 2. Hence, there is a need for novel exploration algorithms that make efficient use of agent resources, and in which agents know when the exploration task is completed.

The remainder of this chapter is organised as follows: Section 1.1 presents the key assumptions about the exploration area and the capabilities of mobile agents and fixed sensors. Section 1.2 introduces the research problem of multi-agent exploration of unknown areas, Section 1.3 outlines the main contributions of the thesis, and Section 1.4 provides a list of papers where these contributions have been published. Finally, Section 1.5 summarises the content of the remaining chapters of the thesis.

1.1 Model

This section presents the main assumptions regarding the exploration environment and the capabilities of mobile agents and tags used for exploration.

Exploration area: The exploration area is assumed to be flat and rectangular, and it is divided into a grid of square cells. We refer to the eight cells around a given cell as *surrounding* cells, and to the four of them that lie in the north, east, south and west directions as *adjacent* cells. A cell can be in one of the following states:

- ***Wall***: The cell cannot be traversed by an agent because it is blocked by an obstacle. For simplicity, a cell is assumed to be blocked by an obstacle, if the obstacle over-

laps any part of the cell[1]. An agent is able to use its built-in sensors (laser, sonar, etc.) to detect whether one of the eight surrounding cells is a wall. It is assumed that the perimeter of the area is always formed by wall cells. Wall cells do not appear or disappear dynamically during the exploration process. The topology of the area is assumed to be static.

- ***Unexplored***: No agent has been in the cell yet.
- ***Explored***: The cell has been traversed at least once, but it is still possible for agents to traverse it again in the future, either to mark it as *explored* or to reach other *unexplored* cells.
- ***Visited***: The agents have already explored the cell, and they do not need to go through it again to reach other cells. Conceptually, this state is equivalent to a *wall* cell, in that no agent is allowed to traverse it.

Tag placement: Fixed inexpensive sensors, also referred to as *tags*, are placed in the middle of each non-wall cell. Tags can either be carefully pre-deployed by humans, or placed by agents when they first move to a previously unexplored cell. In the latter case, if agents are subject to odometry errors, they will be unable to place tags accurately in the middle of cells. The problem of tag placement is not addressed in the current thesis; we make the simplifying assumption that tags are accurately placed at the centres of grid cells and are not displaced thereafter.

Agent movements: Agents can only move to non-wall cells. They are initially deployed in one of the boundary cells and, in each step, they are able to move from the current cell to an adjacent non-wall cell in the north, east, south or west direction. With such moves agents can reach any non-wall cell in the area. In other words, if the area is modelled as an undirected graph where vertices are non-wall cells and edges exist between all adjacent non-wall cells, the resulting graph is assumed to be connected.

Agent-to-tag communication: Agents are aware of which tags lie in the current cell and in the eight surrounding cells. Since long-range wireless communication is

[1]This simplifying assumption must be revisited in future studies of real emergency scenarios, since it is possible that a victim is not detected because it shares the same cell as an obstacle.

unreliable in indoor emergency scenarios, agents do not communicate directly with each other. Instead, they coordinate indirectly by reading and updating the tags installed in the local and surrounding cells. In the distributed exploration algorithms considered in this thesis, agents make independent decisions about how to navigate through the terrain based on local state.

Tag-to-tag communication: The final assumption concerns the ability of tags to communicate with each other. During tag placement, a tag becomes aware of which tags lie in the surrounding cells and their relative positions. A tag is typically able to communicate wirelessly with the tags placed in the surrounding cells. Communication is not perfect, hence some of the links between adjacent tags may be broken in one or both directions.

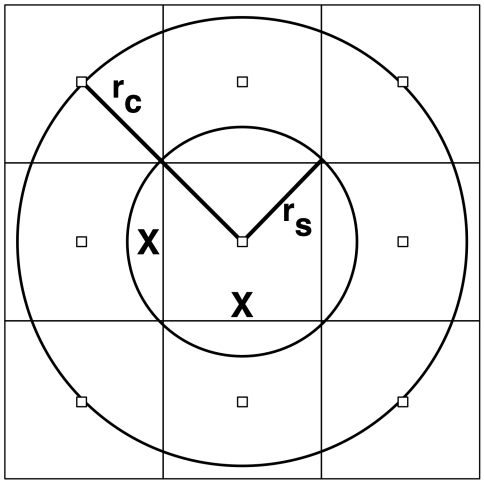

Figure 1.1: The cell size x is determined by the sensing range r_s and communication range r_c of agents. In particular, $x \leq \sqrt{2}r_s$ and $x \leq \frac{r_c}{\sqrt{2}}$.

Constraints on cell size: The cell size depends on the sensing range and communication range of agents, as illustrated in Figure 1.1. In particular, when an agent is at the centre of a cell, it must be able to scan the entire cell for victims or hazards. Therefore, if the sensing range of the agent is r_s, the cell size must be no larger than $\sqrt{2}r_s$. When an agent is at the centre of a cell, it must also be able to communicate with tags placed at the centres of surrounding cells. Similarly, a tag must be able to communicate with tags placed in surrounding cells. Therefore, if the communication range of mobile agents (and

fixed sensors) is r_c, the cell size must be no larger than $\frac{r_c}{\sqrt{2}}$.

Obstacle: An obstacle is a group of one or more *wall* cells which is not directly connected to the perimeter of the area. A more formal description will also be provided in Section 3.2.2, when describing how different algorithms negotiate loops.

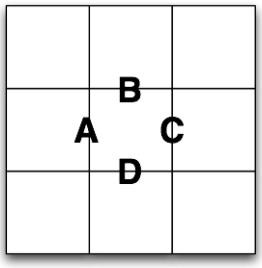

Figure 1.2: A, B, C, D are the access points from the current cell toward the adjacent ones.

Relative sizes of cells and obstacles: we assume that an agent is equipped with sensors (e.g. laser, sonar, etc.) that enable to detect if it can successfully traverse the cell area from the center of it toward the points A, B, C, D (Figure 1.2) which represent all the possible accesses to the adjacent cells. If any one of these routes is blocked by an obstacle (i.e., the size of the obstacle plus the size of the agent is larger than a dimension of the cell), then the cell is considered to be a *wall*, because at least one of its adjacent cells cannot be accessed.

Room: A room is a subset of *unexplored* cells enclosed by a perimeter of *wall* cells, interrupted by at least another *unexplored* cell (door), which connects the cells in the room to the other *unexplored* cells in the area (corridors or other rooms).

Victim: A victim is an outstanding cell in the area which is supposed to contain the victim of an emergency event. A victim is found whenever one of the agent steps into the cell, which simulate the fact that in reality an agent would have to come close to the victim to detect it using one of its sensors.

Cell occupancy: Within this model, we assume that a cell is big enough to contains more than one agent at the same time. In this way, we can focus on solving algorithmic issues instead of trying to overcome technical challenges such as obstacle avoidance (Foka and Trahanias, 2003).

The abstraction presented above is by no means the only possible one, and others could have been chosen instead. This particular solution was however adopted because several of the approaches studied in the literature (Koenig and Liu, 2001; Svennebring and Koenig, 2004; Yamauchi, 1998) were already using a similar model, and therefore could have been compared with our approach to test for possible improvements over them. Another reason is that this model is easily reproducible, and others could test our algorithms using custom built scenarios, or even compare other algorithms to our owns.

1.2 Research Problems

We are now in a position to formulate the research problems tackled in the remainder of the thesis.

1.2.1 Research Problem 1

The main research problem is to devise *distributed* algorithms that enable agents to *explore* an unknown area *as fast as possible*. This section clarifies the various parts of this high-level definition. First, the focus is on *distributed* algorithms, i.e. algorithms that do not assume global knowledge of agent and tag states. Agents can only communicate with tags deployed in the current and surrounding cells, and make navigation decisions based on this local state, as explained in Section 1.1. Second, to *explore* an area means to accomplish one of the following two objectives:

- **Exploration Objective:** This objective is achieved when all cells in the area have been traversed by an agent at least once, or are *wall*, and thus cannot be traversed at all. This means that no cell is left in the *unexplored* state. When this objective is achieved, cells are in any of the *explored*, *visited* or *wall* states.

- **Completion Objective:** This objective is achieved when two conditions are met: First, all cells in the area become *walls* or *visited*. Second, agents can ascertain that the first condition is met based on local information, i.e. by checking that local and surrounding cells are *walls* or *visited*.

The efficacy of an algorithm is assessed based on whether it is capable of achieving the *Exploration* and *Completion Objectives*. The efficiency of an algorithm is assessed based on the time that it takes agents to achieve each objective. Time is measured as the maximum number of steps (or moves) made by any agent until one of the objectives is met. In each step, an agent typically moves from the current cell to one of the four adjacent cells, although there are cases where it might decide to remain in the current cell. This metric has been chosen because it always gives the worst case scenario in which a victim is found in the last cell to be traversed, as opposite to the average time to reach a cell. This is a common solution adopted when critical parameters (such as human life) are affected, and particularly suited to emergency scenarios simulations.

By definition, the *Exploration Objective* is always achieved earlier than (or at the same time as) the *Completion Objective*. Both objectives should be achieved in the minimum amount of time, because in an emergency scenario as the one we are considering, speed is essential. The faster the *Exploration Objective* is achieved, the faster victims and hazards are identified. The faster the *Completion Objective* is achieved, the earlier human responders can enter the area with the certainty that there are no hidden hazards.

In summary, *given a number of agents and an area instrumented with tags, the main research goal is to design distributed algorithms that achieve both the* Exploration *and* Completion Objectives *in the minimum number of steps*.

1.2.2 Research Problem 2

Besides the main goal of area exploration, agents could also be tasked to discover evacuation routes that link critical points in the area, where victims or hazards are identified, to emergency exits. An evacuation route is a sequence of consecutive non-wall cells, starting at a critical cell and ending at an exit cell. To discover such a path means to mark each tag on the route with the direction (north, east, south, west) towards the next tag on the route. The second research problem is to devise distributed algorithms that discover evacuation routes, without prior knowledge of the area's map nor of the location of critical points in the area. Two optimisation goals are defined for this problem:

1. To discover evacuation routes as early as possible in the exploration process, that

is to minimise the average number of steps that agents take to discover the first evacuation routes from critical points to exit points.

2. To keep evacuation routes as short as possible in order to enable easy access of human responders to victims or hazards. That is, given a number of agent steps, the goal is to minimise the average route length from critical points to exit points. Routes that have not been discovered yet are considered to have infinite length.

1.2.3 Research Problem 3

The third research problem of this work is to investigate practical issues that arise from applying distributed exploration algorithms in a realistic environment. More specifically, the goal is to investigate the ability of an agent to accurately detect, identify, and localise tags in surrounding cells and move accurately towards one of them. The key research goals here are to:

1. To deploy a real testbed of agents and tags in a lab environment, and implement practical techniques that allow an agent to detect, identify and localise tags in the current and surrounding cells, and move towards one of them.

2. To derive realistic models of detection, localisation and navigation errors, using the real testbed.

3. To assess the impact of these errors on the performance of exploration algorithms proposed to address the first research problem.

4. To design fault-tolerant versions of the proposed exploration algorithms, which are robust to these errors.

1.3 Contributions

This section highlights the main contributions of this thesis with respect to the three research problems defined above.

1.3.1 First Contribution: Exploration Algorithms

The first contribution concerns the design of three distributed exploration algorithms, namely Multiple Depth First Search (MDFS) (Ferranti et al., 2007), Brick&Mortar (Ferranti et al., 2007) and HybridExploration (Ferranti et al., 2008). In ideal conditions (e.g. accurate tag placement, reliable agent-to-tag communication and accurate agent movement) the three proposed algorithms are shown to achieve the *Exploration* and *Completion Objectives*. Complexity results are provided together with an extensive empirical evaluation of their performance in a simulation environment. Their performance is compared with the existing Ants exploration algorithm (Svennebring and Koenig, 2004) in a variety of settings, for example, as we vary the number of agents, the number of cells, the number of rooms, and so on. Brief descriptions of MDFS, Brick&Mortar and HybridExploration are provided below.

MDFS: The idea behind this algorithm is that agents traverse the cells of an area in a depth-first-search manner. The boundary cell where an agent departs is the root of the tree. As an agent traverses a tree downwards it marks cells as *explored*, and as it traverses it upwards, it marks cells as *visited*. The process continues until the agent has marked all cells as *visited* and returns to the root cell. Local rules are defined to ensure that agents do not interfere with each other while navigating the same or different depth-first-search trees.

Brick&Mortar: The idea behind this algorithm is that agents progressively thicken the existing walls by progressively marking cells that surround walls as *visited*. As blocks of inaccessible (*wall* and *visited*) cells become thicker, corridors of accessible (*unexplored* and *explored*) cells become thinner until they finally disappear. Unlike MDFS, Brick&Mortar agents typically traverse each cell once, especially in open-space areas without many rooms and obstacles.

HybridExploration: Whereas the previous two algorithms rely purely on agent-to-tag communication, this hybrid algorithm exploits both agent-to-tag and tag-to-tag communication. It uses two types of agents: *physical agents*, which are robots navigating through the area and communicating with nearby tags, and *virtual agents*, which are

messages exchanged between tags. The idea is that virtual agents propagate state through the network of tags, which assists physical agents in making better exploration decisions.

1.3.2 Second Contribution: Evacuation Routes

The second contribution concerns the design of distributed algorithms that allow agents to identify short evacuation routes at an early stage. In particular, we propose two distributed mechanisms for evacuation route discovery, one based on agent-to-tag communication (i.e., communication between agents and stationary sensors), and one based on inter-tag communication. We show that these route discovery mechanisms can be easily integrated with exploration algorithms, like Ants (Svennebring and Koenig, 2004), MDFS and Brick&Mortar (Ferranti et al., 2007). The idea is to activate the search for evacuation routes in parallel with the task of area exploration. Furthermore, we carefully examine the impact of the exploration algorithm on the efficiency of our route discovery mechanisms. We address the following questions: i) does the exploration algorithm affect the final length of the evacuation routes? ii) does the exploration algorithm affect when evacuation routes are first discovered, and how they are improved over time? Finally, we measure the performance of our discovery mechanisms in a wide variety of settings. Our goal is to understand how the topology of the area (e.g., size, number of rooms and obstacles, number of emergency exits) affects the quality of evacuation routes. This study could help human responders predict their accessibility to victims and hazards given a rough knowledge of the area's topological features.

1.3.3 Third Contribution: Realistic Error Models and Fault Tolerant Algorithms

The third contribution concerns the investigation of practical issues arising from applying distributed exploration algorithms in a real environment. In particular, we provide a detailed description of our testbed, including the hardware and software architecture of agents and tags. We also propose practical mechanisms that allow agents to detect and localise surrounding tags, and move towards one of them. We test them in the real testbed, and derive realistic models of detection, localisation and navigation errors. We

then discuss pathological cases that illustrate how existing and proposed exploration algorithms are affected by these errors. Finally, we insert our error models into a simulation environment, and assess how the performance of the algorithms degrades as a result of introducing realistic errors. To overcome the limitations of MDFS, Brick&Mortar and HybridExploration in the presence of errors, we propose Fault-Tolerant (FT) versions of these algorithms and we test them in a simulation environment with realistic errors. The proposed FT algorithms are shown to achieve the *Exploration Objective* in fewer steps than the existing Ants algorithm. However, due to errors, they are no longer able to achieve the *Completion Objective*.

1.4 Publications

The main contributions of this thesis have been published in the papers listed below. All papers are co-authored by my supervisor, Dr Niki Trigoni, and several of them, by Professor Mark Levene from Birkbeck College, University of London. Since I provided the majority of ideas in these papers, implemented algorithms and derived experimental results, I have been selected as the first author. I also contributed to writing initial drafts of these papers, which were further improved by Dr Trigoni and Professor Levene.

- E. Ferranti and N. Trigoni *Practical Issues in Deploying Mobile Agents to Explore a Sensor-Instrumented Environment*. The Computer Journal. Oxford University Press. February 2010.

- E. Ferranti and N. Trigoni *The impact of localization errors on the performance of the Ants exploration algorithm*. ATSN09, Third International Workshop on Agent Technology for Sensor Networks (AAMAS-09). 12 May 2009, Budapest, Hungary.

- E. Ferranti and N. Trigoni *Practical Issues in Deploying Mobile Agents to Explore a Sensor-Instrumented Environment*. Technical Report RR-09-02. University of Oxford Computing Laboratory, 2009.

- E. Ferranti, N. Trigoni and M. Levene *Rapid Exploration of Unknown Areas Through Dynamic Deployment of Mobile and Stationary Sensor Nodes*. Journal of Au-

tonomous Agents and Multi-Agent Systems. October 2009. DOI 10.1007/s10458-008-9075-4.

- E. Ferranti, N. Trigoni and M. Levene *HybridExploration: a Distributed Approach to Terrain Exploration using Mobile and Fixed Sensor Nodes*. IROS08, IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems, 22-26 September 2008, Nice, France.

- E. Ferranti and N. Trigoni *Robot-Assisted Discovery of Evacuation Routes in Emergency Scenarios*. ICRA08, IEEE International Conference on Robotics and Automation 19-23 May 2008, Pasadena, CA, USA.

- E. Ferranti, N. Trigoni and M. Levene *Brick&Mortar: An On-Line Multi-Agent Exploration Algorithm*. ICRA07, IEEE International Conference on Robotics and Automation 10-14 April 2007, Roma, Italy.

1.5 Summary

The remainder of this dissertation is organised as follows:

Chapter 2 - Related Work

This chapter presents existing research on exploration algorithms subdividing them into *off-line* and *on-line* algorithms. We pay particular attention to describing the Ants algorithm (Svennebring and Koenig, 2004) which makes the same assumptions as we do and is the most related to this work. A final overview concerning the projects involved in the Robocup Rescue Competitions is provided at the end of the chapter.

Chapter 3 - Exploration Algorithms

This chapter describes three novel exploration algorithms (MDFS, Brick&Mortar and HybridExploration), provides complexity results and intuitive justifications that they achieve the *Exploration* and *Completion Objectives*. It also proposes distributed mechanisms for discovering evacuation routes from critical points in the area to exit points. Several illustrative examples are provided to show the functionality of exploration algorithms and mechanisms for discovering evacuation routes.

Chapter 4 - Evaluation

This chapter evaluates existing and proposed exploration algorithms, as well as mechanisms for discovering evacuation routes, in a simulation environment. An extensive empirical evaluation is provided to understand the impact of several parameters, such as the number of agents, the number of cells, the number of rooms, and so on, on the algorithms performance.

Chapter 5 - Real Deployment

This chapter presents the deployment of the Ants exploration algorithm in a real testbed. Firstly, we provide a detailed description of the hardware and software architecture of a real testbed, consisting of agents and tags. We then propose practical mechanisms that allow agents to detect and localise tags in their vicinity, and move towards one of them. We evaluate these mechanisms in the real testbed, and derive realistic detection, localisation and navigation errors. We then evaluate the performance of Ants, MDFS, Brick&Mortar and HybridExploration in a simulation environment with realistic errors.

Chapter 6 - Fault Tolerant Exploration Algorithms

This chapter presents Fault-Tolerant versions of MDFS, Brick&Mortar and HybridExploration, which are robust to errors described in the previous chapter. It provides intuitive justifications that the proposed FT algorithms achieve the *Exploration Objective*. It also includes an empirical evaluation of the FT algorithms in a simulation environment with realistic errors.

Chapter 7 - Conclusions

This chapter presents the main conclusions of this work, identifies limitations, and proposes directions for future work.

Chapter 2

Related Work

This chapter provides an overview of existing exploration and localisation algorithms related to this thesis. Exploration algorithms are classified based on the assumptions they make about the knowledge of the area's map and the mode of communication between agents. One of these algorithms, Ants, is presented in more detail because it adopts the same model as the one in this thesis, which is discussed in Section 1.1. Localisation techniques are classified based on whether they use radio, infrared, sound or camera technologies. This chapter ends with a brief discussion of how related work on Robocup Rescue and Simultaneous Localisation and Mapping differs from the work proposed in this thesis.

2.1 Exploration Algorithms Overview

Information gathering inside an area is essential to avoid risking the lives of the first responders: for example, if the responders knew the locations of the victims before entering a building, they could immediately get there avoiding hazardous areas such as rooms on fire or collapsed corridors or stairs. Exploring all the area in the minimum amount of time and reporting back to the human personnel outside the building is therefore an essential part of rescue operations. A group of mobile agents should therefore be deployed in the area to acquire all the information that could assist the tasks of the first responders. The existing algorithms used by the agents to perform the exploration task can be distinguished based on two criteria: (i) knowledge of map and (ii) communication mode.

1. **Knowledge of map**: Choset (Choset, 2001) provides a survey of coverage algo-

rithms and distinguishes them into *off-line* and *on-line*. In the former, the agents are previously provided with a map of the area to explore, while in the latter, also called *sensor-based*, no assumption is made concerning the availability of an environmental map for the agents.

2. **Communication modes**: agents can communicate in one of the following ways:

   - *Agent-to-Server communication*: agent coordination occurs through a central server;
   - *Agent-to-Agent communication*: a direct wireless communication occurs between agents within range;
   - *Agent-to-Environment communication*: agents communicate indirectly by interacting with an instrumented (smart) environment.

This work focuses on online algorithms that rely on Agent-to-Environment communication. To the best of our knowledge, little work (Koenig and Liu, 2001; Svennebring and Koenig, 2004; Kleiner et al., 2006; Ferranti et al., 2007) has investigated the problem of *on-line* area exploration by letting agents coordinate indirectly by tagging the environment, subsequently reading and updating the state of the deployed tags. Moreover, the existing algorithms are not able to autonomously decide when the exploration task is successfully completed. Recognising when the exploration is complete is of primary importance in an emergency scenario where robots have to report back to the first responders the situation inside the building as soon as the entire building is explored.

A brief survey of existing *off-line* and *on-line* algorithms is provided in the two following subsections.

2.1.1 Off-line Algorithms

Since the *off-line* algorithms previously know the map of the environment, in theory they could build a set of optimal paths to cover the area using all the available agents in the minimum amount of time (optimal solution). In practice, this is not possible because this problem is NP-complete, therefore heuristics are needed to find a feasible solution in

polynomial time. Such a heuristic is proposed in (Zheng et al., 2005), where the authors prove that the original problem is NP-complete, and propose a polynomial algorithm, which runs in the worst case eight times slower than the optimal solution. Agmon et al. (Agmon et al., 2006) propose a faster tree construction algorithm, while Hazon et al. (Hazon and Kaminka, 2005; Hazon et al., 2006) focus on the robustness of the solution, so that even if only one robot remains in operation, it will be able to carry and complete the exploration task. All the *off-line* algorithms, being centralised in their computation of the exploration paths, use the Agent-to-Server communication mode.

Off-line algorithms are not likely to be deployed in real world scenarios, mainly because of the following reasons:

- to be able to provide agents with a detailed map of the area to explore, one would need a central repository with maps of all buildings, underground areas and public spaces that could be hit by a disastrous event;
- maps should all be encoded with the same protocol, in order to be read by the agents;
- agents should be able to communicate with a central server at all time during the exploration, avoiding obstacles and interferences, to report their position and the status of the exploration;
- agents should be able to localise themselves within the area at all time during the exploration, to be able to communicate their position to the central server. This last point is particularly unfeasible since the main localisation techniques (i.e., GPS, ultrasound etc...) are know to perform particularly bad (or not to perform at all) in buildings and underground.

