Multi-robot Systems: Modeling Swarm Dynamics and Designing Inspection Planning Algorithms

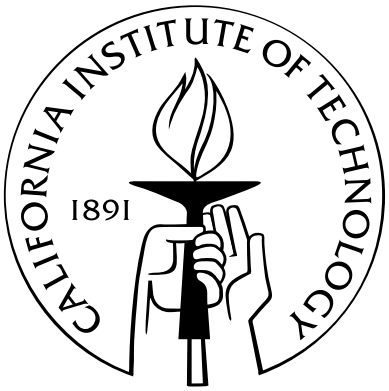
Thesis by

Kjerstin Irja Williams

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2006

(Defended May 5, 2006)

This work is dedicated to the memory of my grandmother, S. Ellen E. Englund. In so many ways, she showed me the truth of her favorite Bible verse: "If you have faith as a grain of mustard seed...nothing shall be impossible unto you." (Matthew 17:20)

Acknowledgements

The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" but "That's funny...."

– *Isaac Asimov*

I had the good fortune to work under the supervision of Alcherio Martinoli and Joel Burdick; both are exceptional scientists and mentors. I thank them for their generosity with their time and resources, their unflagging support, and their friendship. The Division of Engineering and Applied Science took good care of me along the way; I'm particularly grateful to Richard Murray, Pietro Perona, Yaser Abu-Mostafa, and Ali Hajimiri for their thoughtful advice throughout my graduate research. Discussions with Elon Rimon, Eric Klavins, and Howie Choset inspired me always to look for a more elegant solution, and then pose a more interesting question. My fellow graduate students helped make the process a lot more fun through thick and thin, and through collaboration and commiseration. There are too many to mention, but special thanks go to William Agassounon, Nikolaus Correll, Timothy Chung, Nathaniel Gray, Adam Hayes, Patricia Neil, Tracy Teal, Dirk Walther, and Lauren Webb. I was so lucky to have Maria Koeper looking out for me along the way – she takes such good care of us! Through ten years and three Caltech degrees, Bill Bing and the Caltech Bands have been such a bright spot in my life; it has been a privilege to play and sing with you.

I am especially grateful to Nora Portel, Amy Barr, AnnMarie Polsenberg Thomas, Suleyman Gokyigit, Kacie Shelton, and Nathan and Sharon Schara for their friendship and encouragement.

It would be impossible to adequately express my gratitude to my family, but I'll try. Dad, thank you for helping me fall in love with robotics and inspiring me to make a career out of that passion. Mom, thank you for teaching me to let my heart choose my goals; I'll use my head to find a way to get there. Jay, you're the best big brother a girl could ask for; thank you for being my friend and foil. To the Williams family, thank you for your love and support through all these Caltech years.

Finally, I want to thank my husband and dearest friend, my very own personal fountain of wisdom and wisecracks, Travis. I am blessed to have found you. (And so are the kitties.)

Abstract

For a variety of applications, the capability of simultaneous sensing and action in multiple locations that is inherent to multi-robot approaches offers potential advantages over single robot systems in robustness, efficiency, and application feasibility.

At the fully distributed and reactive end of the multi-robot system spectrum, I present mathematical modeling methodologies developed to predict and optimize a self-organized robotic swarm's performance for several tasks. These models allow us to better understand the relationship between agent and group behavior by capturing the dynamics of these highly stochastic, nonlinear, asynchronous systems at various levels of abstraction, in some cases even achieving mathematical tractability. The models deliver qualitatively and quantitatively correct predictions several orders of magnitude more quickly than an embodied simulator can. Swarm modeling lays the foundation for more generalized SI system design methodology by saving time, enabling generalization to different robotic platforms, and estimating optimal design and control parameters.

In considering more complex target tasks and behaviors, efficiency and completeness of execution may be of concern, and a swarm approach may not be appropriate. In such cases a more deliberative approach may be warranted. In that context, I introduce the multi-robot boundary coverage problem, in which a group of robots is required to completely inspect the boundary of all two-dimensional objects in a specified environment. To make such

a guarantee, I present a centralized planning approach that constructs a two-component abstraction of the problem: a graph representing the particular instance of the inspection task and a graph problem whose solution represents a complete plan for inspection. Using the building blocks of this approach, related inspection tasks that require the robotic system to adapt to a changes in team size and task assignment are also explored. The application of these planning methods to the case of long-term deployment for surveillance applications that require repetitive coverage is also discussed.

The recurring theme of this thesis is that we must look beyond implementation and validation of a particular system and ask how its design can contribute to the development of a more general design methodology.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In multi-robot systems, the opportunity for simultaneous sensing and action in multiple locations offers potential advantages over single robot systems in robustness, efficiency, and application feasibility. In a multi-robot system, unit redundancy may allow a team of robots to complete a task despite one or more robot failures, while simultaneous action may allow the team to finish the task more quickly or efficiently than a single robot. In the long term, especially for applications in which robots are subject to heavy use, damage, and replacement, multi-robot systems may also offer a cost advantage due to the potential for mass production. Tasks that allow for cooperation, exploiting this parallelism by allowing robots to share information or collaborate physically, are a natural choice for a multi-robot approach. Other tasks requiring capabilities or resources beyond the capacity of a single robot may also be well-suited to a multi-robot approach.

Though advances in multi-robot system formalism have been made, the design of multi-robot systems is still largely heuristic. A recent survey of these advances [1] observes that diverse multi-robot system design approaches may have a common underlying methodology and notes that "these similarities are encouraging because they suggest that, regardless of the details of the robots or tasks in use, the various authors are all studying a common, fundamental problem in autonomous coordination." The field lacks a general methodology

for multi-robot system design; indeed, most complex multi-robot task allocation problems are $\mathcal{NP}$-hard [2]. However, if an appropriate abstraction for a task can be found so that it might be posed as a known problem, we may exploit related algorithms, heuristics, and analytical tools. As we create a catalogue of such "building blocks" for a variety of multi-robot tasks, we may use them to design and implement incrementally more complex multi-robot systems while we gain a better understanding of methods for their formal classification and analysis.

The multi-robot systems explored in this thesis range from fully distributed, swarm-inspired systems to centrally supervised, deliberative systems, adding a diverse set of such building blocks to the field's ever-increasing collective understanding of multi-robot system design. At the distributed and reactive end of the multi-robot system spectrum, I present mathematical modeling methodologies developed to predict and optimize a robotic swarm's performance for several tasks. These swarm modeling building blocks help lay the foundation for more generalized SI system design methodology by saving time, enabling generalization to different robotic platforms, and estimating optimal design and control parameters. In considering more complex target tasks and behaviors, efficiency and completeness of execution may become a major concern, and a more deliberative approach may be warranted. In that context, I develop a graph-based framework to solve a multi-robot inspection problem. I then apply the building blocks of the graph framework to a related inspection problems abstracted in a similar way.

1.1 Distributed Robotic Systems

In the last several years, distributed control principles have been successfully applied to a series of case studies in collective robotics: aggregation [3, 4, 5] and segregation [6], foraging

[7, 8], collaborative stick pulling [9, 10], cooperative transportation [11, 12, 13, 14], flocking and navigation in formation [8, 15, 16], odor source localization [17, 18], cooperative mapping [19, 20], cooperative coverage [21] and soccer tournaments [22]. Each of these case studies used groups of robots or embodied simulated agents acting autonomously based on their own individual decisions. However, not all the architectures reported in these contributions were designed to control large numbers of robots. For instance, several approaches extensively exploit global communication capabilities [8, 12, 14, 20, 22], a characteristic that represents a bottleneck for the scalability of the collective system and may be, under certain environmental conditions, impossible. In other case studies, although the underlying principles of the control architecture were fully scalable, due to technical difficulties in experimentation with real robots, local explicit communication [17, 19] or specific environmental information (e.g., nest energy [7]) was obtained with global communication, often combined with global positioning systems. While global positioning systems, depending on their specific implementation (e.g., standard GPS or the system used by Billard et al. [19]), do not necessarily prevent the scalability of the collective system, they require an additional level of sophistication of the individual unit in order to process the information broadcast by the central reference device and are not always available in the target environment.

1.1.1 Swarm Robotic Systems

A possible paradigm for overcoming scalability issues and at the same time promoting robustness and individual simplicity is that proposed by a computational and behavioral metaphor for solving distributed problems called *Swarm Intelligence* (SI) [23, 24]. SI takes its inspiration from the biological examples provided by social insects [25] such as ants, termites, bees, and wasps and by swarming, flocking, herding, and shoaling phenomena in

vertebrates [26]. The abilities of such natural systems appear to transcend the abilities of the constituent individual agents. In most biological cases studied so far, robust and coordinated group behavior has been found to be mediated by nothing more than a small set of simple local interactions between individuals, and between individuals and the environment. The SI approach emphasizes self-organization, distributedness, parallelism, and exploitation of direct (peer-to-peer) or indirect (via the environment) local communication mechanisms among relatively simple agents.

The main advantages of the application of the SI approach to the control of multiple robots are fourfold:

1. Static scalability: The control architecture can be kept exactly the same whether thousands of robots are deployed or only a few are used.

2. Dynamic scalability: Robots can be added to or removed from the system on the fly; they may also have the ability to reallocate and redistribute themselves in a self-organized way.