Because of all the reasons discussed above, *on-line* algorithms are much more suited to the exploration of unknown areas, and will therefore be discussed in much more detailed in the next subsection.

2.1.2 On-line Algorithms

Agents running *on-line* algorithms rely only on their sensors to navigate an unknown environment, and take *on-line* decisions about what to do in each step. Having many different possible environments, one algorithm could work better in an indoor environment with tiny rooms and a great number of corridors, while another could be faster in the case of big rooms interconnected by many doors, and so on. Most of these approaches divide the environment into cells, also called regions, which are explored one by one iteratively until the entire area is covered. Furthermore, on-line algorithms are comparable to our approach with regard to hardware cost and complexity, since they all make use of small inexpensive robots with few sensors, and try to minimise their exploration time (Wagner et al., 2008).

A very important contribution can be found in the work of Wagner et al. (Wagner et al., 2008; Yanovski et al., 2003; Wagner and Bruckstein, 2001; Wagner et al., 2000), who use small inexpensive robots (A(ge)nts) to explore an unknown area. They formalised the exploration problem (a one-robot formalisation is also available by Albers et al. (Albers et al., 1999)), proved that the off-line coverage problem is NP-Hard, and proposed very interesting heuristics to solve the on-line version. Furthermore, in their recent work (Wagner et al., 2008) they envisioned an approach (CLEAN) presenting several similarities with the Brick&Mortar, one of the exploration algorithms proposed in this thesis[1]. In particular, robots make use of the status of 'dirtiness' of floor tiles to determine if someone already explored a certain area, and clean a tile to "mark" it as already visited. However, this method is incapable of solving loops when scenarios with obstacles are considered; to address this problem, the authors suggest a loop resolution mechanism (using additional markings) that is similar to the one used by Brick&Mortar (Ferranti et al., 2007). Although the algorithm works well in theory, it could be questioned how an agent can always detect whether a particular cell has already been traversed by itself or by another agent. In particular, the assumption that all the cells are dirty at the beginning of the exploration (or cleaning process) is perceived as particularly strong (what if parts of

[1]Brick&Mortar was however presented earlier in (Ferranti et al., 2007).

the area are not as dirty as the others?). Furthermore, the approach is obviously not well suited for the exploration of emergency scenario, when an instrumented environment will be needed to provide support to the exploring agents and to the victims.

Batalin and Sukhatme proposed the use of radio beacons to guide the navigation of robots and assist them in the coverage of an unknown terrain (Batalin and Sukhatme, 2005; Batalin and Sukhatme, 2003b; Batalin et al., 2004; Batalin and Sukhatme, 2003a). In particular, robots are able to detect beacons (which are pre-deployed in the environment), choose one of them and move toward it. Beacons are able to tell a robot in which direction (North, West, South or East) the least recently visited neighbour beacon lies. Beacons and robots are both equipped with a 2-bit compass, so the former can give the latter indications about which direction to take to reach the next beacon. Furthermore, in (Batalin and Sukhatme, 2007), the authors suggest that beacons could be dynamically deployed by a robot, which could then use the same Least Recently Visited (LRV) navigation approach. The precision of radio localisation however (>3m, see Section 2.2) is not enough to effectively localise victims in an emergency scenario, and the efficiency of the used algorithm is very low when compared to other solutions (the algorithm is very similar to Ants (Koenig and Liu, 2001; Svennebring and Koenig, 2004), discussed in more details in Section 2.1.3.

The use of a pre-deployed embedded network to assist the navigation of a robot in the environment has been extensively studied by O'Hara et al. (O'Hara and Balch, 2004a; O'Hara and Balch, 2004b; O'Hara et al., 2005; O'Hara et al., 2006). In particular, the authors use small and relatively inexpensive embedded network platforms, the GNATs, to guide the navigation of a LEGO Mindstorm/RCX robot using infrared transmitters and receivers. The main strength of this work relies on the extensive real-world experiments that the authors have done (up to 156 nodes deployed), which proves the feasibility of their approach and of the Agent-to-Environment communication mode in general. However, the agents heavily rely on radio signal range to detect their position with respects to the surrounding GNATs, and to navigate toward them. This is unlikely to work in a real environment, where interferences and long range communication issues will make the approach unfeasible. Furthermore, agents in the proposed approach are not really exploring

the area, but are routed by the existing infrastructure in order to reach a particular goal.

Finally, Li et al. (Li et al., 2003; Li and Rus, 2005) also use a sensor network to help a user (human or robot) safely traverse a hostile environment, but the positions of the nodes are always known (by using a GPS on each of them) and in the real experiment there is no communication between the user and the network (information from the LEDs of the nodes are video recorded and examined a posteriori).

A different approach (following an Agent-to-Server communication mode) has been presented in (Kong et al., 2006). In this work, the authors use a Boustrophedon (Choset and Pignon, 1997) technique to let the robots cover the area. The environment is divided into cells and each robot starts covering each cell with forward and reverse phases. While doing this, it builds a graph of the environment which is shared by all the robots of the team. Due to this fact, the robots always know where there are uncovered cells and therefore they can distribute themselves over the area in order to cover the remaining unexplored regions. The proposed algorithm is simple and easy to implement on real hardware. It also yields low coverage time because the robots always know where the unexplored zones are, so they can get there directly without wasting time in idle states. The graph is shared among all the agents so that, if one of them has a fault, the covering procedure can continue without losing any information. However the assumptions are different from the ones we are adopting in the present work, in particular our agents do not rely on perfect wireless communication among them and we do not assume that they always know where they are in the map (perfect localisation). It is easy to understand how these assumptions would be too strong to let the approach be used in a real world scenario, where perfect localisation and communication would not be possible.

Yamauchi presents a frontier-based exploration algorithm (Yamauchi, 1998), where the agents explore the environment, represented by a regular grid of cells, keeping in their memory a map of the area and always directing themselves “to the boundary between open space and uncharted territory". A depth first search algorithm is used to move from the current position to the next frontier. Each agent has a local map and a global map shared with all the other agents. When a local map is updated (the agent explores a new area), it is summed with the global map, and the latter is broadcasted to all the other agents

so that they can update their global maps. The agents do not broadcast information about which area will be explored next, so different agents could explore the same frontiers in the same time resulting in an inefficient utilisation of the team. The algorithm also makes the same strong assumptions that we found in (Kong et al., 2006) namely perfect localisation, communication and mapping, and uses an Agent-to-Server communication approach, because the map is stored in a centralised server.

Another Agent-to-Server communication approach is presented by Howard et al. (Howard et al., 2006), where the authors propose a general approach to explore a building, find objectives, and report them back to the human personnel outside. However, it requires human support to solve problems like loop closures or map merging between agents, so it does not satisfy the requirements for autonomous area coverage.

Finally, works which use the Agent-to-Agent communication paradigm are discussed below.

Burgard et al. (Burgard et al., 2000) do not assume that the environment is divided into grid cells. Agents compute a utility function to go to the next “frontier" in order to maximise the explored territory. Although the agents have to store information about the map and localise themselves, this approach is probability-based and therefore more suitable for a real scenario: the agents have a list of target points to reach in the next step, and each of them is associated with a value which takes into account the cost to reach the point, and the probability of exploring new areas once an agent has positioned itself on that point. The probability takes into account how many agents are going to explore the area in which the target point is, therefore avoiding the situation where all the agents explore the same area. Rekleitis et al. (Rekleitis et al., 1997) try to improve the exploration by mapping the environment while the area is covered by two robots. To localise the robots they use odometry (a position estimate based on the previous movements), but while a robot is exploring the area, the other stands still and observes the former to measure its movements and improve the localisation; after a certain amount of time the two robots exchange roles. The authors map the environment as trapezoids divided into stripes which are connected forming a graph. The agents use a depth first search algorithm to cover all the stripes. Both approaches however suffer from the strong assumptions of

perfect localisation of the robot within the area, and perfect communication in order to coordinate the exploration process.

Another interesting approach, inspired by ants behaviour, is proposed by Payton et al. (Payton et al., 2001). The authors use a swarm of robots which spread into the environment, and coordinate between themselves through infrared communication. Messages are propagated through the robots (which act both as explorers and as relays for messages) to send information about the distance and position of a goal that must be reached. The goal is specified at the beginning and known to the robots, which thus are not properly exploring the area but just trying to reach a particular, already know, point within it.

Finally, Batalin et al. (Batalin and Sukhatme, 2002) focus on agent dispersion and propose two algorithms to make the agents move away from each other when they are in sensing range. This is an important issue because the agents should always be spread out as much as possible in the environment to avoid wasting resources in exploring the same area multiple times. The approach however is not a complete solution to the area exploration problem.

2.1.3 The Ants Algorithm

In this section, we discuss the behaviour, strengths and limitations of Ants, an existing online algorithm that relies on Agent-to-Environment communication (Koenig and Liu, 2001; Svennebring and Koenig, 2004). This is a distributed algorithm that simulates a colony of ants leaving pheromone traces as they move in their environment. The trace left at a cell corresponds to a counter denoting how many times the cell has been traversed by agents. Initially, all cells are marked with value 0 to denote that they are *unexplored*. In each step, an agent reads the values of the four cells around it and chooses to step onto the least traversed cell (the one with the minimum value). Before moving there, it updates the value of the current cell, for example by incrementing its value by one. The authors discuss a few other rules that could be used instead to mark a cell and navigate to the next one. Using either of the proposed rules, agents are guaranteed to eventually explore the entire area, and thus achieve the *Exploration Objective*.

The first advantage of the algorithm is its simplicity: agents do not require memory

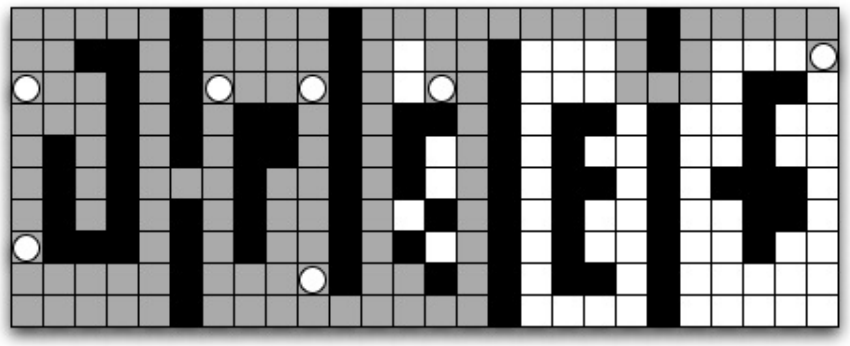

Figure 2.1: The Ants algorithm is not efficient in a scenario with many rooms, because most of the agents explore the first rooms repeatedly, while only few of them set out to discover new areas.

or radio communication, but only one-cell lookahead. Since they are easy to build, many of them can be used to speed up the exploration process. Secondly, there is no map stored inside the agents: if one of them is relocated (accidentally or on purpose) it will not even realise it and it will continue to do its work as if nothing happened. This means that the whole system is flexible and fault tolerant, and the area can be covered even if some markings or agents are lost. At the storage device of each cell, we only need to store an integer counting the number of times that agents have traversed the cell. When the number of times exceeds a threshold, the counter is reset to 0.

The main limitation of the Ants algorithm is that the *visited* state is not used during the process of marking cells. Therefore, the *Completion Objective* is never achieved, and the agents continue the exploration phase until they run out of energy.

Thus, this approach is not suitable in an emergency scenario, in which the primary consideration is to cover the overall area as soon as possible, and be notified immediately after the task is completed. A further drawback of the algorithm is the limited collaboration among agents. As shown in Figure 2.1, in a scenario with many rooms most of the agents would sweep the first few rooms repeatedly, while only a few of them would venture to explore new areas, thus limiting the efficiency of the algorithm when using multiple agents.

2.1.4 Recent Approaches using Ants

Recently, algorithms based on Ants have been applied with success in many different research areas, ranging from TSP problems to image processing and data mining (Mullen et al., 2009). Their application to the exploration of unknown areas in emergency scenarios has however been hindered by the problems stated above (most notably the impossibility of knowing when to stop, and the slow convergence to a common objective). Most of the recent algorithms in the literature in fact, as well as the ones proposed in this work, have tried to solve these problems by using other, more efficient, approaches, although keeping the original stigmergy ideas proposed by the original Ants algorithm (Koenig and Liu, 2001; Svennebring and Koenig, 2004).

Among the few approaches which used techniques closely related to the original Ants algorithm, Dasgupta and Cheng (Dasgupta and Cheng, 2009) tried to port the system in a real world test-bed using a commercial robot called *e-puck*. Although keeping the original structure with cells in a grid-like fashion, and the same counting algorithm to choose which next cell to visit, the authors tried to introduce more variables related to the real sensors with which the robot was equipped. Moreover, some form of direct communication between the robots was allowed, in order to boost collaboration. However, the relative gain with respect to the original algorithm is negligible, and the only real improvement can be observed with respect to the random algorithm, in which all the robots are performing random walk. Regrettably, the authors were not able to exploit the camera with which every robot was equipped, and had therefore to fall back on localising the robots with a camera mounted over the area to explore, an approach which is obviously unfeasible in a emergency scenario.

Although the effectiveness of Ants algorithms with regard to the exploration of unknown areas is somehow limited, the same algorithm can achieve very good performance when applied to patrolling problems, in which a team of robot must go over the same area over and over again without stopping. Glad et al. (Glad et al., 2010; Glad et al., 2008) propose an approach called EVAP, in which they present very interesting results using real *Kephera 3* robots. In their work, they are in fact able to converge to very stable cycling

paths, that the robots use to patrol an area. Another interesting point, regarding the scarce applicability of approaches totally based on theoretical models on real environments, is also raised in their work. This is very similar to our own discussion, that will be presented in Chapter 5.

Finally, Shiloni et al. (Shiloni et al., 2009) have recently provided very interesting results concerning the study of the computational power of two classes of algorithms, namely Ants and Elephants. Their Ants robots have the same properties of the original Ants algorithm which we presented above, while their Elephant robots have more computational and communication capabilities, and can therefore use more sophisticated techniques to explore an unknown area. In their work, the authors show how Ants robots can always achieve the same objectives as Elephant robots, albeit occasionally the first require many more resources (i.e., number of robots or time) to converge to the same results than the latter would need.

2.2 Overview of Localisation Techniques

Online algorithms that use Agent-to-Environment communication require agents to be able to locate tags lying in their vicinity and move accurately toward one of them. In this section we overview techniques that could be used for this purpose. These are based on radio signal strength, infrared, sound, or camera technologies.

- Radio Signals:

  Radio signal strength is not a reliable way of identifying the robot relative position with respect to tags deployed in an environment. In fact it heavily depends on factors like the relative orientation of the deployed motes, their height from the floor, the material of the floor, and the presence of obstacles in the environment. Moreover, in the literature (e.g. (Kotz et al., 2004), (Aguayo et al., 2004), (Park et al., 2003), (Kotz et al., 2003), (Zhao and Govindan, 2003), (Cerpa et al., 2003), and (Ganesan et al., 2002)) it is widely accepted that (i) radio propagation is non-isotropic (i.e. the received signal, at a given distance from the sender, is not the same in all directions), (ii) it has non-monotonic distance decay (i.e. lower distance does

not mean better link quality), and (iii) the communication is based on asymmetrical links (i.e. if A hears B, it cannot be assumed that B hears A).

However, several previous studies used radio frequency to cope with localisation issues. For example, to solve the problem of determining if the robot is in the neighbourhood of a sensor, Batalin et al. (Batalin et al., 2004) create an algorithm called Adaptive Delta Percent, which takes into account the signal strength of the messages received from the various tags while the robot is moving in order to guide it toward one of them. A strong limitation of this approach is that the authors consider an experiment to be successful if the robot is able to reach a tag in the environment within a distance of 3m, and accuracy that is unreasonable for our scenario. In their work, Bhattacharya et al. (Bhattacharya et al., 2005) assume that pre-deployed nodes are location-aware and both robot and nodes have a limited radio communication range, so that a robot can establish both if it is close to a certain node and if it is able to communicate with it. The authors do not provide results regarding the effectiveness of this method, and the approach is not suitable if communication ranges from different nodes overlap.

- Infrared Signals:

  Several systems have been created to define mobile robot localisation in indoor environments. Some of them use ultrasonic and infrared technologies simultaneously (Ghidary et al., 1999), others radio frequency (RF) and infrared together (Kelly and Martinoli, 2004), and some just infrared techniques (Kirchner and Furukawa, 2005). However, infrared signals are not completely suitable for our scenario because they have a particularly limited transmission range (i.e. $\sim$20-30cm), thus the robot risks not being able to identify the deployed tag if the dimension of the cell is bigger than the allowed range. Moreover, interference from the IR component of other light sources could compromise the localisation process (Kataoka and Atagi, 1997).

- Ultrasound and Sound Signals:

  Ultrasonic sensors (Lim and Leonard, 2000) alone could be used to avoid obstacles,

but not to identify specific tags in the environment due to the poor resolution of their readings. Therefore we argue that ultrasound is not a suitable technology for localising tags surrounding an agent or guiding the agent toward one of them.

In (Horchler et al., 2004), the authors propose a bio-inspired approach (from cricket mate calls) where sound beacons are generated by a loudspeaker that is connected to a laptop, and a robot detects and navigates towards these beacons through a rough terrain. Although the system has been proved to work in outdoor areas, and the robot is quite effective in moving over various terrains, we argue that the precision offered by the system ($\sim$1m) is not high enough to be applied to our scenario.

- Camera Technologies:

  Since the previous approaches are not suitable for our scenario, we decided to explore agent localisation using camera technologies. Several approaches investigated this area adopting feature cluster recognition (Djekoune and Achour, 2000; Castle et al., 2007). In particular, some of them use image processing techniques to recognise landmarks in the environment (Davison et al., 2004; Rice, 2007). However, most of the approaches present very sophisticated techniques to recognise specific features and landmarks within the environment to autonomously guide the agent in the exploration (e.g. in SLAM). On the contrary, we plan to use cameras in a simpler way.

  In particular, cameras allow us not only to extract the infrared component of a LED light source, but also to identify the position of the LED within a picture. Moreover, by processing all pictures that the agent progressively takes as it moves through the environment and extracting their differences, it is possible to keep the LED light in the centre of the image so that *odometry errors* can be corrected if the relative position of the LED light within the image changes. In this way, the agent can be driven by more accurate movements and it is able to correct its position autonomously. The camera range is limited only by its resolution, thus we do not have the IR short-range constraints. Practical mechanisms that use camera technologies to localise tags and navigate to selected ones are described in detail in Section 5.1.1.

2.3 The Robocup Rescue Competitions

The goal of the urban search and rescue (USAR, 2008) (USAR) robot competitions is to increase awareness of the challenges involved in search and rescue applications, provide objective evaluation of robotic implementations in representative environments, and promote collaboration between researchers. It requires robots to demonstrate their capabilities in mobility, sensory perception, planning, mapping, and *practical operator interfaces*, while searching for simulated victims in unstructured environments.

We carefully checked the approaches of the teams participating to the most recent edition (2008) and we found out that, although many of the solutions proposed offer an autonomous navigation mode, this is not used during the entire exploration, since a human operator is always required to guide the robot out of situations in which it is blocked. The aim of the competition is more focused on solving mobility, perception and sensory problems while letting a human operator control the robots themselves.

2.4 Simultaneous Localisation and Mapping

In this section, we provide a brief overview of the Simultaneous Localisation and Mapping (SLAM) problem. Our aim is to describe its main characteristics and to emphasise how it is different from our problem. A detailed description of SLAM can be found in two tutorials by Hugh Durrant-Whyte and Tim Bailey (Durrant-Whyte and Bailey, 2006) and (Bailey and Durrant-Whyte, 2006). The SLAM problem investigates the possibility for a mobile robot to be placed at an unknown location in an unknown environment and to incrementally build a map of the environment while simultaneously determining its position within the map. In SLAM, both the trajectory of the robot and the location of all landmarks are estimated online. The robots just use, without actively modifying, the information extracted from the environment both to estimate their position and to dynamically build a map of an unknown area. The focus in SLAM is more on mapping the environment and localise a robot inside it, instead of coordinating a team of robots to maximise exploration speed. The aim of our work focuses instead on the ability of a team of autonomous robots to collaborate in the task of exploring an unknown and hazardous

terrain in the minimum amount of time. Fast exploration and coverage of unknown areas, along with coordination of the agents, are thus the primary concerns of our work. Moreover, our approach actively modifies the environment tagging it with devices able to store information during the exploration process, with the aim of using it as a common memory and computational workspace, thus simplifying the agents themselves, which do not need to be too computationally powerful and thus expensive. SLAM, on the contrary, heavily relies upon one computationally powerful robot, which has on board everything needed to compute a map of the area and localise itself within it. This leads to a more centralised system, which risks to fail whenever the robot itself fails, and is definitely slower since it cannot make use of different resources (agents) in parallel. To conclude, SLAM, while extremely effective ad reliable in certain scenarios where time is not crucial and the risk of losing a very expensive robot is very low, is not adequate in emergency scenarios, where these two factors are essential, and where one would instead use a more flexible approach capable of greater exploration speed and using a great number of much simpler and cheaper robots, easily replaceable in case of failures.

Chapter 3

Exploration Algorithms

This chapter is not available in ORA

Chapter 4

Evaluation

In this chapter, we evaluate the performance of the proposed exploration algorithms (MDFS, Brick&Mortar and HybridExploration) in a simulation environment, and compare them with the existing Ants algorithm. We also assess the performance of Agent2Tag-ERD and Tag2Tag-ERD, the proposed mechanisms for discovering evacuation routes when they are combined with different exploration algorithms. Simulation results are generated for a wide variety of scenarios, by varying the number of obstacles, the size and type of exploration area, the number of rooms and the number of agents. A description of these elements has been provided in Section 1.1, where the model used in this work is described in more detail. Furthermore, Appendix A contains examples of the various scenarios and the algorithms used to generate them.

4.1 Simulation Environment

We developed a simulation tool to test the performance of Ants (Svennebring and Koenig, 2004), MDFS (Section 3.1), Brick&Mortar (Section 3.2) and HybridExploration (Section 3.3). The tool allows us to automatically generate area maps with different topological features. Thus, we are able to study the impact of i) the number of obstacles, ii) the size of the area (total number of cells), iii) the number of rooms and iv) the number of agents on the performance of exploration algorithms. Each point in the following graphs is the average of running an algorithm 20 times, each time with a different randomly generated map that satisfies the input topological features.

Those features have been chosen for the following reasons:

- They reflect the parameters most frequently used in the literature (e.g., in (Dasgupta and Cheng, 2009; Agmon et al., 2006; Batalin and Sukhatme, 2005; Hazon and Kaminka, 2005; Howard et al., 2006), just to name a few) and can therefore be used to compare our approach to others in the field using the same settings. The only difference is that we often used the number of steps (or traversed cells) instead of absolute exploration time as measure to evaluate our results, for the reason that we did not want to tie our results to the performance (in particular, speed) of a certain kind of hardware. In this way, one only has to multiply the time necessary to a particular robot to cross a cell of the given size for the number of steps, to obtain the total time and compare the results with others using the time parameter.