3. Robustness: In a self-organized system, robustness, rather than efficiency, is promoted; a self-organized system will be robust to a priori unknown environmental and team changes not only through unit redundancy but also through a balance between exploratory and exploitative behavior. This balance requires a certain level of randomness be present in the system, such as noise in signals or sensors, or probabilistic "mistake" in individual behavior. A natural example of this system behavior is represented by the foraging strategy of ants [25]. It is well-known that recruitment processes based on trail-laying and -following mechanisms lead an ant colony to focus its current foraging power on a particular source of food (exploitation). What is,

perhaps, less widely known is that not all the ants follow the trail left by teammates (or by themselves in a previous trip). A small percentage of them, depending on the ant species and on the environment in which it has evolved, leave the trail and, in doing so, allow the colony to discover new, possibly richer feeding opportunities close to the main path of the currently exploited source (exploration).

4. Individual simplicity: Simplicity often increases the individual's robustness, allows for a unit's miniaturization, and reduces overall system cost.

While these features of the SI approach are appealing, we lack a methodology for designing individual agents that will demonstrate a specific coordinated, self-organized behavior. These SI principles do not provide us with a way to quantitatively predict the swarm performance according to a particular metric or analyze further possible optimization margins and intrinsic limitations of this approach from an engineering point of view. In other words, if we want to achieve coordinated, self-organized group behavior based on local interactions, we need to have appropriate tools for understanding how to design and control individual units so that the swarm can achieve target behaviors and levels of performance. Models allow the engineer to capture the dynamics of these nonlinear, asynchronous, potentially large-scale systems at more abstract levels, sometimes achieving even mathematical tractability. More generally, modeling is a means for saving time, enabling generalization to different robotic platforms, and estimating optimal system parameters, including control parameters and number of agents in a team.

Although for a long period early in collective robotics research there was relatively little work in modeling of multi-robot systems, recently physicists and engineers have dedicated more attention to this problem (see, for example, related work performed by Kazadi et al.

[27], Lerman and Galstyan [28], and Sugawara and Sano [29, 30]). Moreover, modeling methodologies for swarm robotics systems must take into account mobility, local intelligence, intrinsic stochastic properties of the collective coordination based on SI-principles, and, potentially, several different modalities of interaction among individuals and between an individual and the environment (e.g., mechanical, electromagnetic, chemical). This extremely rich combination of system features has drastically reduced the applicability of modeling techniques developed and commonly used in other fields.

1.1.2 Contributions of the Thesis: Modeling Swarm Dynamics

In Chapter 2, we combine the expertise accumulated in building probabilistic microscopic [4, 5, 9] and macroscopic models [31, 32, 33, 34] for distributed manipulation experiments characterized by different robotic platforms and tasks (aggregation, wall building, stick pulling) in a consistent, thorough, unified framework. This work, previously published in a collaborator's thesis [35] and in two journal papers [36, 37], was one of the first attempts to develop an ad hoc modeling methodology for swarm robotic systems.

The strength of this research lies in the fact that, in contrast to contributions which did not aim to produce quantitatively correct predictions without free parameters [27, 32] and to those with fewer implementation levels [28, 29, 30], we present microscopic and macroscopic models and we validate them with real robot and/or embodied simulations, precisely describing the mapping between different implementation levels of the same experiment. Furthermore, we illustrate the application of the probabilistic modeling methodology with several examples of incremental complexity, providing building blocks that may be applied to the construction of models for other experiments. We also investigate the strengths and limitations of the modeling methodology, particularly in comparison to other popular

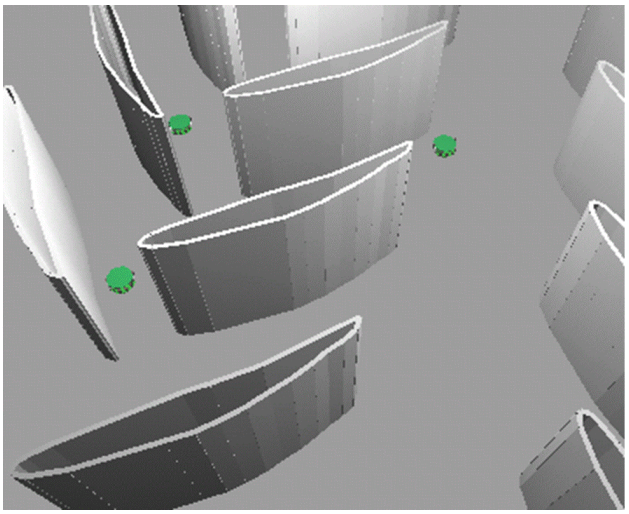


Figure 1.1: Nondestructive inspection of blade surfaces inside a turbine is a motivating application of the multi-robot boundary coverage problem. The green dots represent robots engaged in sensing tasks. (Picture courtesy of Nikolaus Correll.)

simulation tools such as sensor-based, embodied simulators.

1.2 Multi-robot Boundary Coverage

As we consider more complex target tasks and behaviors and as efficiency and completeness of execution become a major concern, a swarm approach may not be appropriate.

1.2.1 Multi-robot Coverage

There are a number of practical mechanical inspection, surveillance, and security applications where completeness of robotic coverage is of particular concern. In the classical robotic coverage problem, one or more robots must "cover" the freespace of a bounded two-dimensional workspace.

The multi-robot approaches to both posing and solving this *freespace coverage problem* have been diverse, exploiting many of the advantages offered by multi-robot systems, from robustness to robot failure [38] to sensor fusion to minimize localization error [39]. Generally,

in methods guaranteeing some form of completeness, a cellular decomposition of some sort is applied to the freespace to be covered, dividing it into regions for which coverage can be guaranteed. This cellular decomposition may be approximate, like the grid-based methods used by Wagner et al. for ant-inspired coverage algorithms [21], or exact, as in Butler's rectilinear coverage work [40]. Further examples may be found in Choset's survey of single- and multiple-robot approaches to the freespace coverage problem [41].

In Chapter 3, we pose and solve the complementary problem of *boundary coverage*, in which a robot or team of robots must inspect the boundary of a bounded two-dimensional environment. The initial motivating application for the boundary coverage work is the problem of inspecting in situ the surfaces of turbine blades inside a jet engine or combustor. Figure 1.1 shows an idealized view of the interior of a turbine, where the cylindrical geometry of the turbine has been "flattened." Using nondestructive sensing technologies, a group of robots (which may be small mobile robots, or the distal tips of manipulators carrying the inspecting sensors) must systematically inspect the surface of each turbine blade for defects. Inspecting the freespace between the blades, as would be done in the classical coverage problem, is not of concern here.

While collaborators pursued SI [42, 43, 44, 45] and evolutionary [46] approaches to solving nondeterministic forms of the turbine blade inspection problem, I considered a generalized, deterministic boundary coverage problem, in which completeness of inspection is required. The environments considered are two-dimensional, i.e., a horizontal cross-section slice through this world. The extension of our method to the geometry of Figure 1.1 (where the robot must inspect the entire height of the turbine blade extrusion) is relatively straightforward. Robotic security, surveillance, and "watchman's route" problems are another class of applications for boundary coverage techniques. Imagine, for example, that a group of

robots is tasked with monitoring the walls of an art gallery, or the surfaces of a set of oil tanks. A multiple-robot approach to such problems should allow the task to be completed or repeated more quickly than with a single vehicle, while offering flexibility through redundancy in case a unit should fail.

1.2.2 Use of Graph Theory in Multi-robot Systems

Graph theory has been widely used in muti-robot applications including the control of vehicles traveling in formation [47, 48, 49], and the description of interactions between self-assembling reconfigurable robots [50]. As communicating robots may be considered to be mobile sensor nodes in a network, the application of graph theory common in sensor network literature is also frequently used to model and analyze the communication in a multi-robot systems [51, 52, 53]. In navigation tasks, graphs offer a convenient way to represent map topology [54, 55] and the connectedness of cellular decompositions [56, 38]. This last set of applications is most relevant to the work presented in this thesis.

1.2.3 Contributions of the Thesis: Designing Inspection Planning Algorithms

In Chapter 3, we formalize the *multi-robot boundary coverage problem*. In this problem (which is more fully described in Section 3.1), a group of k robots is required to completely inspect the boundary of all two-dimensional objects in a specified environment. The boundary coverage problem complements the freespace coverage problem; analogous to that problem, the goal is to devise an algorithm that guarantees the completeness of coverage of all boundary points, i.e., that a solution to the problem will be found and that the conditions guaranteeing the existence of a solution are specified. The efficiency of the robots'

efforts is also of concern, so the algorithm seeks to balance the workload evenly between the cooperating robots. The work presented in Chapter 3 was the first to pose the deterministic multi-robot boundary coverage problem and the first to present a complete algorithm for its solution. Using a simplified sensor model, the inspection problem is converted to an equivalent graph representation. Once a graph representation of the task has been constructed in which the undirected, connected graph G has a subset of edges E_R, every one of which must be traversed for inspection to be complete, the edge coverage problem can be posed as a "postman"-type graph problem. I pose this graph problem, which I term the *k-Rural Postman Problem*, then develop a constructive heuristic for its solution. The robots use the graph solution to plan the their inspection routes.