- Among all the possible parameters, they are the ones with the largest impact on the performed experiments. For example, the size of the area affects different algorithms in different ways, and can therefore be used to test their scalability and compare each other, while the size of a single cell has of course an impact on the exploration time, but affects all algorithms in the same way, and cannot therefore be used to compare them.

- They are the most used by architects and first responders, and generally well known to people working in emergency scenarios, and can therefore be used to explain different topologies when talking to them and to carry across real information from real world to simulators.

- Furthermore, the number of obstacles is very important to test how well an algorithm can cope with the loop problem (see Section 3.2.2 for details), by which for example Brick&Mortar is very affected, while MDFS is largely immune. For this reason, every obstacle added to (or removed from) the map is represented by a single *wall* cell, and cannot be adjacent to an already existing *wall* cell. In this way, we can have a 1-to-1 relationship between number of obstacles and number of possible loops, and compare the algorithms accordingly.

- Finally, the number of agents is essential to test how effective an algorithm is in

using multiple agents at the same time, and in finding the threshold over which adding more agents does not improve the performance.

In each experiment, we vary the values of one parameter, and assign default values to the remaining ones. The default values are: a map of 2500 (50x50) cells with 30 obstacles and 36 (6x6) rooms, which is explored by 20 agents. The agents are deployed from the top left cell of the area. We consider two performance metrics: i) the *exploration time*, i.e. the number of steps required to achieve the *Exploration Objective*, and ii) the *visiting time*, i.e. the number of steps required to achieve the *Completion Objective*. Since Ants does not achieve the *Completion Objective*, it only appears in the graphs measuring exploration time (not in the graphs measuring visiting time).

Furthermore, after having studied possible configurations of real world buildings and underground areas, we chose three different area types, each representing a particular situation: i) *Office*: area inspired by a real building plan, with corridors, offices, and an open-space area, which an emergency has not heavily changed. ii) *Collapsed*: area in which a severe event occurred (i.e. earthquake) and disrupted the normal plan of the building. iii) *Series*: a long corridor which traverses several rooms; this scenario is inspired by a mine or another underground map in which each room has a door entering from the previous one and another door leading to the next. Examples of these area types are provided in Figure 4.1. Our objective is to investigate how the performance of the proposed and competing algorithms varies depending on the spatial layout of rooms and doors in a building.

4.2 Evaluation of Exploration Algorithms

The goal of this section is to show the impact of the area's topological features on the performance of exploration algorithms. Each set of experiments assesses the performance of exploration algorithms in three area types (Office, Collapsed and Series) using two performance metrics (exploration and visiting time). Hence, each figure typically consists of 6 graphs arranged in a 3-by-2 tabular form, where rows denote area types, and columns denote performance metrics (graphs in the left column measure exploration time, and

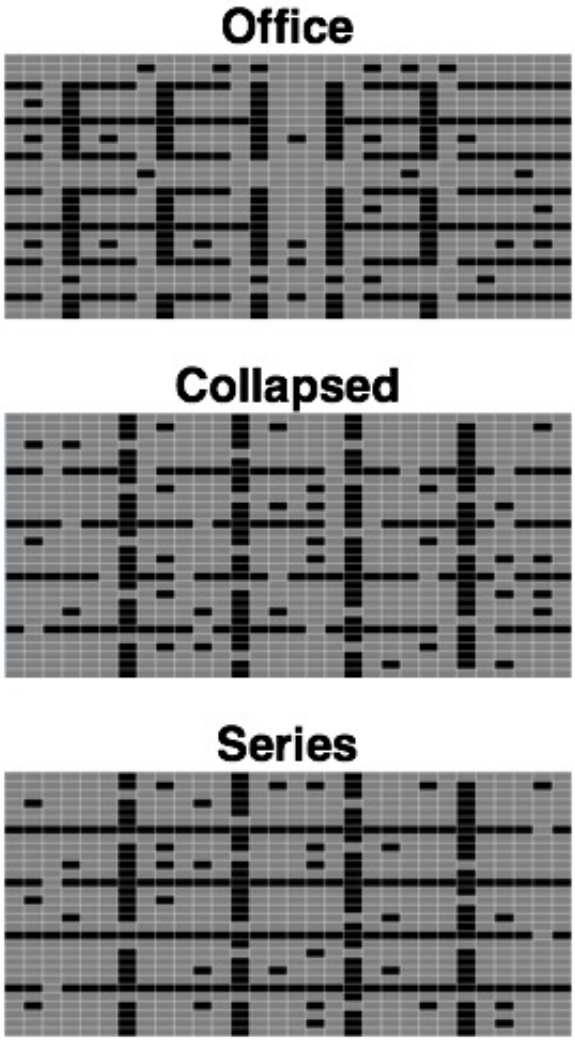

Figure 4.1: Different scenarios in which the algorithms were tested.

those in the right column measure visiting time).

Effect of obstacles: The first set of simulation results (Figure 4.2) shows the impact of varying the number of obstacles on the performance of exploration algorithms. Each obstacle is a single *wall* cell, and obstacles are distributed in the area uniformly at random, ensuring that they do not disconnect accessible parts of the area. We chose not to use bigger obstacles (more than one cell) because single-cell obstacles have the most impact on exploration time of the algorithms. Brick&Mortar for example needs to resolve a loop around each isolated obstacle, but it is not affected by having bigger cluster of *wall* cells, since the loop resolving mechanism does not vary in this case. The other algorithms are equally not affected by the size of the obstacles. On the contrary, bigger

obstacles would result in having less *unexplored* cells to traverse, and thus adding more obstacles might sometimes decrease the exploration time, thus yielding false results. In terms of *exploration time*, the three proposed algorithms are consistently faster than the existing Ants algorithm for all numbers of obstacles. Notice, for example, that MDFS is roughly twice as fast as Ants in the Office and Series scenarios and three times as fast as Ants in the Collapsed scenario. Besides the inferior performance of Ants, there is no clear winner among the three proposed algorithms in all three scenarios. An interesting observation, however, is that MDFS is hardly impacted by an increase in the number of obstacles (its exploration time is almost constant), whereas Brick&Mortar is the algorithm that is slowed down the most as obstacles are introduced. The reason is that the more the obstacles, the more the loops formed around obstacles, which Brick&Mortar agents must resolve during exploration.

In terms of *visiting time*, we notice a similar trend, i.e. that MDFS is hardly impacted by obstacles, and Brick&Mortar is affected the most. Notice that in the Collapsed scenario, the visiting time of Brick&Mortar is much higher than its exploration time, and much higher than the visiting time of the other two algorithms. The reason is that in the Collapsed scenario, each room has several entry/exit points due to collapsed walls. These *holes* in the walls result in the formation of large loops that enclose several rooms at a time. Brick&Mortar agents traverse each loop three times in order to resolve it, and the larger the loop, the longer the loop resolution process. This explains why Brick&Mortar is considerably slower in achieving the Completion Objective than MDFS (which does not deal with loops) and HybridExploration (which uses virtual agents to deal with loops).

Effect of area size: The second set of simulation results (Figure 4.3) shows the impact of varying the area size (total number of cells) on the performance of exploration algorithms. As expected, the exploration and visiting times of all algorithms increase with the area size. In terms of *exploration time*, the performance degradation of Ants is faster than that of the three proposed algorithms. The relative performance of MDFS and Brick&Mortar varies depending on the size and type of area, whereas HybridExploration, which combines the strengths of MDFS and Brick&Mortar, remains among the most efficient exploration algorithms in most cases.

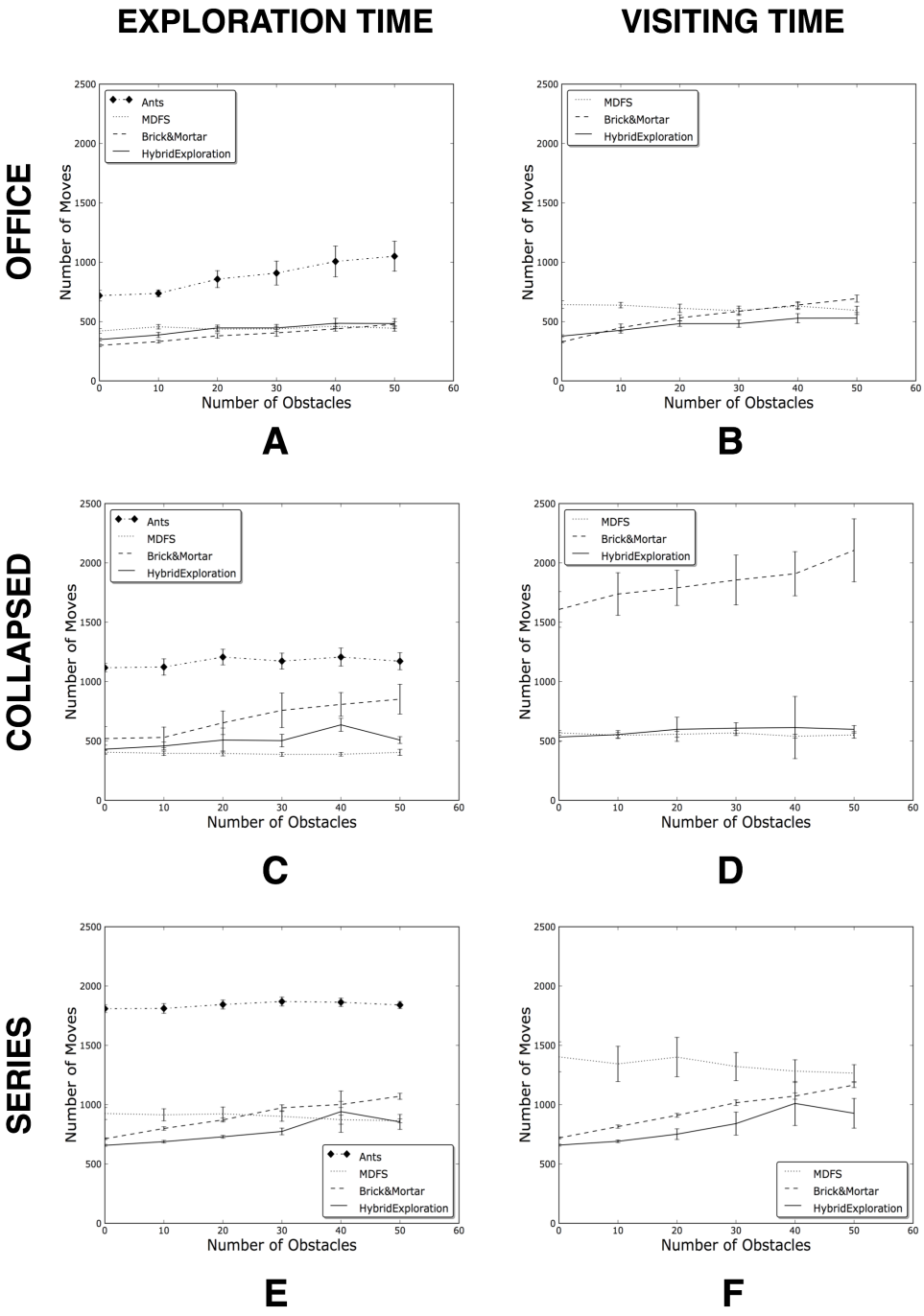

Figure 4.2: Effects of changing the number of obstacles.

In terms of *visiting time*, HybridExploration scales gracefully in the number of cells and remains among the fastest algorithms in achieving the *Completion Objective* in all cases. The visiting time of Brick&Mortar is significantly higher than that of MDFS and HybridExploration in the Collapsed scenario for all area sizes. The reason is that the collapsed walls in this scenario result in the formation of large loops that take a long time to resolve by Brick&Mortar agents, as explained above (effect of obstacles). In the other two scenarios (Office and Series), the relative performance of MDFS and Brick&Mortar depends on the area size. MDFS tends to be more efficient in areas with fewer than 2000-2500 cells, whereas Brick&Mortar becomes superior in larger areas. The reason is that MDFS typically covers each additional cell twice, whereas Brick&Mortar typically covers each additional cell once (unless it needs to leave it as a corridor). As the size of the area increases, the proportion of cells that Brick&Mortar uses as corridors and traverses more than once decreases.

Effect of rooms: The third set of simulation results (Figure 4.4) shows the impact of varying the number of rooms in the area while maintaining the same number of cells, and thus the same area size. In terms of *exploration time*, the proposed exploration algorithms are two to three times as fast as Ants, for all area types and numbers of rooms. Interestingly, there is no clear effect of the number of rooms in the exploration time of an algorithm: for example, as we increase the number of rooms, the exploration time of Ants decreases or remains constant in the Office scenario and Collapsed scenarios, and increases in the Series scenario. The reason is that the Office and Collapsed scenarios have corridors that allow agents to disperse through the rooms, whereas, in the Series scenario with many and small rooms, agents are congested and underutilise their resources. Similar trends are observed for MDFS and HybridExploration, i.e. their performance does not vary much in the Office and Collapsed scenarios, but degrades in the Series scenario, as we increase the number of rooms. Brick&Mortar deviates from this trend in the Collapsed scenario: the reason that its exploration time increases with the number of rooms is that the more the rooms, the more the collapsed walls in these rooms, and hence, the more the loops that agents must resolve.

In terms of *visiting time*, the performance of Brick&Mortar degrades as we increase

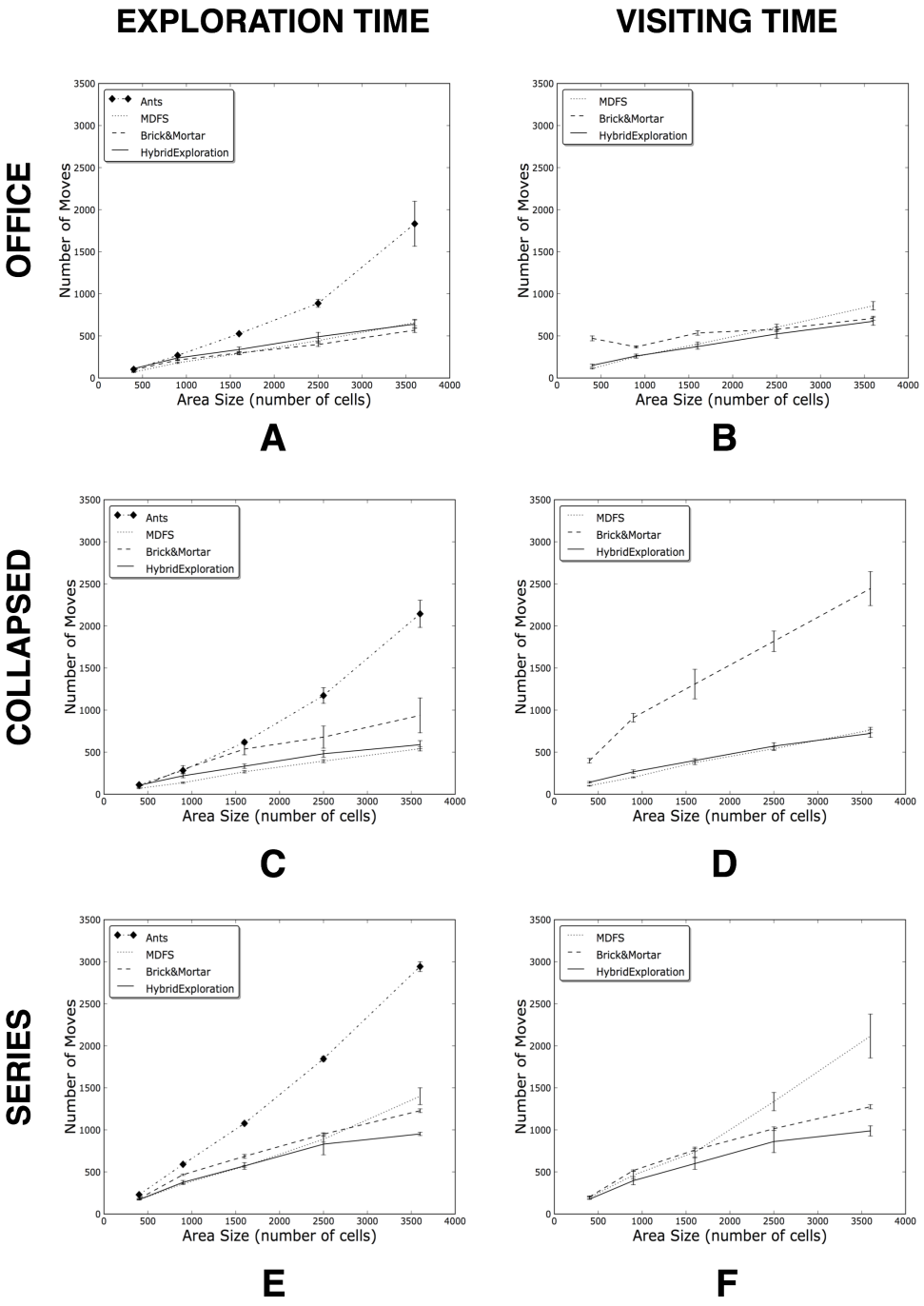

Figure 4.3: Effects of changing the size of the map.

the number of rooms in the Collapsed scenario, because of an increase in the number of loops formed by collapsed walls. Interestingly, in the other two scenarios, Brick&Mortar becomes faster in areas with more rooms. The reason is that in open-space areas, loops around obstacles are not bound within small rooms, but can be as large as the perimeter of the entire area. In areas with many rooms, loops around obstacles are typically small, and easy to resolve. The behaviour of MDFS is also different depending on the scenario. In the Office and Collapsed scenarios, the visiting time of MDFS hardly depends on the number of rooms. However, in the Series scenario, MDFS is slowed down by the presence of rooms, because its agents are confined to small and congested rooms, and are not able to disperse effectively in the area. HybridExploration, that combines the benefits of MDFS and Brick&Mortar, outperforms these two algorithms in all scenarios; the Series scenario presents the most interesting case, where the benefits of HybridExploration are more pronounced.

Effect of agents: The fourth set of simulation results (Figure 4.5) shows the impact of varying the number of agents on the performance of exploration algorithms. In terms of *exploration time*, the proposed algorithms are roughly 2 to 4 times as fast as Ants, depending on the area type and the number of agents. As expected, adding agents improves the performance of all algorithms. Note however that when the number of agents is small, adding one more agent significantly speeds up exploration (for example, in the Series scenario, using 5 instead of 3 agents almost halves the exploration time of Ants), whereas, beyond a certain number of agents (10 or 20), adding more agents has a minor influence on the algorithms' performance. The performance of the proposed algorithms is comparable in the Office and Series scenario, whereas Brick&Mortar is noticeably slower in the Collapsed scenario because of the time agents take to resolve loops.

In terms of *visiting* time, the performance of the proposed algorithms improves significantly as we add agents. However, beyond a certain point (of approximately 10 agents), adding more agents has minor effects. In the Office scenario, the proposed algorithms behave similarly; small differences are noticeable in scenarios with few agents, and they fade out as more agents are introduced. In the Collapsed scenario, Brick&Mortar is roughly three times slower than the other two algorithms, for most numbers of agents,

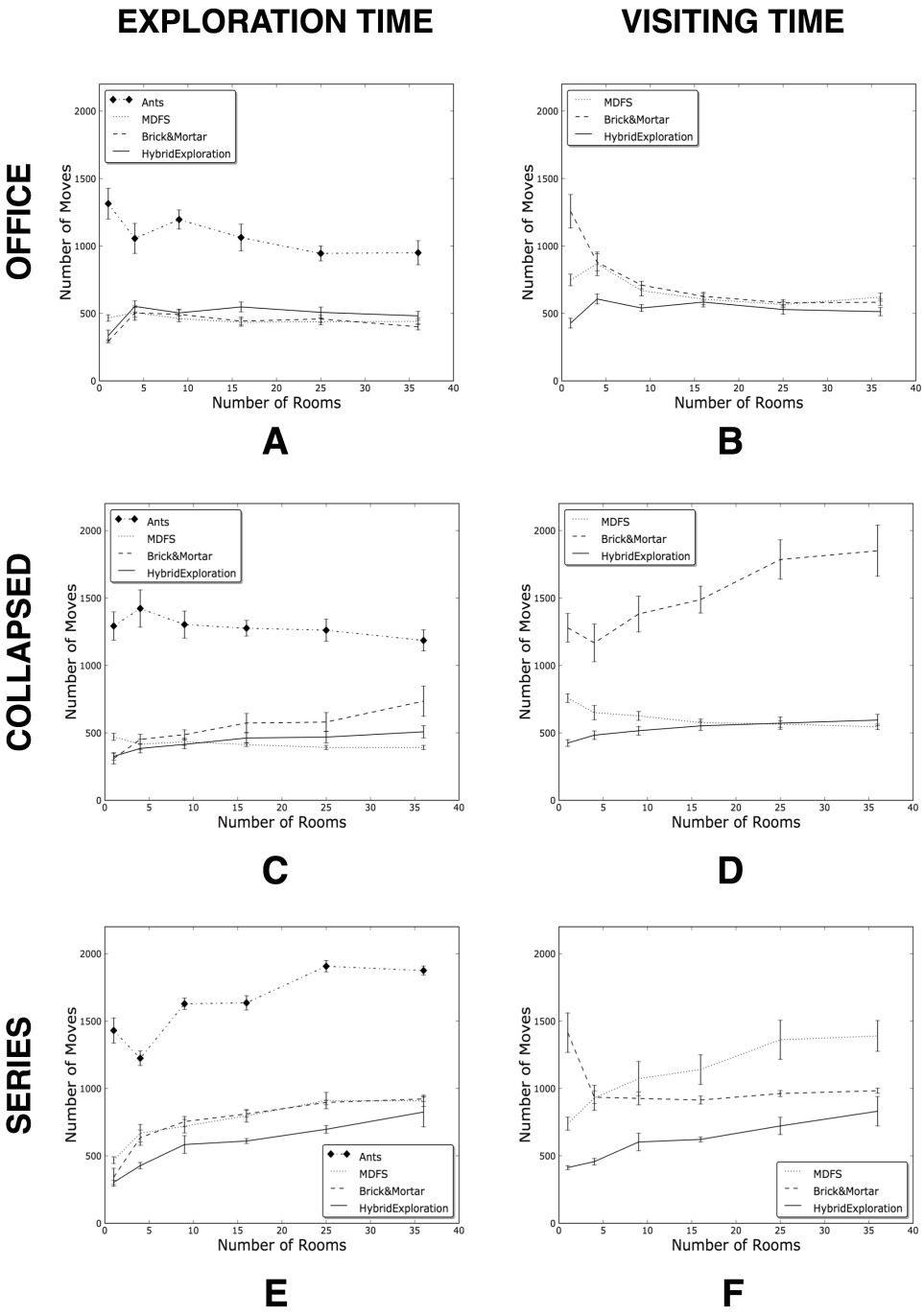

Figure 4.4: Effects of changing the number of rooms.

because of the time agents take to resolve loops. In the Series scenario, MDFS is faster than Brick&Mortar when one agent is used, but slower when more than three agents are used. This shows that Brick&Mortar makes better utilisation of agent resources than MDFS in confined spaces, where agents have few opportunities to disperse in the area. HybridExploration consistently outperforms the other two algorithms in all scenarios, and its benefits are more pronounced in the Series scenario.

4.2.1 Summary of Results

We are now in a position to summarise the most interesting results drawn from our empirical simulation study.

- The proposed exploration algorithms (MDFS, Brick&Mortar and HybridExploration) are consistently shown to be twice to 5 times as fast as the existing Ants algorithm in terms of exploration time. Hence, they are more efficient in achieving the Exploration Objective, and faster in detecting hazards and victims in the area. In addition, unlike Ants, they are able to achieve the Completion Objective, which means that they are locally aware of when the entire area has been visited.

- As expected, the exploration and visiting times of all algorithms increase with the area size and decrease with the number of agents. However, the effect of varying the number of rooms and obstacles is not as clear; it depends on the algorithm and on the area type (Series, Collapsed, Office).

- The relative performance of MDFS and Brick&Mortar varies depending on the simulation scenario. However, MDFS and Brick&Mortar consistently exhibit special strengths and weaknesses. On the one hand, MDFS is better than Brick&Mortar in dealing with loops formed around obstacles, or around rooms with collapsed walls. This is why MDFS has lower exploration and visiting times than Brick&Mortar in the Collapsed scenario. On the other hand, in the absence of loops, Brick&Mortar typically exhibits lower visiting times than MDFS. The reason is that Brick&Mortar agents typically mark cells as *visited* the first time they traverse them, so that they do not need to revisit them. MDFS agents, however, typically traverse cells more

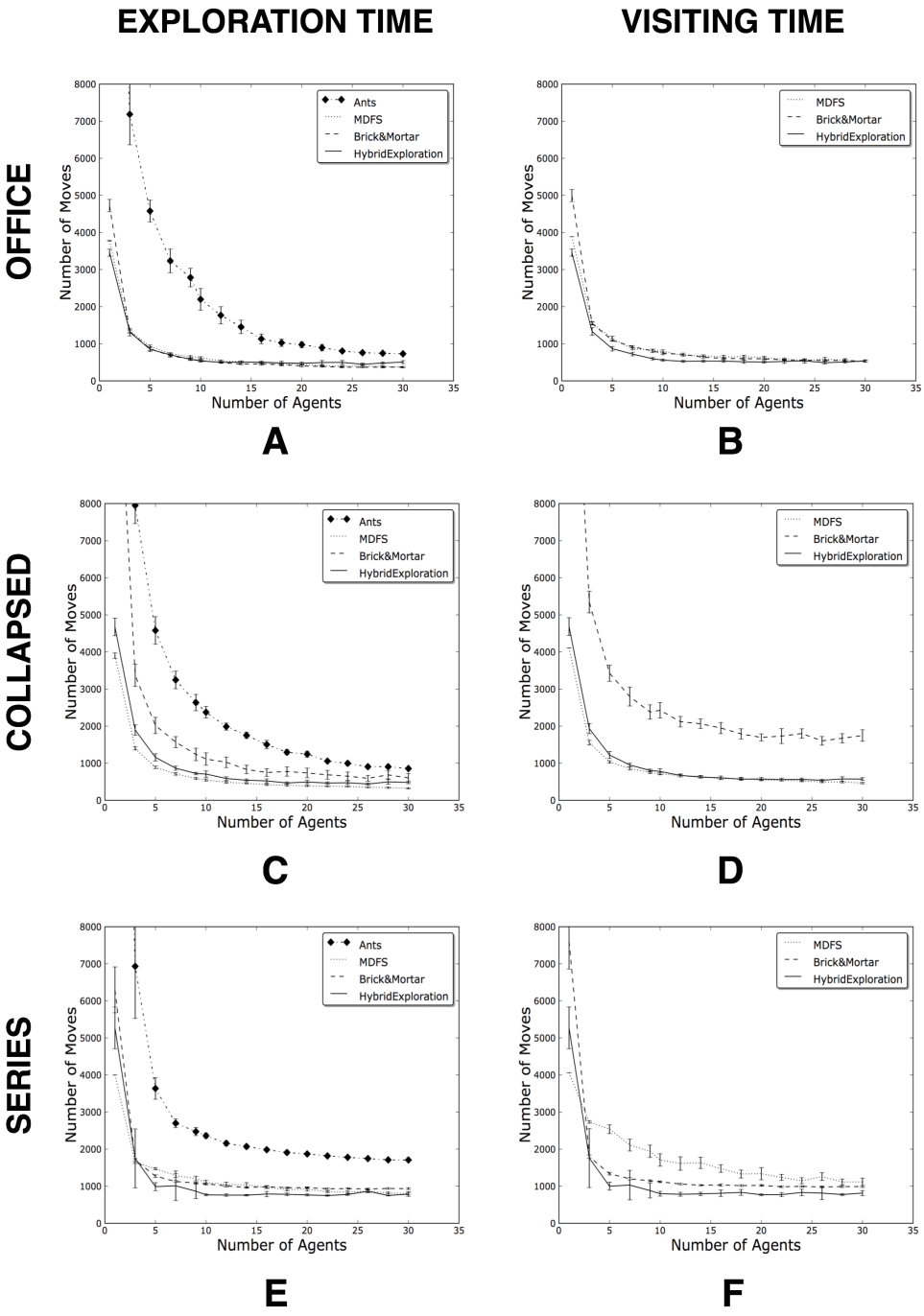

Figure 4.5: Effects of changing the number of agents.

than once, especially in the Series scenario, where agents cannot easily disperse in the area.

- The HybridExploration algorithm outperforms the other exploration algorithms in terms of exploration and visiting times, in the majority of simulated scenarios. The reason is that it combines the strengths of all the other algorithms. First, like MDFS, it adopts a depth-first-search approach to resolve loops around obstacles, but using virtual agents (messages) instead of physical agents. Second, like Brick&Mortar, physical agents running HybridExploration typically traverse cells once before they can mark them as *visited*. Third, in the presence of obstacles, the physical agents of HybridExploration use the Ants approach to escape from loops without resolving them. Hence, whereas Brick&Mortar agents spend a lot of time resolving loops, HybridExploration agents escape from loops as soon as possible, and leave the tedious task of loop resolution to virtual agents.

4.3 Evaluation of Evacuation Route Discovery Mechanisms

The last section of this chapter is dedicated to the evaluation of the evacuation route discovery (ERD) mechanisms proposed in Section 3.4. We first investigate the impact of the area's topological features on the discovery of evacuation routes, and then study the interplay between exploration and ERD techniques.

4.3.1 Impact of Area Topology on Evacuation Route Discovery

In Chapter 3, we proposed two distinct mechanisms for discovering evacuation routes: Agent2Tag-ERD, which is based on agent-to-tag communication, and Tag2Tag-ERD, which is based on inter-tag communication. In this section, we evaluate the performance of the two ERD mechanisms as we vary topological features of the area, such as the number of obstacles, the number of cells, the number of rooms, the number of agents, the number of exits and the layout of the area (Office, Collapsed, Series). For the purposes of

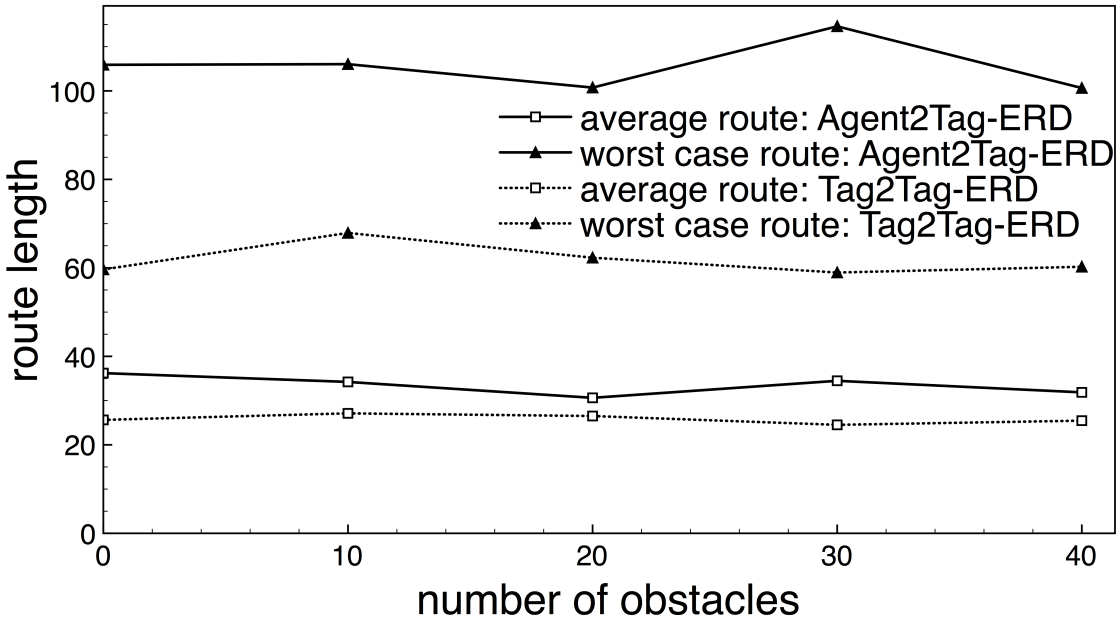

Figure 4.6: Effects of changing the number of obstacles.

this experiment we fix the exploration algorithm to Brick&Mortar, and only vary the number of agents tasked to explore an area and discover evacuation routes; the performance of ERD mechanisms with other exploration algorithms is evaluated in the next subsection.

In each of the following simulations, we vary the values of one parameter, and assign default values to the remaining ones. The default values are an Office area of 2500 (50×50) cells, with 4 rooms, no obstacles and 5 emergency exits, which is explored by 5 Brick&Mortar agents. Our goal is to compare Agent2Tag-ERD and Tag2Tag-ERD in terms of two performance metrics: 1) the average length of evacuation routes and 2) the length of the worst-case (longest) evacuation route. To evaluate the average and worst case route lengths, we assume that victims are found in all cells of the area, and evacuation routes are formed between each cell and one of the emergency exits. Optimum routes have been omitted because of their constant overlapping with Tag2Tag-ERD plots, since Tag2Tag-ERD is able to achieve optimum routes in most scenarios.

Impact of obstacles: In Figure 4.6 we can see that obstacles (e.g., desks in the middle of rooms) do not greatly influence the length of evacuation routes, because they typically occupy at most one cell.

Impact of area size: Figure 4.7 measures the average and worst-case route lengths of

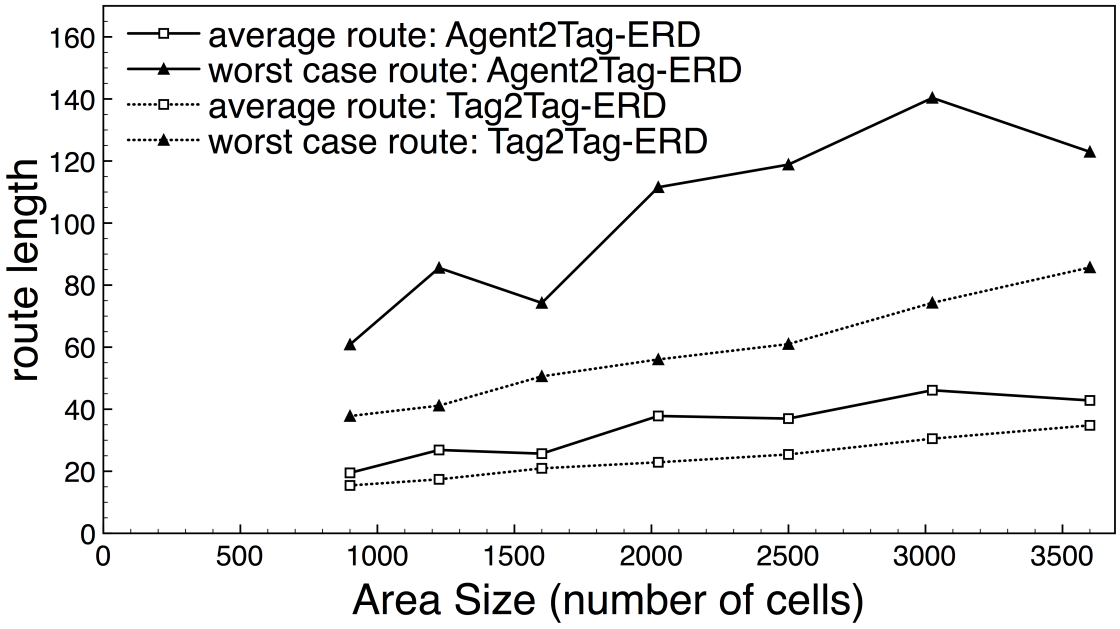

Figure 4.7: Effects of changing the area size.

Agent2Tag-ERD and Tag2Tag-ERD, as we vary the number of cells in the area. As one would expect, the route lengths increase as we increase the area size for both evacuation route discovery mechanisms. Whereas the difference between the average route lengths is not significant (see lower two line-plots), the use of inter-tag communication is shown to reduce significantly the length of the worst-case evacuation routes (see higher two line-plots). Hence, Tag2Tag-ERD is much more efficient in addressing the worst-case scenario where a victim is found far away from an emergency exit. The large gap between Agent2Tag-ERD and Tag2Tag-ERD in the worst-case scenario is observed for most area sizes.

Impact of rooms: Figure 4.8 shows an interesting trend in the interval between 1 and 10 rooms, while with more than 10 rooms the performance of Agent2Tag-ERD and Tag2Tag-ERD remains constant. The average and worst-case routes with 1 room are short, because the area is an open space, and agents do not have to navigate around room walls to access internal room cells. Initially, as we increase the number of rooms, the room walls act as long obstacles, leading to longer evacuation routes. As we increase the number of rooms further (> 10) what used to be long walls of a few rooms, become smaller walls interrupted by many room doors. Hence, although the total number of wall

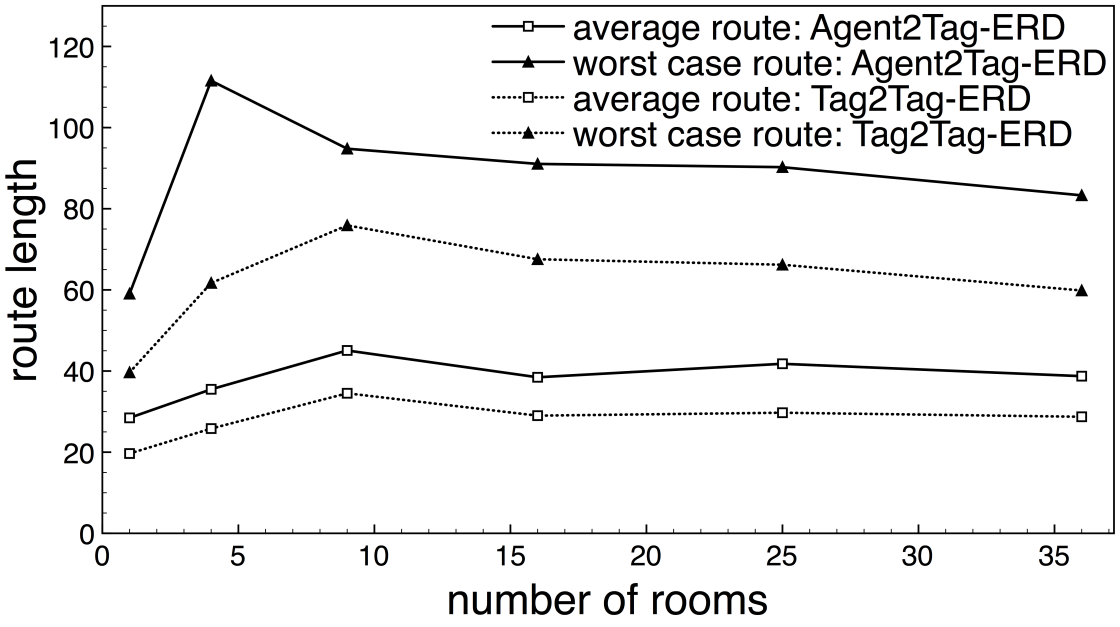

Figure 4.8: Effects of changing the number of rooms.

cells increases (leading to longer routes), access to internal room cells becomes easier (leading to shorter routes). Hence the cumulative effect is that agents tend to find routes of similar length as we increase the number of rooms from 10 to 35.

Impact of agents: Figure 4.9 shows that the number of agents that explore the area and look for evacuation routes, has little effect on the length of evacuation routes discovered eventually when the exploration process is finished. This is not to be confused with the fact that, during exploration, the more the agents the faster these routes will be discovered. As observed in most of the previous graphs, using or not inter-tag communication does not significantly improve the average route length. On the other hand, the fact that Tag2Tag-ERD uses inter-tag communication makes it much more efficient than Agent2Tag-ERD, in evacuating victims that are far from emergency exits.

Impact of emergency exits: As one would expect, Figure 4.10 shows that the more the emergency exits, the shorter the evacuation routes to them. Inter-tag communication only slightly improves the average route length - an observation that is consistent with most of the previous graphs. In terms of the worst case scenario (longest evacuation routes), Tag2Tag-ERD tends to take better advantage of adding more emergency exits than Agent2Tag-ERD. The reason is that it always manages to find the shortest routes to

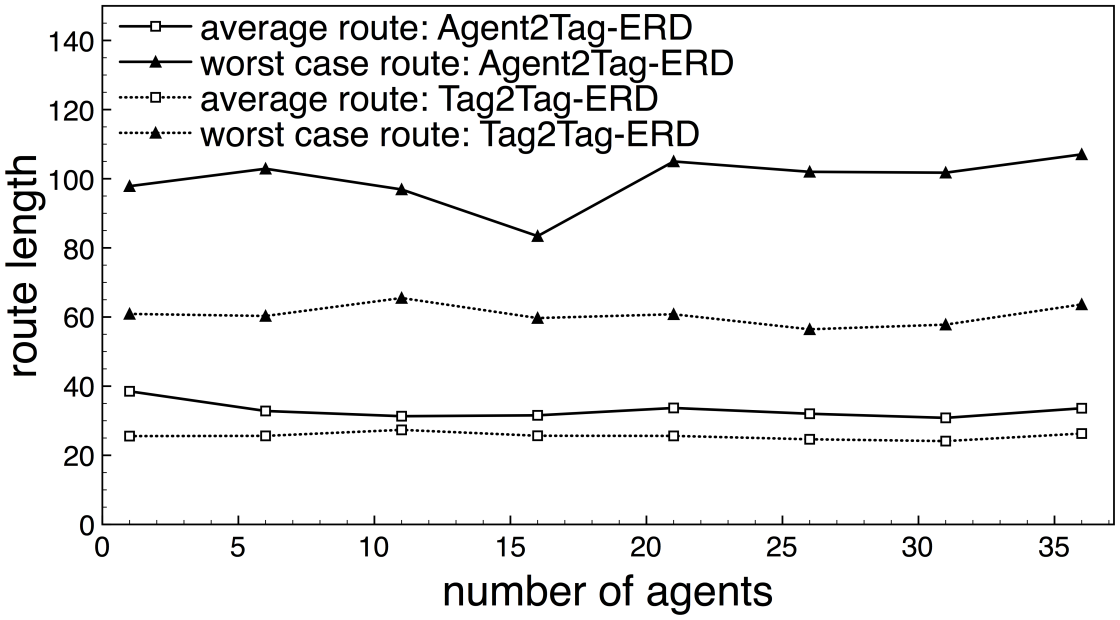

Figure 4.9: Effects of changing the number of agents.

these exits, independent of routes that agents followed whilst exploring the area.

Impact of room layout: Finally, in Figure 4.11 we used the default values of cells, rooms, obstacles, agents and emergency exits, and only varied the layout of rooms in the area, leading to three different scenarios: Office, Collapsed and Series. Introducing inter-tag communication is more beneficial in the Series scenario, followed by Office and Collapsed. In general, when considering all three scenarios, Tag2Tag-ERD is up to 35% more efficient than Agent2Tag-ERD in building evacuation routes on average, and up to 45% more efficient in finding evacuation routes to remotely located victims.

4.3.2 Interplay Between Exploration and Route Discovery Techniques

In this section, we study the interplay between exploration algorithms and mechanisms for discovering evacuation routes. We consider an *office* area with 2500 (50×50) cells consisting of 4 rooms and 5 emergency exits, in which we deploy 5 agents to explore it and discover evacuation routes.

The first question that arises is how different exploration algorithms perform when integrated with Agent2Tag-ERD, our first mechanism for evacuation route discovery. Fig-

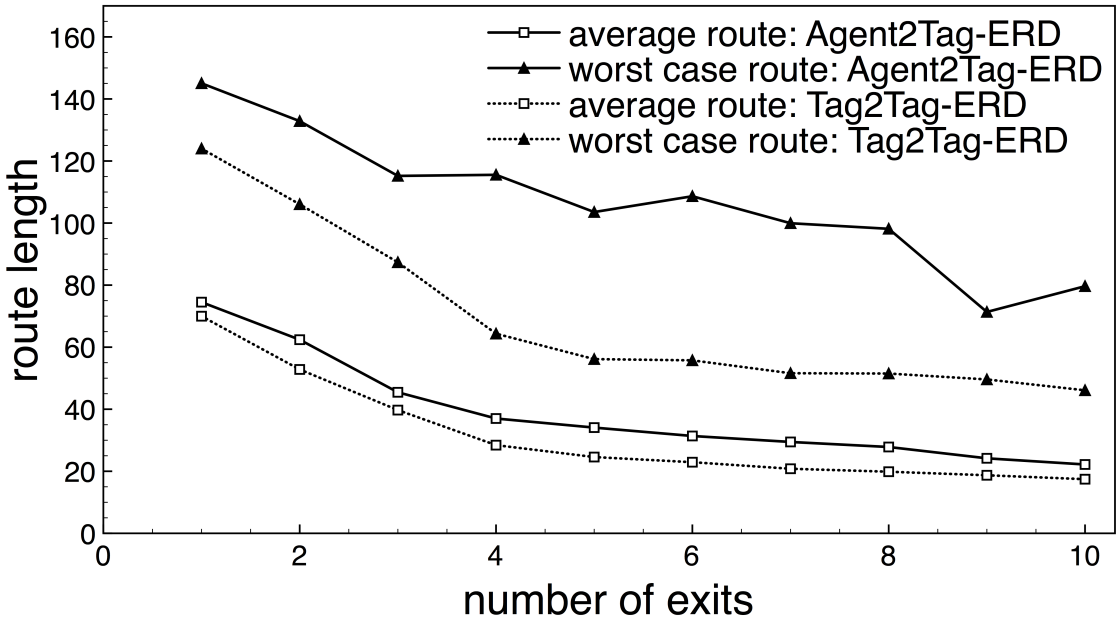

Figure 4.10: Effects of changing the number of exits.

ure 4.12 shows the lengths of evacuation routes from a victim location as time elapses. We have set the victim location to be the cell in the middle of the exploration area. Brick&Mortar, which explores the area faster, finds a route to the victim earlier than the other two algorithms. However, without tag-to-tag communication, once it finds a route, it rarely replaces it later with a shorter route. On the other hand, MDFS and Ants find evacuation routes later in time, but they often get the opportunity to gradually update these routes with shorter ones, yielding eventually more efficient routes than Brick&Mortar. The reason is that agents running Brick&Mortar usually traverse most of the cells only once in order to save time, and thus do not return to improve the routes in already visited areas. On the other hand, MDFS agents typically traverse each cell twice (first to explore it, and then to mark it as *visited*) and thus perform better in terms of disseminating route information. Finally, agents running the Ants algorithm always manage to find the optimal evacuation routes, because they continue to explore the area indefinitely, revisiting cells multiple times, and continuously improving existing evacuation routes.