In Chapter 4, extensions to the boundary inspection planning method are explored. Using building blocks from the boundary inspection planning algorithm, related inspection tasks that require the robotic system to adapt to a changes in team size and task assignment are addressed. The application of these planning methods to the case of long-term deployment for surveillance applications that require repetitive coverage is also discussed.

Chapter 2

Modeling Swarm Robotic Systems

In this chapter, we present a time-discrete, incremental methodology for modeling the dynamics of distributed manipulation experiments using swarms of autonomous robots endowed with reactive controllers; most of the results presented here were previously published in a collaborator's thesis [35] and in two journal papers [36, 37].

Because the methodology does not take into account robots' trajectories or the spatial distribution of objects in the environment, it is well-suited for nonspatial metrics, where these factors do not play a role. The strength of the methodology lies in the fact that it has been generated by considering incremental abstraction steps, from real robots to macroscopic models, each with well-defined mappings between successive implementation levels. Though advances in hybrid system abstraction [57, 58] offer insight into the analysis and design of some classes of hybrid systems, the self-organized systems abstracted here are not well-suited to current abstraction methodologies. Precise heuristic criteria based on geometrical considerations and systematic tests with one or two real robots prevent the introduction of free parameters in the calibration procedure of models. As a consequence, we are able to generate highly abstracted macroscopic models that can capture the dynamics of a swarm of robots at the behavioral level while still being closely anchored to the characteristics of the physical setup.

The methodology is developed incrementally, using an experiment we refer to as the *stick pulling experiment* as a modeling testbed. This case study is concerned with pulling sticks out of the ground, an action that requires the collaboration of two robots to be successful. Experiments were carried out with teams consisting of two to 600 individuals at different levels of implementation (real robots, embodied simulations, microscopic and macroscopic models). Results show that models can deliver both qualitatively and quantitatively correct predictions at least four orders of magnitude faster than an embodied simulation and that they represent a useful tool for generalizing the dynamics of these highly stochastic, asynchronous, nonlinear systems, often outperforming intuitive reasoning.

A case study concerned with object aggregation and worker allocation is also presented as an example of how the modeling methodology may be applied to other experiments in distributed manipulation.

Finally, in addition to discussing subtle numerical effects, small prediction discrepancies, and difficulties in generating the mapping between different abstractions levels, we conclude the chapter by reviewing the intrinsic limitations of the current modeling methodology and by reviewing the continuing progress made by collaborators pursuing a swarm robotic approach to the nondeterministic boundary coverage problem.

2.1 Multi-level Probabilistic Modeling

The central idea of the probabilistic modeling methodology is to describe a multi-robot experiment as a series of stochastic events with probabilities computed from the robot interactions' geometrical properties. In the experiments considered in this chapter, a robot may be described with a Probabilistic Finite State Machine (PFSM) or Markov chain whose state-to-state transitions depend on the interaction probabilities of a robot with another

teammate and with the environment.

While in lower-level microscopic models each robot is represented by its own PFSM, in the high-level macroscopic models a single PFSM directly describes the whole robotic team, each of its states representing the average number of teammates in a particular state at a certain time step. In both types of models, the robots' PFSM(s) are then coupled with the environment. This coupling among robots via the environment (or in other experiments, direct peer-to-peer coupling, for example, through explicit communication) shapes the microscopic-to-macroscopic mapping, in particular determining its linear or nonlinear properties.

At both modeling levels, the environment can be viewed as a passive, shared resource reflecting any modifications generated by the parallel actions of the robots. In order to compute an arbitrary nonspatial performance metric, the modification of the environment is tracked; each simulated modification is recorded in the microscopic model while the macroscopic model estimates average environmental quantities.

The current modeling methodology relies on two main, somewhat overlapping, common assumptions: spatial uniformity and the fulfillment of Markov properties.

We assume that the coverage of the arena by the groups of robots is as uniform as if the robots could hop around randomly on the surface. Robots' trajectories or specific robot spatial distributions therefore are not considered in the current models. We also assume that the absolute position of a given object to manipulate in the arena does not play a role: for instance, the object will have the same probability of being manipulated whether it is placed in the center or in the periphery of the arena. For the random walk implementation used in the systems explored in this chapter, as we consider the average distribution of the robots over long periods, this assumption holds true.

We also assume that the robot's future state depends only on its present state and on how much time it has spent in that state. This assumption is correct for a reactive robot controller extended with a time-out or following a predetermined sequence of actions (e.g., gripping a stick, dancing) that lasts a certain amount of time. The robots (and the environment) in the stick pulling case study clearly obey this Markov property if we assume we are considering all robots' and environment's states of interest for computing the desired nonspatial metric (e.g., trajectory states–position and heading–can be neglected).