The second question that arises is how exploration algorithms perform when integrated with Tag2Tag-ERD, our second mechanism for evacuation route discovery. Recall that this mechanism exploits tag-to-tag communication to improve discovered evacua-

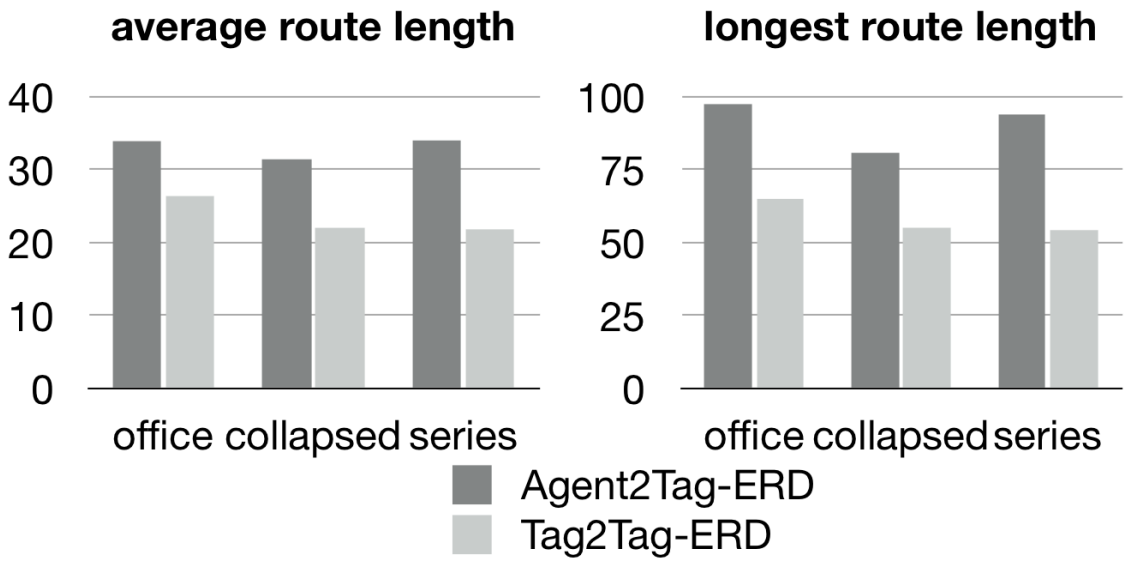

Figure 4.11: Effects of changing the type of scenario.

tion routes without requiring the presence of agents. Figure 4.13 measures the length of the average and worst-case evacuation routes discovered when the three algorithms complete the exploration task. In the x-axis we vary the percentage of faulty links between adjacent tags. The value 0% corresponds to perfect communication between adjacent tags, whereas the value 100% reflects no inter-tag communication. As we increase the percentage of broken links from 0% to 100%, the Tag2Tag-ERD mechanism degrades from its ideal performance (when all inter-tag links are available) to the performance of Agent2Tag-ERD that utilises no inter-tag links.

Figure 4.13 shows that with perfect inter-tag communication (left part of the graph), the performance of the three exploration algorithms is identical. The average length of evacuation routes to various victims, as well as the longest (worst-case) route to the remotest victim, do not depend on the exploration algorithm. However, as we increase the number of faulty links, the gaps between the three exploration algorithms increase, especially in the case of the worst-case (longest) evacuation route. As inter-tag links deteriorate, the Ants algorithm continues to identify short evacuation routes, because its agents roam the network multiple times, whereas the other two algorithms are only able to discover suboptimal routes.

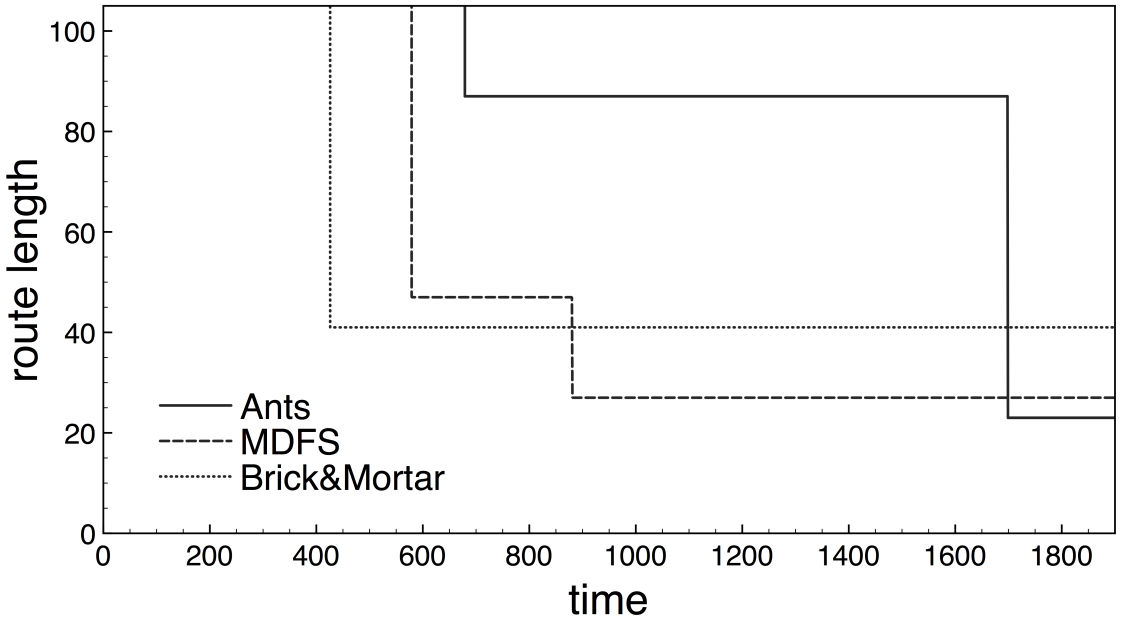

Figure 4.12: Length of evacuation route from a victim to an emergency exit, when Agent2Tag-ERD is combined with the three exploration algorithms. As long as agents have not reached the victim, the route length is infinite.

4.3.3 Summary of Results

We are now in a position to summarise the most interesting results drawn from our empirical simulation study of evacuation route discovery (ERD) mechanisms.

- The choice of exploration algorithm largely affects the time when victims are found and evacuation routes are first discovered. For example, in areas without obstacles, where Brick&Mortar tends to be more efficient than MDFS and Ants in terms of exploration time, it is also faster in discovering evacuation routes.

- Without inter-tag communication (Agent2Tag-ERD), the choice of the exploration algorithm also affects the length of discovered evacuation routes. Algorithms that tend to revisit the same cells multiple times (Ants and MDFS) gradually discover shorter evacuation routes than faster exploration algorithms (Brick&Mortar) that avoid going back to the same cells.

- With inter-tag communication (Tag2Tag-ERD), all exploration algorithms have the same performance both in terms of the average and worst-case evacuation routes.

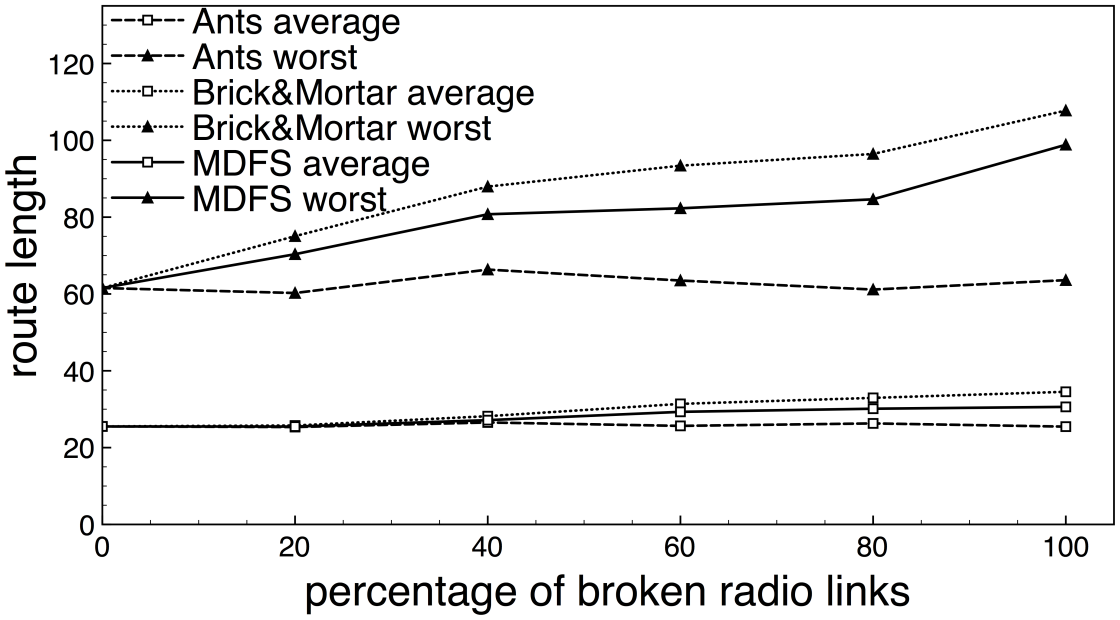

Figure 4.13: Average and worst-case (longest) route lengths as we vary the percentage of broken links between adjacent tags. The left part of the x-axis corresponds to perfect inter-tag communication (Tag2Tag-ERD) whereas the right part corresponds to no inter-tag communication (Agent2Tag-ERD).

- For a given exploration algorithm, Agent2Tag-ERD yields longer evacuation routes than Tag2Tag-ERD. The benefits of inter-tag communication become more obvious (up to 50%) when one considers the length of the longest route to remote victims.

- The quality of evacuation routes depends on the topological features of the area. The dependence is higher on the area size, the layout of rooms and the number of exits, and lower on the number or rooms and the number of obstacles in the rooms.

Chapter 5

Real Deployment

This chapter is not available in ORA

Chapter 6

Fault Tolerant Exploration Algorithms

Simulation results in Chapter 4 showed that, in the absence of detection errors, the three proposed algorithms (MDFS, Brick&Mortar and HybridExploration) typically outperform Ants in terms of exploration time. Further simulations in Chapter 5, however, revealed that, unlike Ants, these algorithms fail to successfully explore an area when tag detection errors are introduced. The objective of this chapter is to improve the three algorithms and make them robust to tag detection errors while maintaining their relative advantage over Ants. The remainder of this chapter is organized as follows: Section 6.1 proves that Ants always achieves the *Exploration Objective*, even in the presence of detection errors. Sections 6.2-6.4 present Fault-Tolerant (FT) versions of the existing exploration algorithms and provide intuitive justifications that they all achieve the Exploration Objective. Section 6.5 evaluates the performance of the FT algorithms in a simulation environment, and compares them with the existing Ants algorithm.

6.1 Ants is Fault Tolerant

In this section, we prove how the Ants exploration algorithm always achieves the *Exploration Objective*, even in the case in which detection errors are injected into the system. The intuitive justifications are based on the fact that Ants assume that the agents have infinite energy and can therefore run forever. This makes sense if one thinks that Ants is an algorithm proposed to constantly patrol an area, with the possibility for the agents to obtain more energy (e.g., from power plugs) during time. The first intuitive justification

(without detection errors) has already been provided by Koenig et al. in (Koenig et al., 2001), and is summarised below:

Theorem 7. *The Ants algorithm always achieves the* Exploration Objective *in the absence of detection errors.*

Intuitive Justification Let us assume that the Ants algorithm does not achieve the *Exploration Objective*. This would imply that all agents are repeatedly traversing a cyclic path of cells without leaving it. For this to happen, the pheromone level of the cells of the path will increase indefinitely, and thus there will be a time in which one of the cells adjacent to the path will have a pheromone level lower than the ones belonging to the path itself. At this point, the agents will need to step on that cell, thus breaking the cycle. This proves that agents running Ants repeatedly traverse every cell in the map, and thus they always achieve the *Exploration Objective*. □

Theorem 8. *The Ants algorithm achieves the* Exploration Objective *in the presence of detection errors.*

Intuitive Justification Let us assume that the Ants algorithm does not achieve the *Exploration Objective*. This would imply that all agents are repeatedly traversing the same subset of cells C without leaving it. The pheromone level of each of these cells will increase indefinitely. Let c be one of the cells in the set C with continuously increasing pheromone levels that has at least one adjacent cell c' not belonging to the same group C. Let c' have the least pheromone among all adjacent nodes of c. Every time an agent visits cell c, it will step on c' with a non-zero probability $p_{cd} = 1 - detectionError$, where p_{cd} is the probability of the agent correctly detecting the tag on cell c'. As the number of times n that an agent visits c goes to infinity, the expected number of times that it will also visit c' also goes to infinity (since $E(X) = n * p_{cd}$ in Binomial distribution). Hence, c' also belongs to the group of cells C whose pheromone levels increase continuously, which is a contradiction. □

6.2 Fault Tolerant - Multiple Depth First Search Algorithm

In this section, we present the Fault-Tolerant - Multiple Depth First Search (FT-MDFS) algorithm, which is designed to cope with detection errors. Recall that MDFS agents build an exploration tree and mark cells as *explored* while traversing the tree downwards, and as *visited* while traversing the tree upwards (see Algorithm 1 in Chapter 3).

- If a detection error occurs when traversing a tree downwards, an agent may fail to traverse *unexplored* cells that lie beyond the undetected tag. Instead, it starts traversing the tree upwards, marking cells as *visited*, and isolating in this way the unexplored area beyond the undetected tag.

- If a detection error occurs when traversing a tree upwards, an agent may erroneously consider itself to be surrounded by *visited* and *wall* cells. In this case, it will mark the current cell as *visited* and will stop exploring the area, although there still remain cells that have been left *unexplored*.

In either case, the result is similar, i.e. an agent running MDFS stops before all cells are traversed at least once.

The new FT-MDFS algorithm overcomes this problem, by starting with MDFS and switching to the Ants approach whenever an agent thinks that it is surrounded by *visited* and *wall* cells. In this way, the agent will eventually manage to escape from the *visited* area where it is trapped, and will switch back to MDFS when it encounters *explored* or *unexplored* cells. In particular, if it encounters an *explored* cell, the agent will join the existing depth-first-search tree that the cell belongs to, whereas, if it encounters an *unexplored* cell, it will start a new depth-first-search tree rooted at that cell. The pseudocode of the FT-MDFS algorithm is provided in Algorithm 6.

Theorem 9. *The FT-MDFS algorithm always achieves the* Exploration Objective.

Intuitive Justification In FT-MDFS, each agent alternates between applying MDFS rules and Ants rules. In the MDFS phase, it marks at least one cell as *visited* (the cell

Algorithm 6: Fault Tolerant - Multiple Depth First Search Algorithm

```
1 while true do
2 |   if there is at least one adjacent unexplored or explored cell then
3 |   |   agents run the original MDFS algorithm (see Algorithm 1);
4 |   end
5 |   else if all adjacent cells are visited or wall then
6 |   |   agents runs the Ants algorithm on visited cells;
7 |   end
8 end
```

where it finds itself trapped within *visited* and *wall* cells). In the Ants phase, it always manages to walk out of *visited* areas and to access an *unexplored* or *explored* cell (if any such cell still exists in the area)[1]. This alternation between MDFS and Ants is repeated until no cell is left in the *explored* or *unexplored* state. However, when this happens, agents are not aware of it, and they continue to run Ants rules indefinitely. Thus, FT-MDFS achieves the *Exploration Objective*, but fails to satisfy the local condition of the *Completion Objective*. □

6.3 Fault Tolerant - Brick&Mortar Algorithm

In this section, we present the Fault Tolerant - Brick&Mortar (FT-Brick&Mortar) algorithm, which is designed to cope with detection errors. The FT-Brick&Mortar algorithm combines the Physical Agent Protocol (see Section 3.3.1) with Ants. Recall that the Physical Agent Protocol uses Brick&Mortar combined with Ants to escape from loops of *explored* cells. The pseudocode of the FT-Brick&Mortar algorithm is provided in Algorithm 7.

Theorem 10. *The FT-Brick&Mortar algorithm always achieves the* Exploration Objective.

Intuitive Justification

In FT-Brick&Mortar, each agent alternates between applying Physical Agent Protocol rules and Ants rules. In the Physical Agent Protocol phase, it marks at least one

[1]The underlying assumption here is that the tag detection error is less than 100%

Algorithm 7: Fault Tolerant - Brick&Mortar Algorithm

```
1 while true do
2 |   if there is at least one adjacent unexplored or explored cell then
3 |   |   agents run the Physical Agent Protocol (see Section 3.3.1);
4 |   end
5 |   else if all adjacent cells are visited or wall then
6 |   |   agents runs the Ants algorithm on visited cells;
7 |   end
8 end
```

cell as *visited* (the cell where it finds itself trapped within *visited* and *wall* cells). In the Ants phase, it always manages to walk out of *visited* areas and to access an *unexplored* or *explored* cell (if any such cell still exists in the area)[2]. We only need to prove that an agent will not run the Physical Agent Protocol forever, while endlessly traversing a subset of *explored* cells, disconnected from other *explored* or *unexplored* cells. Indeed, with a non-zero detection error, there is a non-zero probability that all four cells adjacent to the current cell are perceived by an agent as *visited* or *walls*. This probability tends to 1 as the number of steps tends to infinity. Thus, eventually, an agent will be trapped within *visited* and *wall* cells and will switch to Ants. This alternation between Physical Agent Protocol and Ants is repeated until no cell is left in the *explored* or *unexplored* state. However, when this happens, agents are not aware of it, and they continue to run Ants rules indefinitely. Thus, FT-HybridExploration achieves the *Exploration Objective*, but fails to satisfy the local condition of the *Completion Objective*. □

6.4 Fault Tolerant - HybridExploration Algorithm

In this section, we present the Fault Tolerant - HybridExploration (FT-HybridExploration) algorithm, which is designed to cope with detection errors. Recall that HybridExploration consists of two different protocols, the Physical Agent Protocol and the Virtual Agent Protocol, as described in Section 3.3. In FT-HybridExploration, the Physical Agent Protocol is combined with Ants exactly in the same way as in FT-Brick&Mortar, since the two algorithms are very similar. Thus, whenever an agent is surrounded by *visited* or *wall* cells,

[2]The underlying assumption here is that the tag detection error is less than 100%

it switches to Ants until it finds an *explored* or *unexplored* cell (if any), at which point it falls back to the Physical Agent Protocol. In FT-HybridExploration, the Virtual Agent Protocol is modified as follows: whenever a physical agent switches back from Ants to the Physical Agent Protocol, it spawns a new virtual agent, which will help it resolve loops in this new part of the map. The pseudo code of FT-HybridExploration is provided in Algorithm 8.

Algorithm 8: Fault Tolerant - HybridExploration Algorithm

```
1 while true do
2 |   if there is at least one adjacent unexplored or explored cell then
3 |   |   agents run the HybridExploration algorithm (see Section 3.3.1);
4 |   end
5 |   else if all adjacent cells are visited or wall then
6 |   |   agents run the Ants algorithm on visited cells;
7 |   end
8 end
```

Theorem 11. *The Fault Tolerant - HybridExploration algorithm always achieves the* Exploration Objective.

Intuitive Justification The physical agents of FT-HybridExploration alternate between the Physical Agent Protocol and Ants, exactly like FT-Brick&Mortar agents. Hence, FT-HybridExploration achieves the *Exploration Objective* and fails to achieve the *Completion Objective*, for the reasons presented in the intuitive justification of Theorem 10. □

6.5 Evaluation of Fault Tolerant Exploration Algorithms

In this section, we evaluate the performance of Ants and the proposed fault-tolerant exploration algorithms in the presence of tag detection errors. Simulation experiments are performed on automatically generated environments representing office-like scenarios, like the one in Figure 5.11, with a default area size of 2500 (50x50) cells and 36 (6x6) rooms. The positions of doors and walls are changed randomly during the experiments,

while the default number of agents is set to 20. The default number of obstacles introduced is 30. The performance metric we use is *exploration time*, which is defined as the number of moves that an algorithm requires to achieve the *Exploration Objective*. The exploration time of different algorithms is evaluated as we vary the number of obstacles, the area size (number of cells), the number of rooms and the number of agents. We also consider three different values of detection errors: 0%, 5% and 10%

Effect of obstacles: Figure 6.1 shows the performance of Ants and the proposed FT algorithms, as we vary the detection error and the number of obstacles. Increasing the number of obstacles does not significantly affect the performance of the proposed FT-algorithms, which consistently remain faster than Ants. However, the relative advantage over Ants drops as we increase the tag detection errors.

Effect of area size: Figure 6.2 shows the performance of Ants and the proposed FT algorithms, as we vary the detection error and the size area, i.e. the number of cells. Obviously, the larger the area, the longer the exploration times for all algorithms. The FT algorithms are superior to Ants, and their relative advantage increases as the area size increases (a trend that was also observed in the original MDFS, Brick&Mortar and HybridExploration algorithms). However, their performance deteriorates and approach that of Ants as we increase the tag detection error.

Effect of rooms: Figure 6.3 shows the performance of Ants and the proposed FT algorithms, as we vary the detection error and the number of rooms. Notice that in open spaces, the performance advantage of the FT algorithms over Ants is more significant than in spaces with many rooms. For example (Figure 6.3B), with a 5% detection error, FT algorithms are 4 to 5 times as fast as Ants in exploring areas with 1 room, and only $\sim 20\%$ faster in exploring areas with 36 rooms. This is due to the fact that when an agent running Ants takes a wrong decision because of a detection error it usually goes back towards already traversed parts of the map, and away from the exploration front of the other agents. Since those explored parts usually have very homogenous pheromone levels, the agent needs to find a wall to change its direction and go back towards the other exploring agents. In an open space area, the only walls are the ones at the border of the map, and thus agents need to wander for a great amount of time before starting traversing

unexplored parts again. The opposite is true when adding more rooms (and thus walls to "contain" the agents) to the scenario. The performance of all FT algorithms is comparable in most settings.

Effect of agents: Figure 6.4 shows the performance of Ants and of the proposed FT algorithms, as we vary the detection error and the number of agents. As expected, all algorithms improve their exploration time as more agents are used, and deteriorate as more detection errors are introduced (plots A-C of Figure 6.4). The proposed FT algorithms maintain a significant performance advantage over the existing Ants algorithm in all settings, but this relative advantage drops as the detection error increases. The reason is that the higher the detection error, the more FT algorithms utilise Ants rules to deal with cases of undetected tags. For example, with 10 agents, FT-HybridExploration is 4 to 5 times as fast as Ants when the detection error is 0% (Figure 6.4(A)), and $\sim 25\%$ faster than Ants when the detection error is 10% (Figure 6.4(C)). The three FT algorithms have very similar performance, especially when multiple agents are used for exploration.

6.6 Discussion

In this chapter, fault-tolerant (FT) versions of MDFS, Brick&Mortar and HybridExploration were proposed to cope with tag detection errors. The proposed FT algorithms combine the original algorithms with the Ants approach, in order to ensure that their agents achieve the Exploration Objective. Simulation experiments have shown that the proposed FT algorithms are typically faster than the existing Ants algorithm in a variety of settings with different numbers of obstacles, rooms, cells and agents. The performance advantages of FT algorithms over Ants become less pronounced as the tag detection error increases. The reason is that higher detection errors lead FT algorithms to utilize Ants rules more frequently, and make the behavior of FT algorithms more and more similar to Ants.

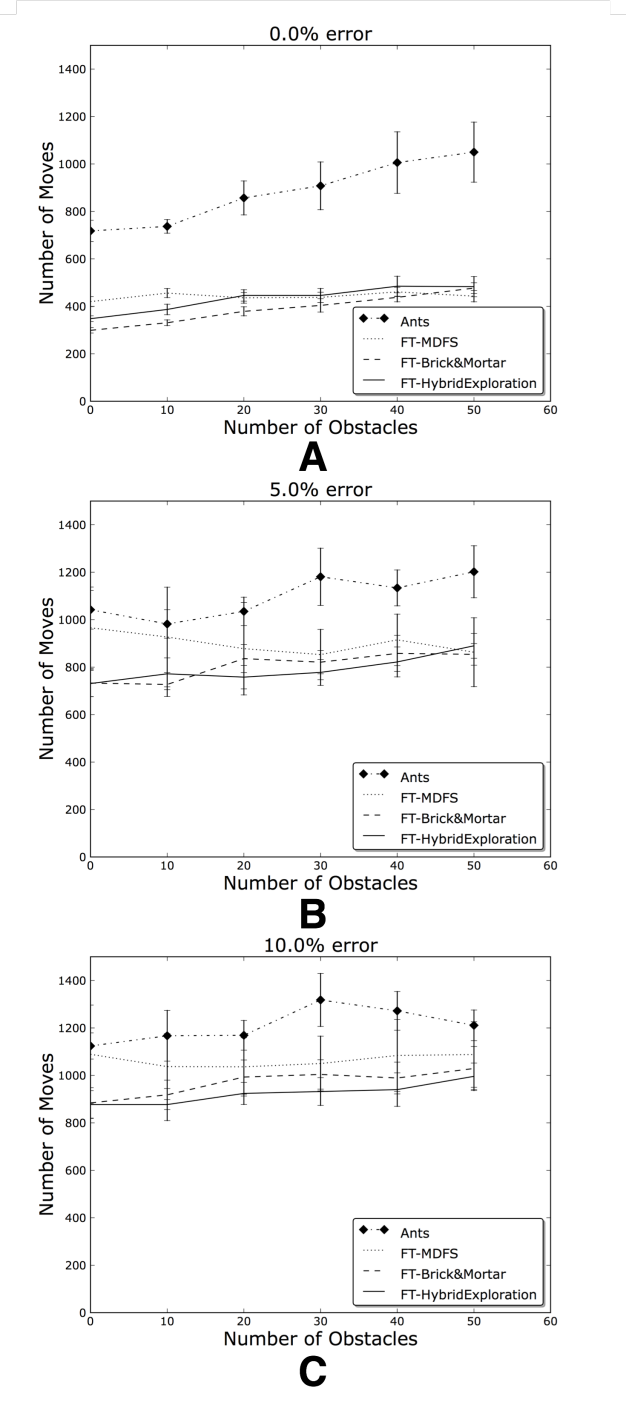

Figure 6.1: Effect of introducing detection errors (A: 0%, B: 5%, C: 10%) on the exploration time of Ants and FT algorithms, while changing the number of obstacles.

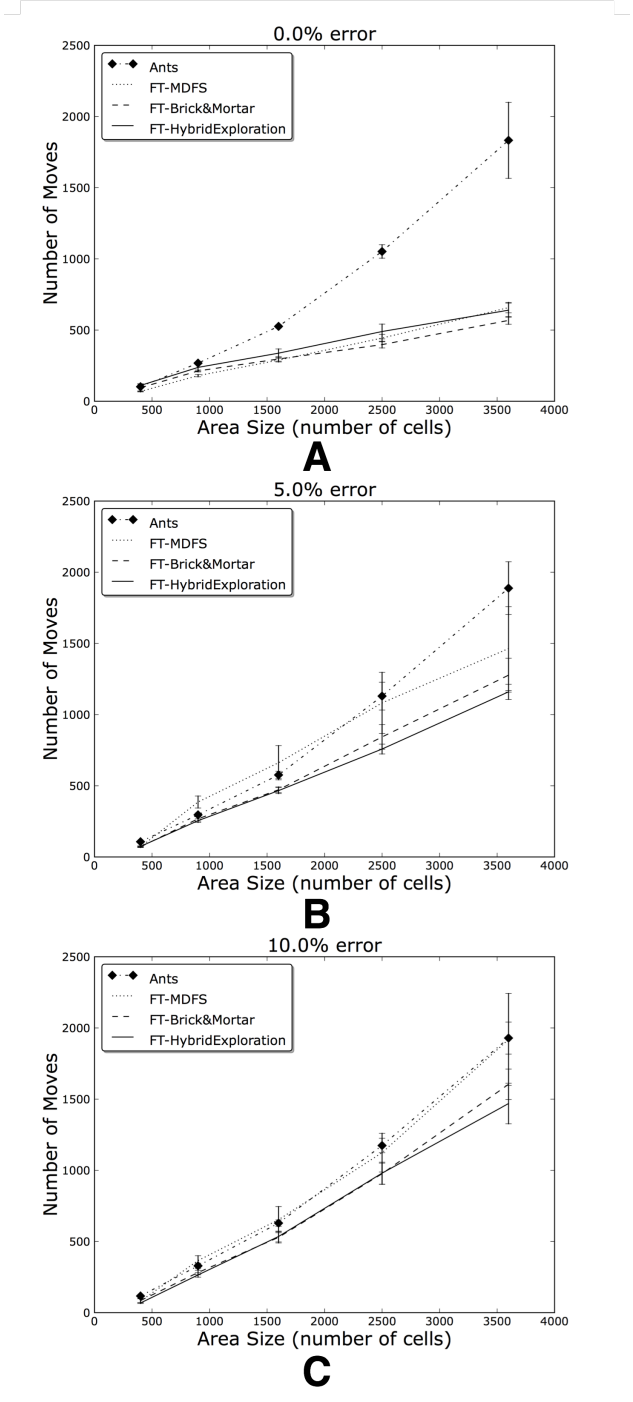

Figure 6.2: Effect of introducing detection errors (A: 0%, B: 5%, C: 10%) on the exploration time of Ants and FT algorithms, while changing the size of the area.

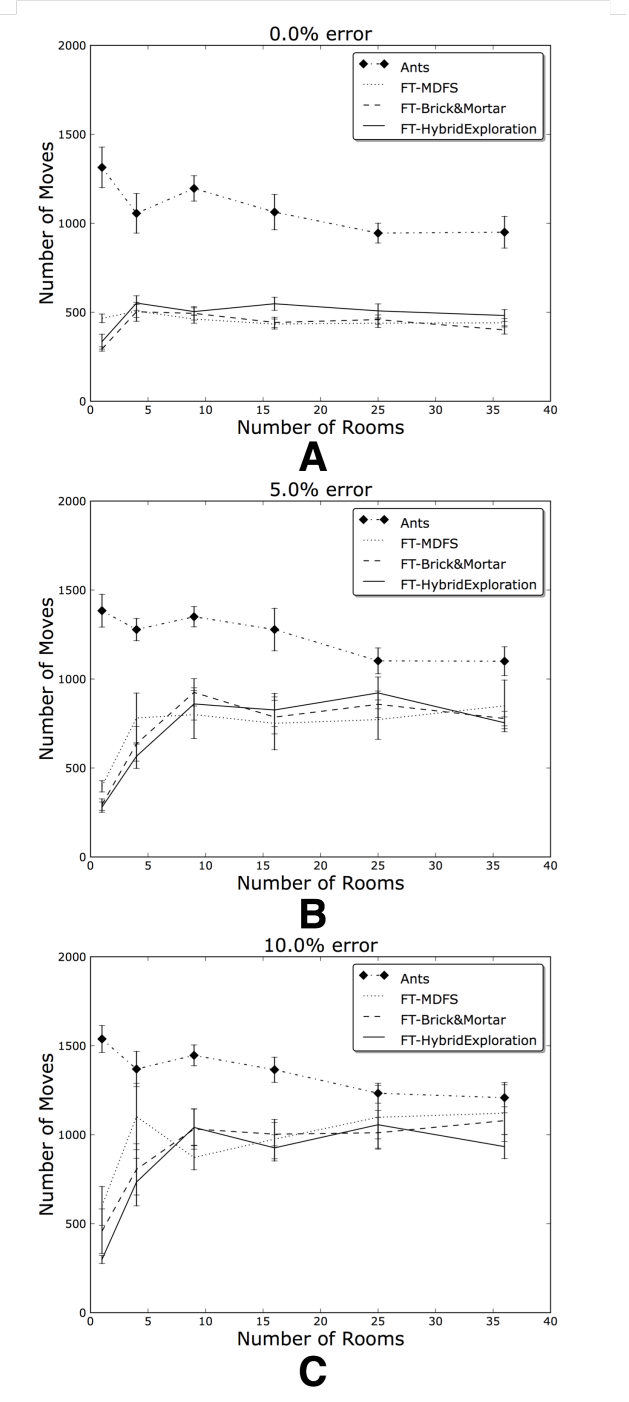

Figure 6.3: Effect of introducing detection errors (A: 0%, B: 5%, C: 10%) on the exploration time of Ants and FT algorithms, while changing the number of rooms.

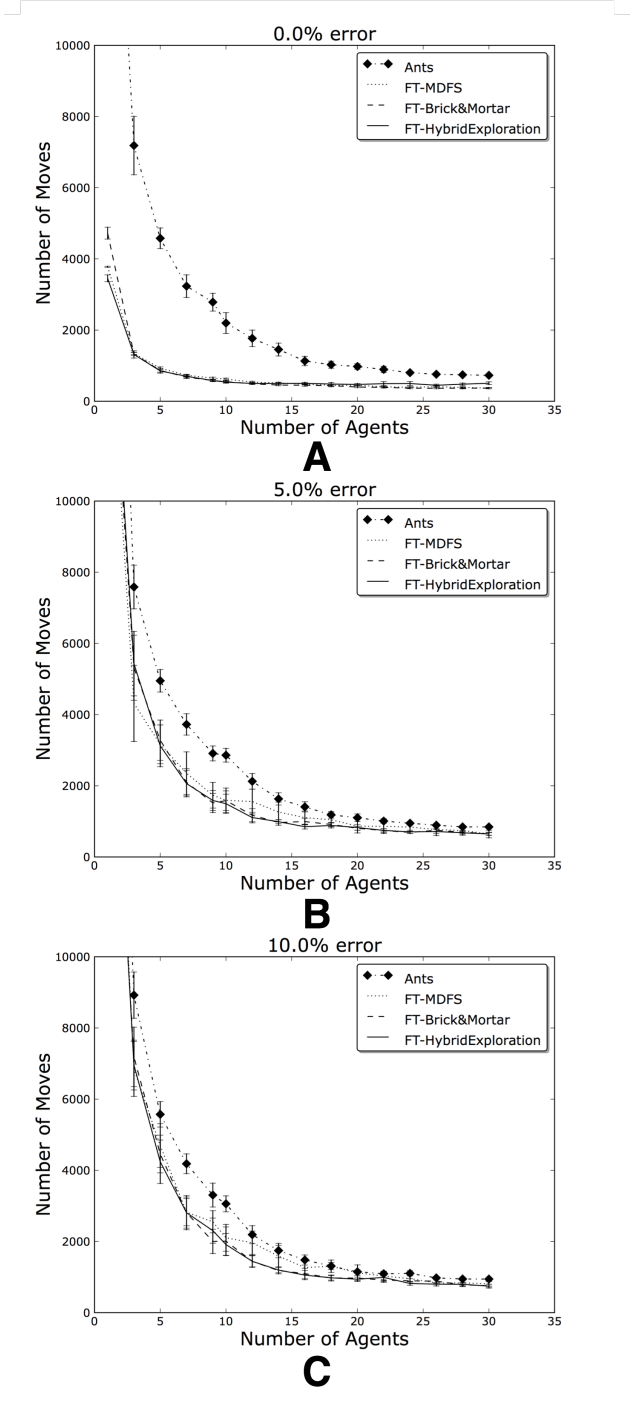

Figure 6.4: Effect of introducing detection errors (A: 0%, B: 5%, C: 10%) on the exploration time of Ants and FT algorithms, while changing the number of agents.

Chapter 7

Conclusions

This chapter summarises the main contributions of this work, highlights its limitations and proposes directions for future work.

7.1 Summary of Contributions

The focus of this thesis was on distributed algorithms for multi-agent exploration of unknown areas. This section summarises our contributions in this area, and presents the main conclusions drawn from our investigation.

In Chapter 1, we motivated the need for multi-agent exploration of unknown areas, defined the assumptions of our model, and suggested the use of stigmergy to coordinate agents in tag-instrumented environments. We carefully formulated three research problems: 1) how to coordinate agents to achieve the *Exploration* and *Completion Objectives* as fast as possible; 2) how to coordinate agents to discover short evacuation routes early in the exploration process; and 3) how to deal with practical problems of agents localising tags and accurately moving towards them.

In Chapter 2, we reviewed previous work, and classified it based on whether agents assume prior knowledge of the area's map, and whether they can communicate directly, through a central server, or through an instrumented environment. The existing Ants algorithm, which does not assume prior knowledge of a map, and coordinates its agents through an instrumented environment, was identified as the most appropriate algorithm to compare our work with.

In Chapter 3, we proposed three novel exploration algorithms, namely Multiple

Depth First Search, Brick&Mortar and HybridExploration, and compared them with the existing Ants algorithm. Their main advantages over Ants are that they make more efficient use of agent resources, and they are capable of achieving the *Completion Objective*. A complexity analysis was carried out to provide lower and upper bounds on the time they take to achieve the *Completion Objective*. All algorithms are shown to do so in polynomial time. In Chapter 3, we also proposed two mechanisms for discovering evacuation routes between critical cells in the area and emergency exits. The first mechanism, Agent2Tag-ERD relies only on agent-to-tag communication, whereas the second, Tag2Tag-ERD also exploits inter-tag communication. These mechanisms can be easily integrated with one of the exploration algorithms, like Ants, MDFS, Brick&Mortar and HybridExploration.

In Chapter 4, we empirically evaluated the performance of the proposed and existing exploration algorithms in a simulation environment. We investigated the effect of varying a number of simulation parameters, such as the number of agents, the size and layout of the area, the number of rooms and the number of obstacles. The main conclusions drawn from the simulation results are:

- The proposed exploration algorithms (MDFS, Brick&Mortar and HybridExploration) are consistently shown to be twice to 5 times as fast as the existing Ants algorithm in terms of exploration time. Hence, they are more efficient in achieving the Exploration Objective, and faster in detecting hazards and victims in the area. In addition, unlike Ants, they are able to achieve the Completion Objective, which means that they are locally aware of when the entire area has been visited.

- As expected, the exploration and visiting times of all algorithms increase with the area size and decrease with the number of agents. However, the effect of varying the number of rooms and obstacles is not as clear; it depends on the algorithm and on the area type (Series, Collapsed, Office).

- The relative performance of MDFS and Brick&Mortar varies depending on the simulation scenario. However, MDFS and Brick&Mortar consistently exhibit special strengths and weaknesses. On the one hand, MDFS is better than Brick&Mortar in dealing with loops formed around obstacles, or around rooms with collapsed walls.

This is why MDFS has lower exploration and visiting times than Brick&Mortar in the Collapsed scenario. On the other hand, in the absence of loops, Brick&Mortar typically exhibits lower visiting times than MDFS. The reason is that Brick&Mortar agents typically mark cells as *visited* the first time they traverse them, so that they do not need to revisit them. MDFS agents, however, typically traverse cells more than once, especially in the Series scenario, where agents cannot easily disperse in the area.

- The HybridExploration algorithm outperforms the other exploration algorithms in terms of exploration and visiting times, in the majority of simulated scenarios. The reason is that it combines the strengths of all the other algorithms. First, like MDFS, it adopts a depth-first-search approach to resolve loops around obstacles, but using virtual agents (messages) instead of physical agents. Second, like Brick&Mortar, physical agents running HybridExploration typically traverse cells once before they can mark them as *visited*. Third, in the presence of obstacles, the physical agents of HybridExploration use the Ants approach to escape from loops without resolving them. Hence, whereas Brick&Mortar agents spend a lot of time resolving loops, HybridExploration agents escape from loops as soon as possible, and leave the tedious task of loop resolution to virtual agents.

In Chapter 4, we also empirically evaluated the performance of the proposed evacuation route discovery mechanisms in a simulation environment. The main conclusions drawn from the simulation results are:

- The choice of exploration algorithm largely affects the time when victims are found and evacuation routes are first discovered. For example, in areas without obstacles, where Brick&Mortar tends to be more efficient than MDFS and Ants in terms of exploration time, it is also faster in discovering evacuation routes.

- Without inter-tag communication (Agent2Tag-ERD), the choice of the exploration algorithm also affects the length of discovered evacuation paths. Algorithms that tend to revisit the same cells multiple times (Ants and MDFS) gradually discover

shorter evacuation paths than faster exploration algorithms (Brick&Mortar) that avoid going back to the same cells.

- With inter-tag communication (Tag2Tag-ERD), all exploration algorithms have the same performance both in terms of the average and worst-case evacuation paths.
- For a given exploration algorithm, Agent2Tag-ERD yields longer evacuation paths than Tag2Tag-ERD. The benefits of inter-tag communication become more obvious (up to 50%) when one considers the length of the longest path to remote victims.
- The quality of evacuation routes depends on the topological features of the area. The dependence is higher on the area size, the layout of rooms and the number of exits, and lower on the number or rooms and the number of obstacles in the rooms.

In Chapter 5, we discussed our experiences of building a real system consisting of a Surveyor SRV-1 robot and Tmote Sky sensors running the Contiki OS. We discussed challenges in trying to implement the exploration algorithms on our testbed. To address these challenges, we proposed practical solutions that allow a mobile agent to: (i) detect, identify and localise fixed sensors deployed in its vicinity; and (ii) accurately move towards a carefully selected fixed sensor. By testing these techniques in a real testbed, we derived realistic models of detection, localisation and navigation errors, and investigated how they affect the performance of the exploration algorithms. We concluded that 1) localisation errors, although non negligible, have almost no impact on the performance of the algorithms; 2) detection errors significantly compromise the performance of MDFS, Brick&Mortar and HybridExploration, whereas they hardly affect the simpler and more robust Ants algorithm; and 3) navigation errors are negligible and can be completely ignored. Hence, although our proposed exploration algorithms were shown to outperform Ants in idealised conditions, their performance is severely compromised in the presence of sensor detection errors. More specifically, our simulation results show that they fail to achieve the *Exploration Objective* when detection errors are introduced.

In Chapter 6, we addressed the aforementioned problem, by designing FT versions of the proposed exploration algorithms. The resulting FT-MDFS, FT-Brick&Mortar and

FT-HybridExploration algorithms always achieve the *Exploration Objective*, although they are still unable to achieve the *Completion Objective*. We extensively tested the FT algorithms in a variety of scenarios (the same ones used for the evaluation of the original algorithms in Chapter 4), with and without tag detection errors. In the absence of tag detection errors, the FT algorithms behave similarly as the original exploration algorithms, and consistently outperform Ants in terms of exploration time. In the presence of tag detection errors, the FT algorithms still outperform Ants, but their relative advantage gradually fades out as we increase the tag detection error.

To summarise, during the planning and execution of this research project, we learned that the proposed approach is not yet mature to let agents be deployed directly in an emergency scenario, because of all the technical difficulties and the limitations involved (detailed in Section 7.2). However, we hope to have been able to add our modest contribution to the research field of autonomous exploration of unknown areas, and that the proposed algorithms could be used in the future to coordinate a swarm of real robots. There is no clear winner among the proposed approaches (MDFS, Brick&Mortar, HybridExploration) in terms of exploration time, since each of them has a particular scenario (also included for this reason in the tests) in which performs particularly well. For this reason, we think that the human factor should still be important in choosing one of them according to the estimated characteristic of the area to explore. If this should not be possible, however, then probably HybridExploration might be the best candidate, since its performance are good in all scenarios. We believe that the use of algorithms for evacuation routes discovery should also be a very important part of every system for the autonomous exploration of an area, and we showed how even very simple approaches could achieve very important results in this direction. During the deployment of the real world experiments, we learned how an apparently trivial measurement error (detection) could yield very severe disruption, and we hope to have amended the algorithms in order to avoid this kind of risk. Finally, we believe that using a camera to determine the position of an agent, detect obstacles and identify tags should be also considered for all future applications of the exploration algorithms, even if combined with other kind of sensors (ultrasound, infrared) to augment the sensing capabilities of the robots.

7.2 Limitations of Current Work

The work presented in this thesis is mainly focused on algorithmic aspects of area exploration by a team of mobile agents. To deploy the proposed system using real robots in an emergency scenario, however, one would first need to overcome some limitations. This section summarises these limitations, which are mainly related to assumptions about the exploration area, as well as the communication and sensing capabilities of agents.

Assumptions about the area: The exploration area is assumed to be divided into a grid of square cells, and tags are assumed to be placed accurately in the middle of cells. When a real robot was used in the experiments presented in Chapter 5, tags were pre-deployed inside cells, since our focus was not on the realisation of a mechanism to let agents carry the tags and place them on the ground. Pre-deployment, however, is hardly realistic because of the difficulties involved in maintaining tags in place during a long time, such as battery replacement cycles, high deployment costs and failure detection. An alternative would be to assume that agents can dynamically deploy tags during the exploration; this approach, however, is hard to implement because of agent odometry errors, which are more pronounced when agents move on uneven surfaces. Hence, it is hard to ensure that agents place tags accurately in the middle of square cells in a grid-like fashion, especially in emergency scenarios such as collapsed buildings.

Another limiting assumption about the area is that we considered a cell to be a wall, even if it is partially occupied by an obstacle. In practice, if a small part of the cell is occupied by an obstacle, agents can still traverse this cell. Also we assume that if two cells are not walls, there is easy access from one to another. In practice, this could not be the case, because of a step that an agent cannot climb, or a particularly steep slope.

We also assumed that, once deployed, tags are not displaced from their original positions and continue to operate without faults until the completion of the exploration process. This is hardly realistic in an emergency scenario, where collapsing walls, humans or agents could move the deployed tags, or where hazards like fires or explosions could damage or even destroy them.