Both assumptions are valid in most of the experiments presented in this chapter. The exceptions can be found in Section 2.6.2.2, where the nonspatial model assumption is experimentally validated, and in Section 2.6.2.3, where overcrowding is discussed as a case in which both assumptions are no longer valid.

2.2 A Case Study: The Stick Pulling Experiment

To illustrate the incremental construction of these models, we consider the stick pulling experiment as a testbed for our modeling methodology. The experiment presented here follows initial tests presented by Martinoli and Mondada [10]. The task is to locate sticks in a circular arena and to pull them out of the ground[1] using Khepera [60] robots equipped with grippers and capable of distinguishing the sticks from walls and other robots with their frontal sensors. Due to the sticks' length, a single robot cannot pull a stick out of the ground alone; collaboration between two robots is necessary. As the robots have only local sensing capabilities and do not exploit a fully connected communication network, there is neither central nor global coordination among robots. Coordination is purely probabilistic and

[1] Although the experiment is not intended to reproduce a biological system, it presents several similarities with the extraction and transportation of twigs performed by some ant colonies [59].

happens based on local interactions, strictly following the SI principles outlined in Section 1.1. Though all of the described experiments using real robots were conducted with teams ranging from one to six robots, only the cost of the equipment prevented them from being carried out with larger team sizes, as we did with the help of an embodied simulator and with our models. The metric measured to quantitatively investigate and model the effects of variations of system parameters is the collaboration rate among robots, i.e., the number of sticks successfully taken out of the ground over time. The metric is nonspatial (i.e., we consider all the sticks pulled by the robots throughout the arena, independent of the sticks' placement) and measured by the experimenter rather than by the robots themselves.

The specific interest for the stick pulling experiment lies in its strictly collaborative nature. In the class of distributed manipulation experiments considered by my collaborators thus far, the stick pulling experiment is unique, since the other tasks studied can be completed by individual robots, the collective effort allowing the swarm mainly to improve its performance over time or robustness in task accomplishment. From a modeling point of view, two coupling mechanisms among robots are present in this experiment. First, robots modify a "shared blackboard" (i.e., the environment) and each robot's actions therefore indirectly influence those of its teammates, as in other distributed manipulation experiments. Second, when a robot triggers a mutual action with another robot, both robots carry out a precise temporal sequence of actions culminating in a successful collaboration.

Finally, it is worth noting that similar collaboration dynamics could require an arbitrary number of robots (see, for example, the description reported in Lerman et al. [32]) and arise in a completely different experiment, one not involving object manipulation at all. For example, self-locomoted sensor nodes characterized by pseudo-random movement patterns, endowed with local communication capabilities, and engaged in a monitoring task over a

well-delimited area, could be represented in the same abstracted way as robots engaged in the stick pulling experiment. In this case, the metric used to assess the swarm performance could be related to the number of successful event detections reported by the swarm to a base-station knowing that, before emitting an alarm signal, a threshold number of robots within the swarm should collectively agree to have detected the same event.

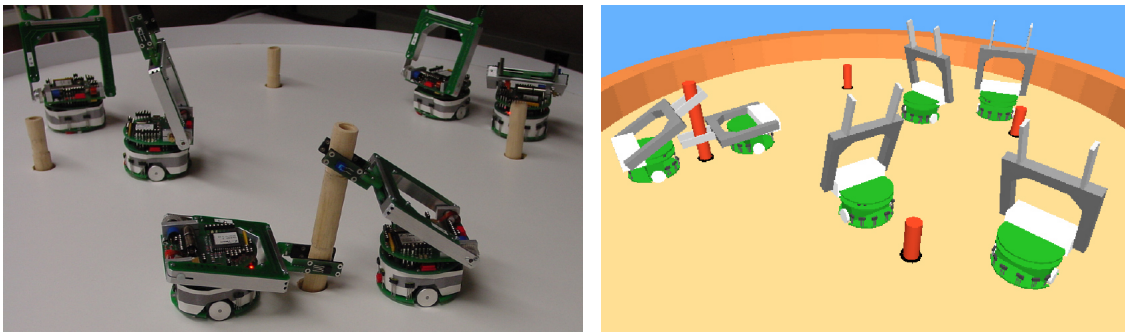
2.2.1 The Physical Setup

The experiment is carried out in a circular arena (40 cm in radius) delimited by a white wall. Four holes situated at the corners of a square with 30 cm edges hold white sticks (15 cm long, diameter of 1.6 cm) that, in their lowest position, protrude 5 cm above the ground (see Figure 2.1(a)).