Assumptions about communication: Although the proposed algorithms do not

need long range wireless communication, we still assume that a message can always be exchanged between an agent and the tags in its current and surrounding cells. Furthermore, in HybridExploration we assume perfect communication between a tag and its four adjacent ones. This might not be the case in scenarios with high radio interference, or where the radio equipment in some of the tags might be malfunctioning.

In this work, we do not take advantage of wireless communication between agents, even when it is available, and it could be used to enhance agents' performance. For example, agents could exchange information about which part of the area they have recently been exploring, and where they intend to go next, in order to better disperse in the area and speed up exploration. Furthermore, we do not exploit agent-to-server communication even when an agent is within range of a central point of command-and-control. This last option might be used by agents to offload information about the area explored so far, in order to keep human responders updated while the exploration is still in progress.

Assumptions about agent sensing capabilities: We used tag detection and localisation mechanisms based on camera technologies, albeit these solutions may not operate well in emergency situations, for example when fumes are present in a room because of a fire or a leakage, or in rooms with particularly adverse light conditions.

Finally, in this work, we did not address problems related to obstacle detection and avoidance. We assumed that agents are able to detect and avoid hitting wall cells, agents and tags deployed on the ground.

7.3 Future Work

In the future, we would like to lift some of the limiting assumptions discussed in Section 7.2, and design robust algorithms that can be applied to a real emergency scenario.

First, we would like to lift our assumption that agents can easily move between two adjacent non-wall cells. In a real environment, an agent may be able to move from cell A to cell B, but not in the other direction, because of a steep slope, a step, or an obstacle between the two cells. We thus plan to devise algorithms that represent the area as a directed graph, where vertices correspond to cells and directed edges correspond to

feasible passages from one cell to another.

Second, it would be interesting to design exploration algorithms that do not assume a grid-like tag placement. As a first step, we would like to consider environments where tags are deployed in an area uniformly at random. As a next step, we plan to consider environments where the tag distribution is not uniform, and there exist void areas without any tags placed in them. If these areas are accessible by agents, the challenge is to explore them in the absence of tags; if these areas are inaccessible, the challenge is to efficiently move around them in order to reach other accessible parts of the area.

Third, we would like to extend our exploration algorithms with the ability to exploit agent-to-agent and agent-to-server communication. By combining these modes of communication with the existing agent-to-tag and tag-to-tag communication, we could potentially improve agent coordination and achieve lower exploration and visiting times. A research challenge here would be to determine what kind of information should be exchanged between two agents, when they are within communication range, or between an agent and the central server. There might also be a need to extend the state stored in tags, and how it is updated during the exploration process.

Fourth, an interesting direction for future work is to make exploration algorithms robust to dynamic changes in the topology of the tag network. Tags could dynamically fail because they run out of battery, or they are destroyed by a hazard. Tags could also be displaced by agents, humans or hazards. Communication links between adjacent tags are also expected to be intermittent in emergency scenarios. Hence, the topology and connectivity of the tag network is expected to change over time, and exploration algorithms must be devised to adapt to network dynamics.

Fifth, we would like to investigate the effects of using agents with limited amount of available energy. Agents could in this case run out of energy before the exploration has been completed, and contingency strategies should be taken in order to maximise the explored area before this happens (e.g., by spreading the agents as much as possible). Another strategy could be to stop agents with energy below a certain threshold, and make them stationary in order to act as radio repeaters for other agents in the area.

Finally, in certain emergency scenarios where camera technologies cannot be used

to detect and localise tags (because of fumes or adverse light conditions), alternative technologies must be considered. For example, it would be interesting to use RF time-of-flight ranging for determining the distance between two nodes (agent-to-agent, agent-to-tag or tag-to-tag distance). Once distances between pairs of nodes are determined, they could be used for node identification, localisation and navigation purposes. New models of detection, localization and navigation errors must be derived, and exploration algorithms must be designed that are robust to these errors.

In conclusion, in this thesis, we proposed novel algorithms for multi-agent exploration and discovery of evacuation routes and evaluated them extensively in a simulation environment. Using a real testbed, we derived realistic models of localisation, detection and navigation errors, and designed fault-tolerant exploration algorithms to deal with these errors. However, there are still many practical problems that need to be addressed before exploration algorithms are ready to implement in a real emergency scenario. In future work, assumptions about tag placement, communication and sensing capabilities of agents must be revisited, and new algorithms must be proposed that deal with irregular and dynamic tag networks, multiple and realistic modes of node communication, and alternative sensing modalities.

Appendix A

Generation of Test Scenarios

This Appendix contains the algorithms which have been used to generate the explorations scenarios in our simulation environment. The algorithms are in pseudocode, but can be executed by a Python interpreter with very few changes. A brief description of the single methods is provided below:

- **generateOfficeRooms**: given the number of rooms (rows by column) to be created, it generates a scenario of the type *office*.
- **generateCollapsedRooms**: given the number of rooms (rows by column) to be created, it generates a scenario of the type *collapsed*.
- **generateSeriesRooms**: given the number of rooms (rows by column) to be created, it generates a scenario of the type *series*.
- **generateLoops**: given the number of obstacles to be created, it adds them to the current scenario.

Figures A.1, A.2, A.3, A.4, A.5, A.6 represent sample areas that can be generated by those algorithms.

```
def generateOfficeRooms(roomsRows,roomsColumns):

  if roomsRows <=0 or roomsColumns <=1:
    return
  leftLimit=(columns/2)-2
  rightLimit=(columns/2)+2
```

```
corridors=(roomsRows/2)+1
roomHeight=((rows-(corridors*3))/roomsRows)-1
roomWidth=(columns-3)/roomsColumns

linePoint=-1
counter=0
lines=0
rowsDone=0

while True:
  if counter==0:
    linePoint+=4
  else:
    linePoint+=roomHeight
    lines+=1
  counter+=1
  if counter==3:
    counter=0

  for j in range(leftLimit):
    set(WALL,linePoint,j)
  for j in range(rightLimit+1,columns+1):
    set(WALL,linePoint,j)

  if counter!=0 and rowsDone<roomsRows:
    rowsDone += 1
    verticalLeft=leftLimit
    verticalRight=rightLimit
    for j in range(linePoint,linePoint+roomHeight):
      set(WALL,j,leftLimit)
      set(WALL,j,rightLimit)
    left=True
    for z in range(roomsColumns-2):
      if left:
        left=False
        verticalLeft-=roomWidth
        for j in range(linePoint,linePoint+roomHeight+1):
          set(WALL,j,verticalLeft)

      else:
        left=True
        verticalRight+=roomWidth
        for j in range(linePoint,linePoint+roomHeight+1):
          set(WALL,j,verticalRight)

  right=False;
```

```
    if counter==0 or counter==1:
      verticalLeft=leftLimit
      verticalRight=rightLimit
      for z in range(roomsColumns):
        if not right:
          right=True
          newValue = verticalLeft-random(1,roomWidth)
          if newValue<0:
            newValue=0
          set(UNEXPLORED,linePoint,newValue)
          verticalLeft-=roomWidth
        else:
          right=False
          newValue = verticalRight+random(1,roomWidth)
          if newValue>columns-1:
            newValue=columns-1
          set(UNEXPLORED,linePoint,newValue)
          verticalRight+=roomWidth

      if(lines== roomsRows):
        break

def generateCollapsedRooms(roomsRows,roomsColumns):
  if roomsRows<=0 or roomsColumns<=0:
    return
  rStep=rows/roomsRows
  cStep=columns/roomsColumns
  rAcc=rStep
  cAcc=cStep
  count=0
  flag = False

  while count < roomsRows-1:
    rand = 1 + random(0,cStep-2)
    acc = 1
    countGaps = 0
    for i in range(0,columns+1):
      if i != rand:
        set(WALL,rAcc,i)
      elif countGaps < roomsColumns - 1:
        set(UNEXPLORED,rAcc,i)
        rand = 1 + random(0,cStep-2) + (acc*cStep)
        acc += 1
        countGaps += 1
    rAcc += rStep
    count += 1
```

```
  count = 0
  while count < roomsColumns-1:
    rand = random(0,rStep-1)
    acc = 1
    countGaps = 0
    for i in range(0,rows):
      if i != rand:
        set(WALL,i,cAcc)
      elif countGaps < roomsRows - 1:
        set(UNEXPLORED,i,cAcc)
        rand = 1 + random(0,rStep-2) + (acc*rStep)
        acc += 1
        countGaps += 1
    cAcc += cStep
    count += 1

def generateSeriesRooms(roomsRows,roomsColumns):
  if roomsRows<=0 or roomsColumns<=0:
    return
  rStep=rows/roomsRows
  cStep=columns/roomsColumns
  rAcc=rStep
  cAcc=cStep
  count=0
  flag = False
  while count < roomsRows-1:
    rand = 1 + random(0,cStep-2)
    rand2 = 1 + random(0,cStep-2)
    for i in range(0,columns):
      set(WALL,rAcc,i)
    if flag:
      set(UNEXPLORED,rAcc,rand)
      flag = not flag
    else:
      set(UNEXPLORED,rAcc,rand2+((roomsColumns-1)*cStep))
      flag = not flag
    rAcc += rStep
    count += 1
  count = 0
  while count < roomsColumns-1:
    rand = random(0,rStep-1)
    acc = 1
    countGaps = 0
    for i in range(0,rows):
      if i != rand:
        set(WALL,i,cAcc)
```

```
        elif countGaps < roomsRows - 1:
          set(UNEXPLORED,i,cAcc)
          rand = 1 + random(0,rStep-2) + (acc*rStep)
          acc += 1
          countGaps += 1
      cAcc += cStep
      count += 1

def generateLoops(loops):
  if loops<=0:
    return
  maxR=rows
  maxC=columns
  candidates = []
  for row in range(maxR):
    for col in range(maxC):
      if cell(row,col).state == WALL:
        continue
      free = True
      for r in range(3):
        for c in range(3):
          temp=cell(row-1+r,col-1+c)
          if temp==None or temp.state==WALL:
            free=False
            break
          for cell in candidates:
            if cell.row==temp.row and cell.col==temp.col:
              free=False
              break
          if not free:
            break
      if free:
        candidates.append(cell(row,col))
  for i in range(loops):
    if len(candidates)==0:
      break
    index = random(len(candidates))
    c = candidates[index]
    set(WALL,c.row,c.col)
    candidates.remove(c)
```

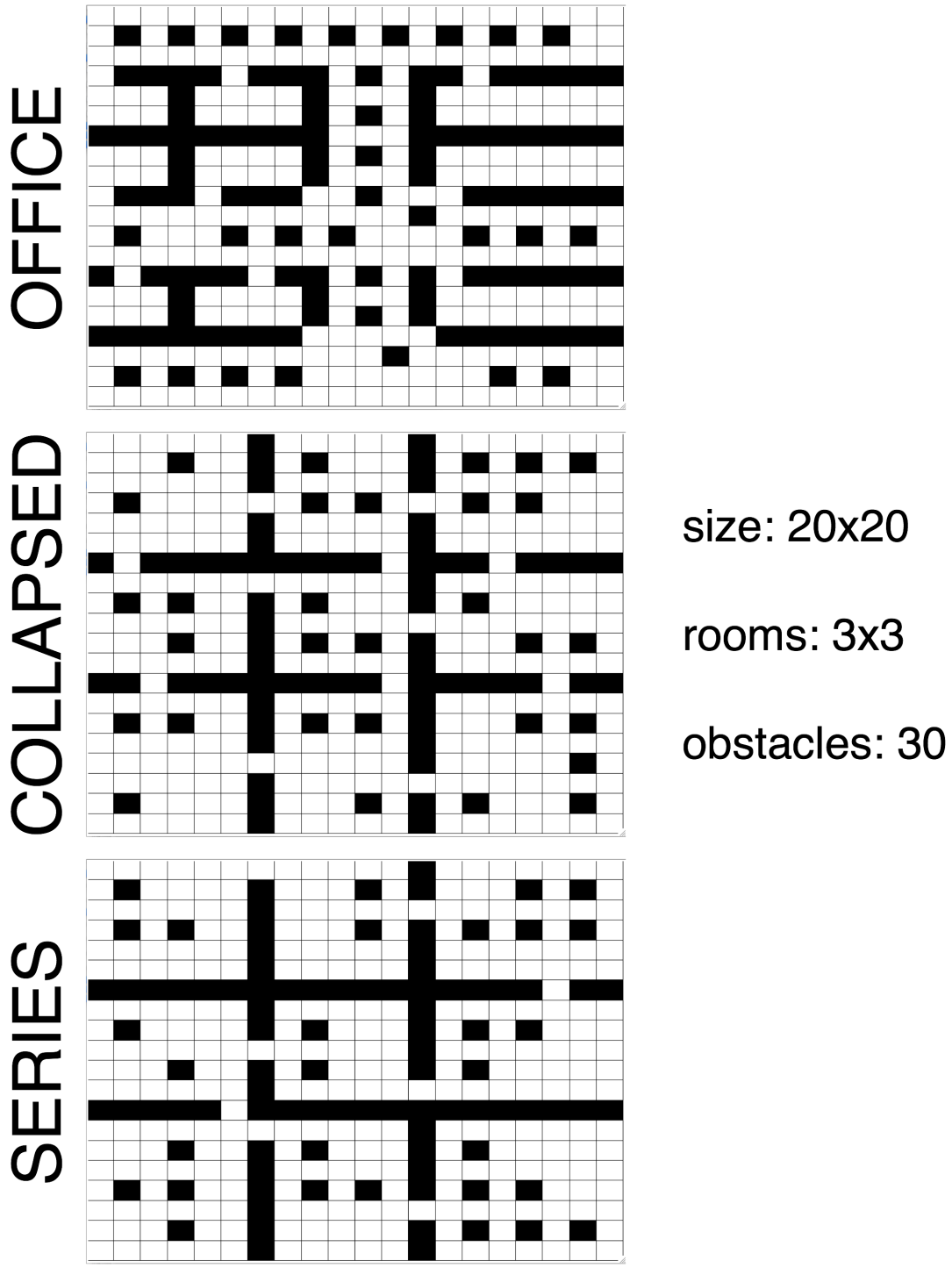

Figure A.1: Examples for the Office, Collapsed and Series scenarios with 20x20 cells size, 3x3 rooms and 30 obstacles.

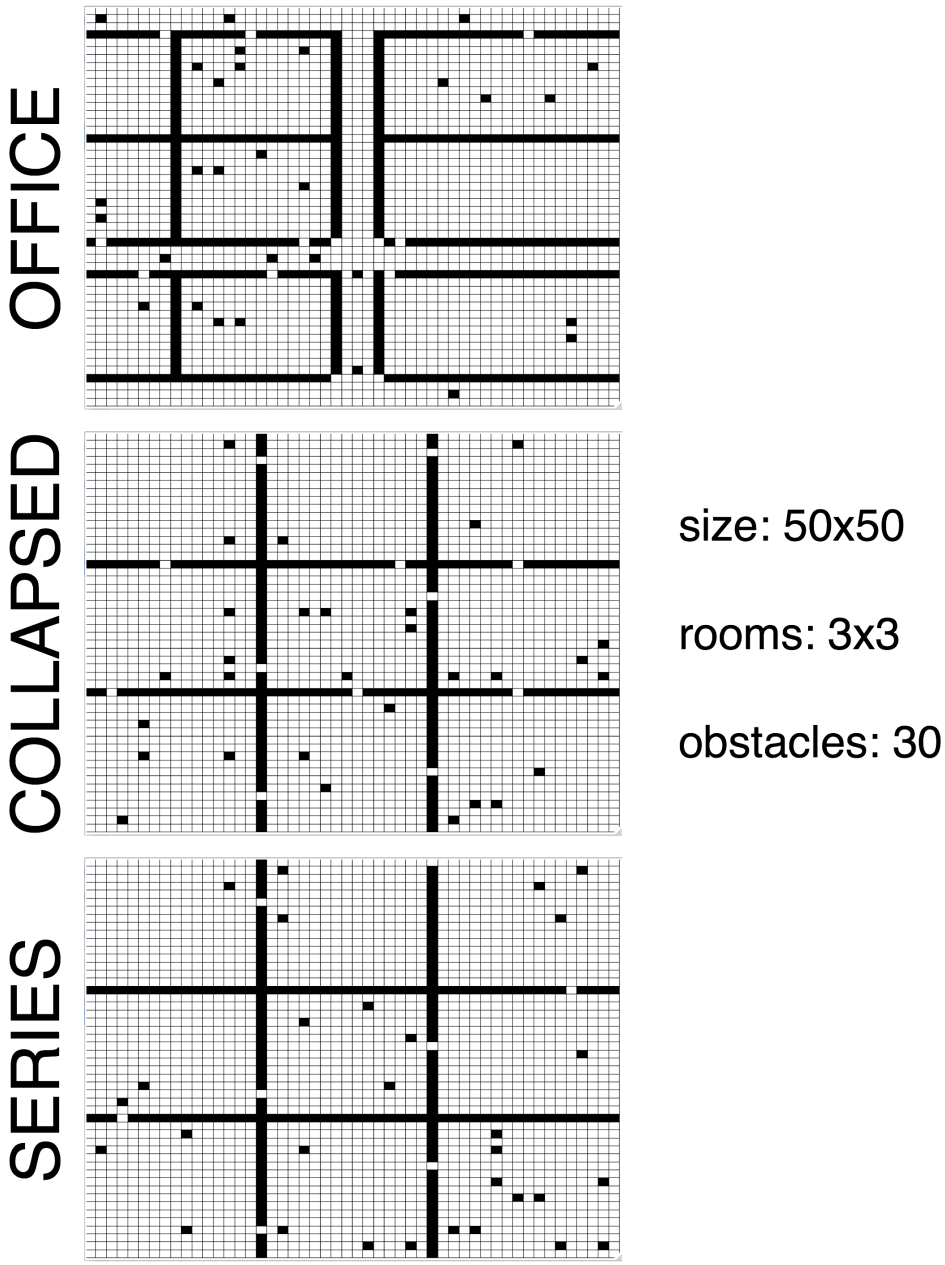

Figure A.2: Examples for the Office, Collapsed and Series scenarios with 50x50 cells size, 3x3 rooms and 30 obstacles.

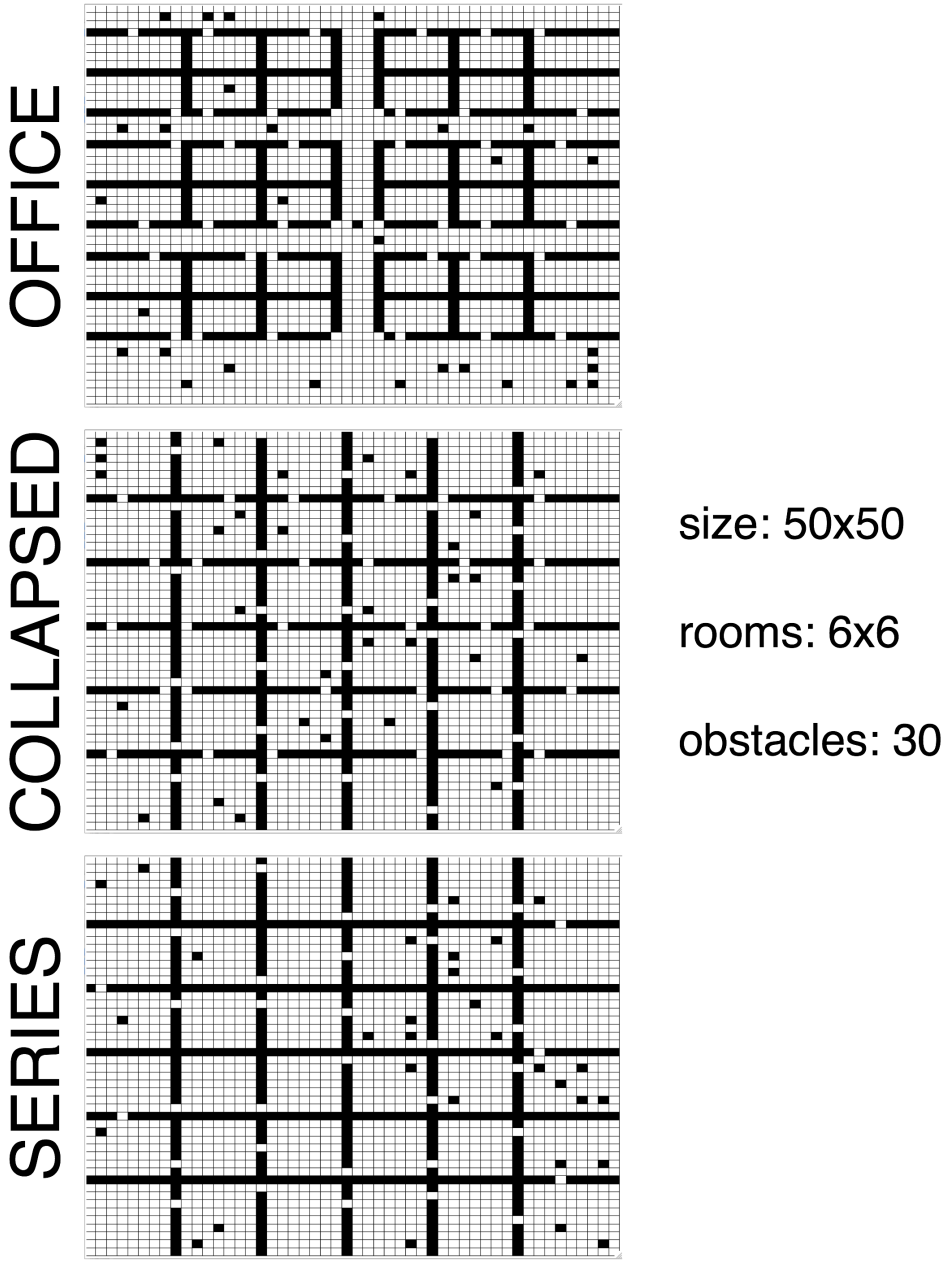

Figure A.3: Examples for the Office, Collapsed and Series scenarios with 50x50 cells size, 6x6 rooms and 30 obstacles.

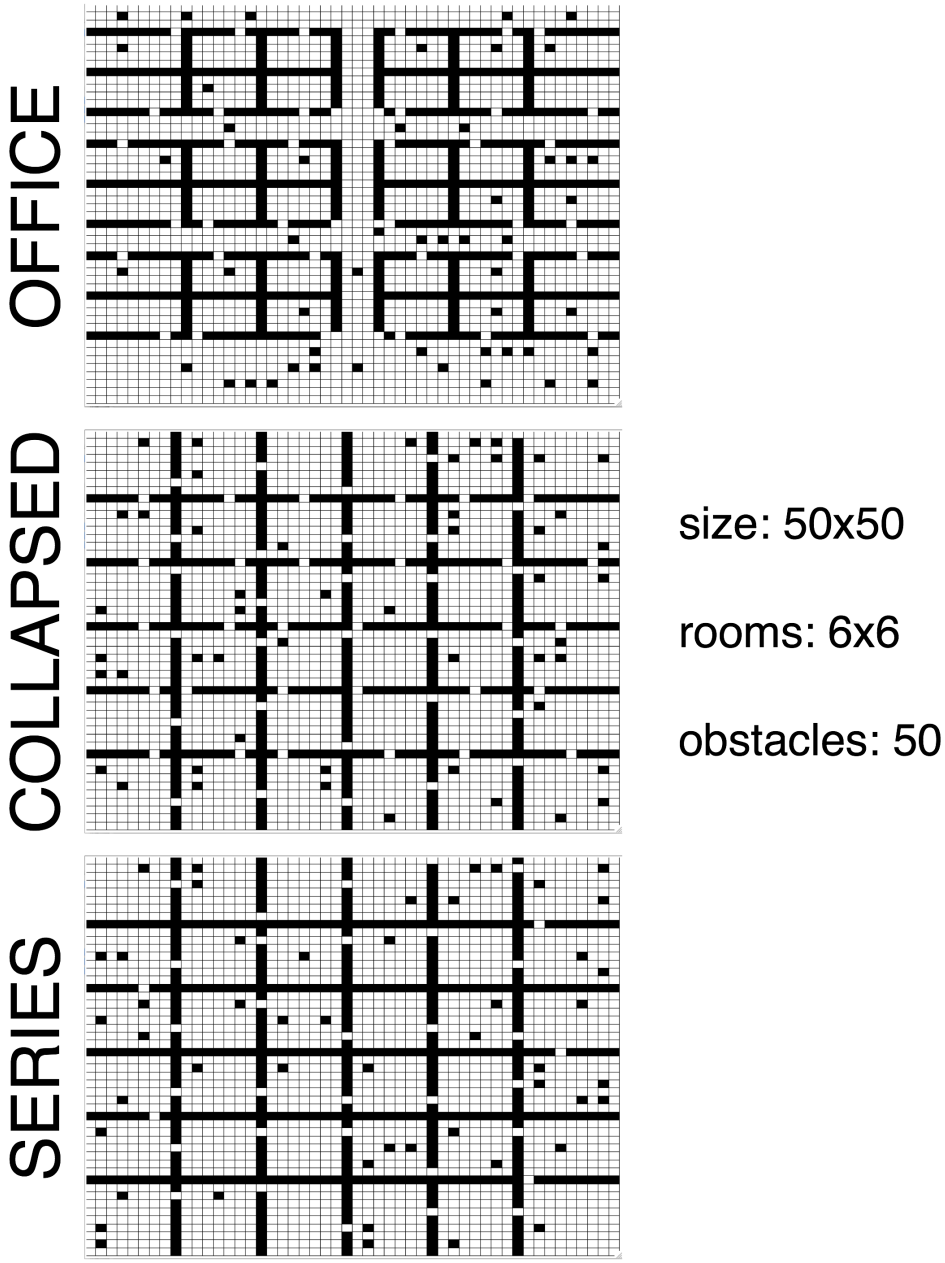

Figure A.4: Examples for the Office, Collapsed and Series scenarios with 50x50 cells size, 6x6 rooms and 50 obstacles.

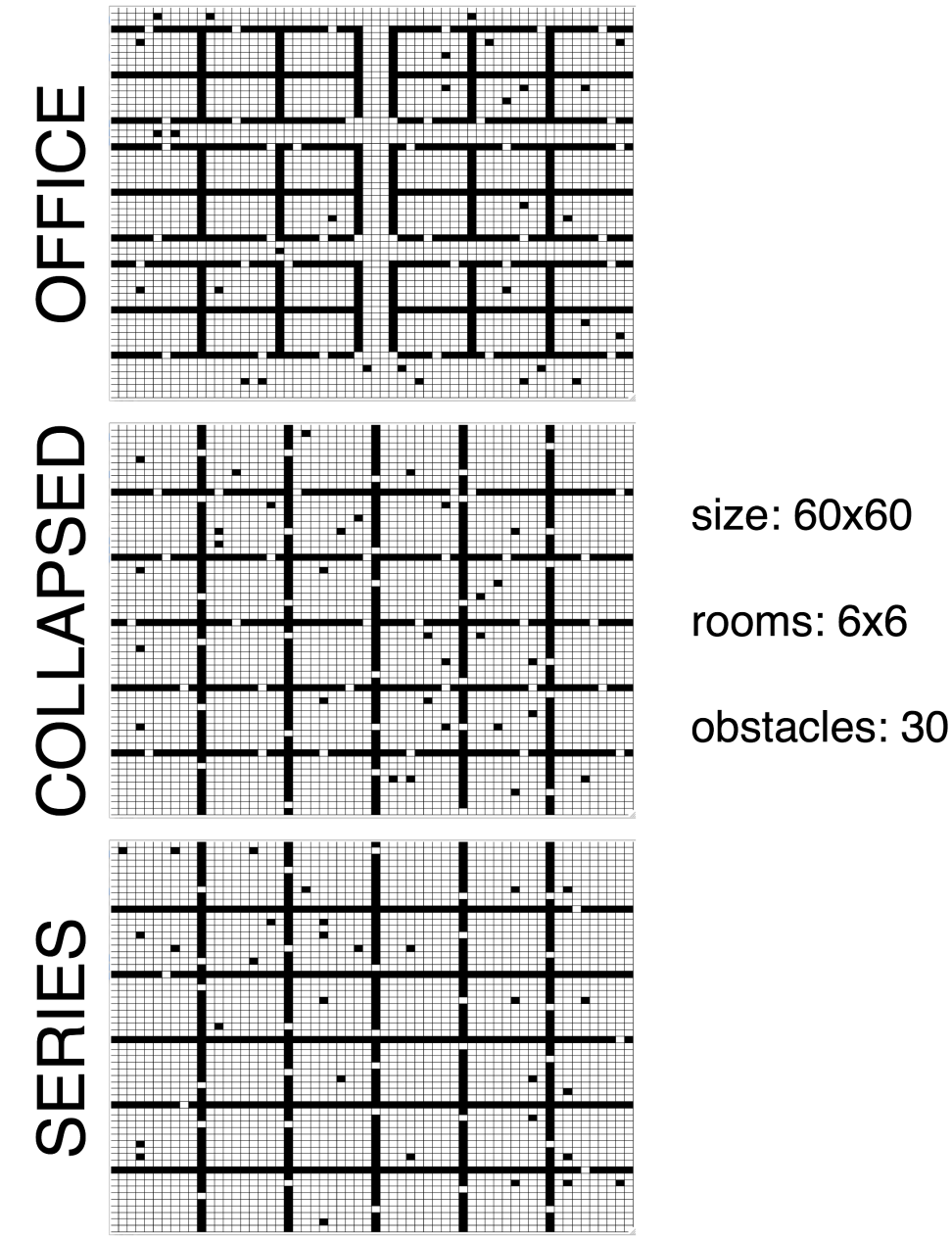

Figure A.5: Examples for the Office, Collapsed and Series scenarios with 60x60 cells size, 6x6 rooms and 30 obstacles.

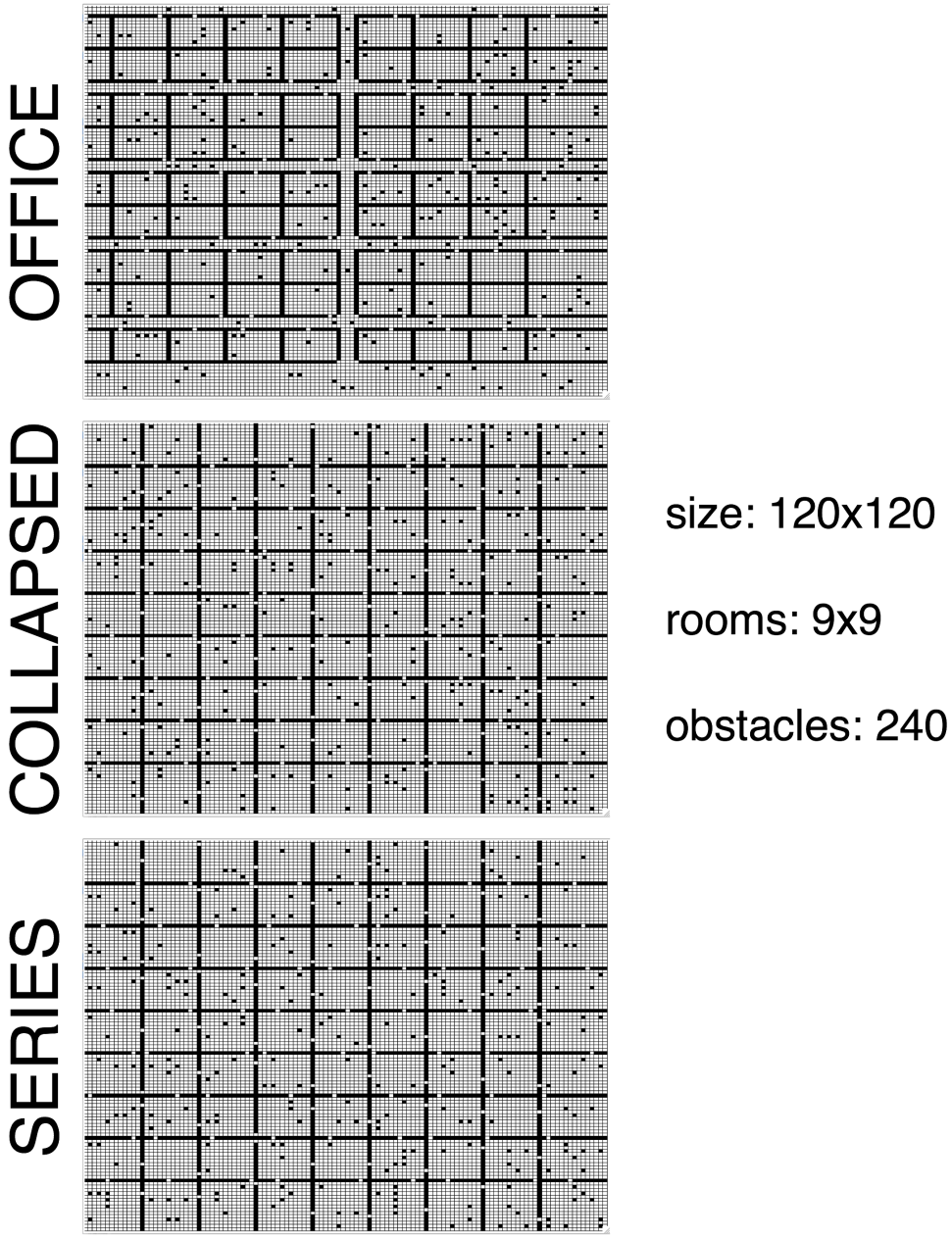

Figure A.6: Examples for the Office, Collapsed and Series scenarios with 120x120 cells size, 9x9 rooms and 240 obstacles.

Bibliography

Agmon, N., Hazon, N., and Kaminka, G. A. (2006). Constructing Spanning Trees for Efficient Multi-Robot Coverage, booktitle = ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation. pages 1698–1703. IEEE Press.

Aguayo, D., Bricket, J., Biswas, S., Judd, G., and Morris, R. (2004). Link-level Measurements from an 802.11b Mesh Network. In *SIGCOMM04: Proceedings of the 2004 ACM Conference of the Special Interest Group on Data Communication*, pages 121–132. ACM Press.

Albers, S., Kursawe, K., and Schuierer, S. (1999). Exploring unknown environments with obstacles. In *Proceedings of the tenth ACM-SIAM symposium on Discrete algorithms*, pages 842–843. ACM Press.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous Localization and Mapping: Part II. *Robotics and Automation Magazine IEEE*, 13(3):108–117.

Batalin, M., Sukhatme, G., and Hattig, M. (2004). Mobile Robot Navigation using a Sensor Network. In *ICRA04: Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 636–642. IEEE Press.

Batalin, M. A. and Sukhatme, G. S. (2003a). Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. In *Proceedings of the International Workshop on Information Processing in Sensor Networks*, pages 376–391. ACM.

Batalin, M. A. and Sukhatme, G. S. (2003b). Efficient Exploration Without Localization . In *ICRA03: Proceedings of 2003 IEEE International Conference on Robotics and Automation*, pages 2714–2719. IEEE Press.

Batalin, M. A. and Sukhatme, G. S. (2005). The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment. In *ICRA05: Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pages 3478–3485. IEEE Press.

Batalin, M. A. and Sukhatme, G. S. (2007). The design and Analysis of an Efficient Local Algorithm for Coverage and Exploration. *IEEE Transactions on Robotics*, 23(4):661–675.

Batalin, M. A. and Sukhatme, S. G. (2002). Spreading Out: A Local Approach to Multi-robot Coverage. In *DARS02: Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems*, pages 373–382.

Bhattacharya, S., Atay, N., Alankus, G., Lu, C., Bayazit, B., and Roman, G.-C. (2005). Roadmap Query for Sensor Network Assisted Navigation in Dynamic Environments. Technical Report Technical Report WUCSE-2005-41, Washington University in St. Louis, Department of Computer Science and Engineering.

Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative Multi-Robot Exploration. In *ICRA00: Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 476–481. IEEE Press.

Castle, R. O., Gawley, D. J., Klein, G., and Murray, D. W. (2007). Video-rate recognition and localization for wearable cameras. In *BMVC07: Proceedings of the 2007 British Machine Vision Conference*, pages 1100–1109.

Cerpa, A., Busek, N., and Estrin, D. (2003). SCALE: A tool for Simple Connectivity Assessment in Lossy Environments. Technical Report Technical Report CENS-21, UCLA.

Choset, H. (2001). Coverage for robotics - A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:9608–9617.

Choset, H. and Pignon, P. (1997). Coverage path planning: The boustrophedon cellular

decomposition. In *FSN97: Proceedings of 1997 International Conference on Field and Service Robotics*.

Dasgupta, P. and Cheng, K. (2009). Distributed coverage of unknown environments using multi-robot swarms with memory and communication constraints. Technical Report Technical Report No. cst-2009-1, University of Nebraska, Omaha.

Davison, A. J., Cid, Y. G., and Kita, N. (2004). Real-Time 3D SLAM with Wide-Angle Vision. In *IAV04: Proceedings of the 2004 IFAC/EURON Symposium on Intelligent Autonomous Vehicles*.

Djekoune, A. and Achour, K. (2000). Visual Guidance Control Based on the Hough Transform. In *Proceedings IEEE Intelligent Vehicles Symposium 2000*, pages 614–619. IEEE Press.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localization and Mapping: Part I. *Robotics and Automation Magazine IEEE*, 13(2):99–110.

Ferranti, E., Trigoni, N., and Levene, M. (2007). Brick&Mortar: An On-Line Multi-Agent Exploration Algorithm. In *ICRA07: Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 761–767. IEEE Press.

Ferranti, E., Trigoni, N., and Levene, M. (2008). HybridExploration: a Distributed Approach to Terrain Exploration using Mobile and Fixed Sensor Nodes. In *IROS08: Proceedings of the 2008 IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems*, pages 1280–1286. IEEE Press.

Foka, A. and Trahanias, P. (2003). Predictive control of robot velocity to avoid obstacles in dynamic environments. In *IROS03: Proceedings of the 2003 IEEE International Conference of Intelligent Robots and Systems*, pages 370–375.

Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., and Wicker, S. (2002). Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Technical Report Technical Report CSD-TR 02-0013, UCLA.

Ghidary, S. S., Tani, T., Takamori, T., and Hattori, M. (1999). A new Home Robot Positioning System (HRPS) using IR switched multi ultrasonic sensors. In *Proceedings of the IEEE SMC Conference*. IEEE Press.

Glad, A., Simonin, O., Buffet, O., and Charpillet, F. (2008). Theoretical study of ant-based algorithms for multi-agent patrolling. In *ECAI08: Proceedings of the 2008 European Conference on Artificial Intelligence*. ECAI.

Glad, A., Simonin, O., Buffet, O., and Charpillet, F. (2010). Influence of different execution models on patrolling ant behaviors: from agents to robots. In *AAMAS10: Proceedings of the 2010 International Conference on Autonomous Agents and Multiagents Systems*. ACM.

Hazon, N. and Kaminka, G. A. (2005). Redundancy, Efficiency, and Robustness in Multi-Robot Coverage. In *ICRA05: Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pages 735–741. IEEE Press.

Hazon, N., Mieli, F., and Kaminka, G. A. (2006). Towards Robust On-line Multi-Robot Coverage. In *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 1710–1715. IEEE Press.

Horchler, A., Reeve, R., Webb, B., and Quinn, R. (2004). Robot phonotaxis in the wild: a biologically inspired approach to outdoor sound localization. *Advanced Robotics*, 18(8):801–816.

Howard, A., Parker, L. E., and Sukhatme, G. S. (2006). Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection. *The International Journal of Robotics Research*, 25(5–6):431–447.

Icking, C., Kamphans, T., Klein, R., and Langetepe, E. (2005). Exploring Simple Grid Polygons. In *COCOON05: Proceedings of the 2005 Annual International Conference of Computing and Combinatorics*, pages 524–533. Springer Berlin / Heidelberg.

Kataoka, S. and Atagi, K. (1997). Preventing IR interference between infrared waves

emitted by high-frequency fluorescent lighting systems and infrared remote controls. *IEEE transactions on industry applications*, 33(1):239–245.

Kelly, I. and Martinoli, A. (2004). A scalable, on-board localisation and communication system for indoor multi-robot experiments. *Sensor Review*, 24(2):167–179.

Kirchner, N. and Furukawa, T. (2005). Infrared Localisation for Indoor UAVs. In *ICST05: Proceedings of the 2005 International Conference on Sensing Technology*, pages 60–65.

Kleiner, A., Prediger, J., and Nebel, B. (2006). RFID Technology-based Exploration and SLAM for Search And Rescue. In *IROS06: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4054–4059. IEEE Press.

Koenig, S. and Liu, Y. (2001). Terrain Coverage with Ant Robots: a Simulation Study. In *AGENTS01: Proceedings of the 2001 ACM International Conference on Autonomous Agents*, pages 600–607. ACM Press.

Koenig, S., Szymanski, B., and Liu, Y. (2001). Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):41–76.

Kong, C. S., Peng, N. A., and Rekleitis, I. (2006). Distributed Coverage with Multi-Robot System. In *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 2423–2429. IEEE Press.

Kotz, D., Newport, C., and C., E. (2003). The mistaken axioms of wireless-network research. Technical Report Technical Report TR2003-467, Dartmouth College Computer Science.

Kotz, D., Newport, C., Gray, R. S., Liu, J., Yuan, Y., and Elliott, C. (2004). Experimental Evaluation of Wireless Simulation Assumptions. Technical Report Technical Report TR2004-507, Dartmouth College Computer Science.

Lee, C. Y. (1961). An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, EC(10):346–365.

Li, Q., De Rosa, M., and Rus, D. (2003). Distributed Algorithms for Guiding Navigation across a Sensor Network. In *Mobicom 2003: Proceedings of 2003 ACM International Conference on Mobile Computing and Networking*, pages 313–325. ACM.

Li, Q. and Rus, D. (2005). Navigation protocols in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):3–35.

Lim, J. H. and Leonard, J. J. (2000). Mobile Robot Relocation from Echolocation Constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1035–1041.

Mullen, R., Monekosso, D., Barman, S., and Remagnino, P. (2009). A review of ant algorithms. *Expert Systems with Applications*, 36(6):113–126.

O'Hara, K. J. and Balch, T. R. (2004a). Distributed Path Planning for Robots in Dynamic Environments Using A pervasive Embedded Network. In *AAMAS04: Proceedings of 2004 International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM.

O'Hara, K. J. and Balch, T. R. (2004b). Pervasive Embedded Networks for Supporting Multi-Robot Activities. In *Proceedings of the AAAI-04 workshop on Sensor Networks*. AAAI Press.

O'Hara, K. J., Bigio, V., Dodson, E. R., Irani, A. J., Walker, D. B., and Balch, T. R. (2005). Physical Path Planning Using the GNATs. In *ICRA05: Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pages 709–714. IEEE Press.

O'Hara, K. J., Bigio, V., Whitt, S., Walker, D., and Balch, T. R. (2006). Evaluation of a Large Scale Pervasive Embedded Network for Robot Path Planning. In *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 2072–2077. IEEE Press.

Palis, Rusin, Schumucker, Schneider, and Zavgorodniy (2005). *Legged Robot with Articulated Body in Locomotion Over Complex Terrain*, pages 321–328. Springer Berlin Heidelberg.

Park, J., Park, S., Kim, D., Cho, P., and Cho, K. (2003). Experiments on radio interference between wireless LAN and other radio devices on a 2.4 GHz ISM band. In *VTC03: Proceedings of the 2003 IEEE Semiannual Vehicular Technology Conference*, pages 1798–1801. IEEE Press.

Payton, D., Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone Robotics. *Autonomous Robots*, 11(3):319–324.

Rekleitis, I., Dudeck, G., and Milios, E. (1997). Multi-robot Exploration of an Unknown Environment, Efficiently Reducing the Odometry Error. In *IJCAI97: Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, pages 1340–1345. Morgan Kaufmann.

Rice, A. C. (2007). Dependable Systems for Sentient Computing. *PhD Thesis, Digital Technology Group, Computer Laboratory, University of Cambridge*.

Shiloni, A., Agmon, N., and Kaminka, G. A. (2009). Of robot ants and elephants. In *AAMAS09: Proceedings of the 2009 International Conference on Autonomous Agents and Multiagents Systems*. ACM.

Svennebring, J. and Koenig, S. (2004). Building Terrain-Covering Ant Robots: A Feasibility Study. *Autonomous Robots*, 16(3):313–332.

Theraulaz, G. and Bonabeau, E. (1999). A Brief History of Stigmergy. *Artificial Life*, 5(2):97–116.

USAR (2008). Robocup rescue. http://www.isd.mel.nist.gov/projects/USAR/competitions.htm.

Wagner, I. A., Altshuler, Y., Yanovski, V., and Bruckstein, A. M. (2008). Cooperative Cleaners: A Study in Ant Robotics. *International Journal of Robotics Research*, 27(1):127–151.

Wagner, I. A. and Bruckstein, A. M. (2001). From Ants to A(ge)nts: A Special Issue on Ant-Robotics. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):1–5.

Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M. (2000). MAC vs. PC - Determinism and Randomness as Complementary Approaches to Robotic Exploration of Continuous Unknown Domains. *International Journal of Robotics Research*, 19(1):12–31.

Yamauchi, B. (1998). Frontier-Based Exploration using Multiple Robots. In *AGENTS98: Proceedings of the 1998 ACM International Conference on Autonomous Agents*, pages 47–53. ACM Press.

Yanovski, V., Wagner, I. A., and Bruckstein, A. M. (2003). A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186.

Zhao, J. and Govindan, R. (2003). Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *SenSys03: Proceedings of the 2003 ACM Conference on Embedded Networked Sensor Systems*, pages 1–13. ACM Press.

Zheng, X., Jain, S., Koenig, S., and Kempe, D. (2005). Multi-Robot Forest Coverage. In *IROS05: Proceedings of the 2005 IEEE International Conference of Intelligent Robots and Systems*, pages 3852–3857. IEEE Press.