Groups of two to six Khepera robots, equipped with gripper turrets, are used to pull the sticks out of the ground. Because of their thinness, the sticks can be distinguished from the wall and from other robots using the Khepera's six frontal infrared proximity sensors. Because the sticks are too long to be pulled from the ground by a single robot's lifting motion, collaboration between two robots is required. After a successful collaboration, the stick taken out of the ground is released by the robot, and replaced in its hole by the experimenter.

2.2.2 Embodied Simulations

In order to more systematically investigate the collaboration dynamics, we also implemented the experiment in Webots [61], a 3-D, kinematic, sensor-based simulator of Khepera robots (see Figure 2.1(b)). Teams of two to 24 robots were simulated using Webots. The simulator computes trajectories and sensory input of the robots in an arena corresponding to a given



(a) (b)

Figure 2.1: (a) Overview of the physical setup for the stick pulling experiments (4 sticks, arena 40 cm in radius). (b) Corresponding setup in the embodied simulator.

physical setup. The resulting simulation is sufficiently faithful for the controllers to be transferred to real robots without changes and for the simulated robot behaviors to be very similar to those of the real robots, as shown in several previous papers [4, 5, 9, 17].

A simulated stick pulling experiment using five robots proceeds on average 18 times faster than real time on a 900 MHz Pentium III. In comparison, the mean speed-up ratio for this experiment with five robots between the microscopic model (implemented in C) and Webots simulations is about 25,000 on a 900 MHz Pentium III machine. The macroscopic model (implemented in Matlab®) will run approximately $4 \times N \times R$ faster than the microscopic model, where N is the total number of robots in the team and R is the total number of runs used for obtaining the microscopic model's mean performance.

2.2.3 The Robots' Controllers

The behavior of a robot is determined by a simple hand-coded program that can be represented with a standard flow chart or a Finite State Machine (FSM), as depicted in Figure 2.2(a). The behavioral granularity shown in Figure 2.2(a) is arbitrary and is chosen by the experimenter so that the FSM captures all the details of interest.

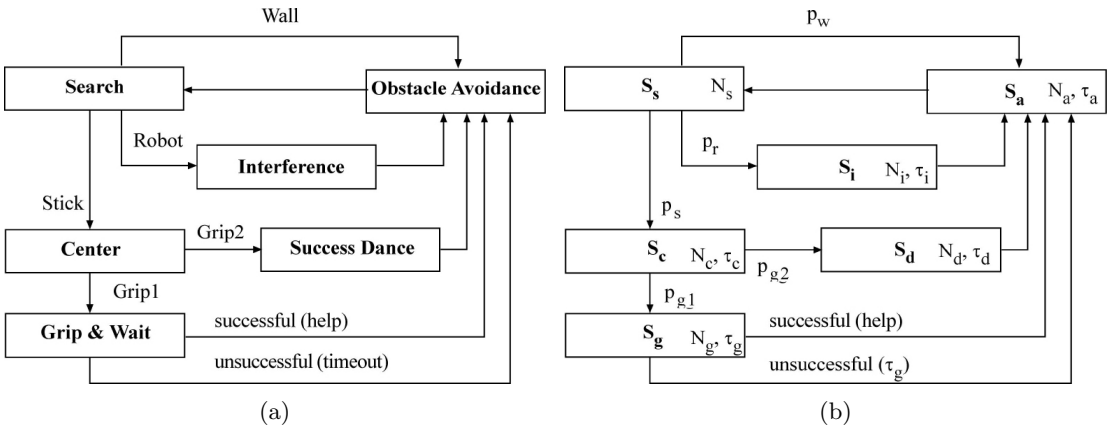


Figure 2.2: (a) FSM representing the robot controller. Transitions between states are deterministically triggered by sensory measurements. (b) PFSM representing an agent in the microscopic model or the whole robotic team in the macroscopic model. The parameters characterizing probabilistic transitions and states are explained in the text.

In addition to the default search behavior (moving in a straight line) and an obstacle avoidance behavior, the robot is endowed with a stick-gripping and -pulling procedure. The robot can determine from its arm elevation speed while pulling whether another robot is already gripping the same stick. While waiting for collaboration, another robot's attempt to lift the stick is similarly detected with the arm elevation sensor. If a single robot is holding the stick, we call such a grip *grip1*. If another robot is already holding the stick and a searching robot finds and grips it, such a grip is called *grip2*. When a robot makes a grip1, it holds the stick raised halfway out of the ground and releases it when either the duration of the grip exceeds the *gripping time parameter* τ_g (a failed collaboration), or another robot comes to make a grip2 (a successful collaboration). Once the stick is released, the robot turns away, performs obstacle avoidance for a few seconds, then returns to the search procedure. When a robot makes a grip2, the robot making the grip1 will release the stick, allowing its teammate to raise the stick completely. The robot making grip2 performs a short "success dance" (moving the arm up and down) to mark the successful

collaboration, then releases the stick, performs obstacle avoidance for a few seconds, and resumes searching for sticks.

Because the sticks are recognized only by their thinness, a stick that is held by one robot can only be recognized as such when approached from the opposite side within a certain angle (approx. 125 degrees in the physical setup). For the other angles of approach, both the stick and the robot are detected and the whole is taken for an obstacle. More details are reported in Ijspeert et al. [9] but we note that the acceptable approach angle, expressed as a ratio R_g over the whole approaching perimeter, is an important parameter in the collaboration dynamics of the system.

2.2.4 Calibrating Modeling Parameters

Figure 2.2(b) shows a Probabilistic Finite State Machine (PFSM) describing the stick pulling behavior whose state-to-state transitions depend on the interaction probabilities of a robot with another teammate and with the environment.

The models presented are characterized by two different categories of parameters: transition probabilities and delays. In the following subsections, we will describe how we calculated and measured all the parameters belonging to either one or the other category.

2.2.4.1 Transition probabilities

Consistent with previous publications [4, 5, 9, 31, 33, 34], we compute the transition probabilities from a state to another based on simple geometrical considerations about the interaction (e.g., detection areas, approaching perimeters). The numerical values used for these geometrical parameters correspond to the average values measured in systematic tests with one or two real robots, as mentioned above. In nonspatial models, robots' positions on

the arena are assigned randomly at each new iteration (or time step). At each iteration, the probability that a robot in the search mode will encounter a wall, a stick, or another robot is determined by each object's corresponding detection area divided by the whole arena area A_a. For instance, the probability of finding a stick can be computed as $p_s = A_s/A_a$, A_s being the detection area of a stick. Similarly, $p_w = A_w/A_a$ and $p_r = A_r/A_a$ represent the probability of encountering a wall and another robot, respectively. The total probability of encountering any other robot on the arena can be computed as $p_R = (N_0 - 1)p_r, N_0$ being the total number of robots in the arena. Additionally, since robots can perform a grip1 from any angle of approach, $p_{g1} = p_s$. On the other hand, since a stick available for grip2 can only be approached from a certain angle, as mentioned above, the probability of a grip2 event is $p_{g2} = R_g p_{g1}$. The total number of sticks in the arena is M_0. See Ijspeert et al. [9] for more details.

Furthermore, the current implementation of the model does not allow for overlapped areas of detection between objects of different type (e.g., robot, wall). Equation 2.1 expresses this limit mathematically:

$$p_w + p_s M_0 + p_R \leq 1. \tag{2.1}$$

This, in turn, sets a boundary for the maximal detection areas occupied by robots to $p_R = 1 - p_w - p_s M_0$. As will be seen in Section 2.6.2.3, this formula is just an approximation that forces additional robots to "squeeze" into their maximum available freespace. In overcrowded scenarios, the modeling methodology reaches its limitations.

The numerical values used for the mean robot speed, the mean approaching angle for grip2, and the different mean detection radii of the objects are summarized in Table 2.1.

Table 2.1: Parameters used in the models of the stick pulling experiment. Detection distances are measured from the center of the robot to the center of the object detected.

Mean robot speed v [cm/s]	Mean approaching angle ratio R_g	Mean wall detection distance R_w [cm]	Mean stick detection distance R_s [cm]	Mean robot detection distance R_r [cm]	Arena radius R_a [cm]
8	0.35	6	6.4	10	40 to 5000

These values are exactly the same as those reported in Ijspeert et al. [9], although a single arena of 40 cm in radius was used at that time.

2.2.4.2 Time discretization and delays

The current probabilistic methodology generates time-discrete models. In this subsection we would like to motivate our choice and explain how we establish the discretization interval T as well as the discretized delays considered in the models.

Time-Discrete vs. Time-Continuous Models At a first glance, we might think that, since the physical setup operates in continuous, real time and agents operate asynchronously, the best way to obtain a faithful model would also be to use a time-continuous model. As usual, we would emulate continuous time in simulation by choosing a small time step combined with a standard numerical integration algorithm (e.g., Runge-Kutta). However, if we look more closely, we realize that the description of the system we are interested in (see Figure 2.2(b)) is at a much higher level than that which would involve mass, speed, forces, positions, and so on: it is a description characterized by logical operators and behavioral states. Robots are natural hybrid systems, in a strictly automatic control sense: they consist of elements whose dynamics evolve continuously (e.g., motors, sensors, and analog electronics), but they are also endowed with microcontrollers that, in addition to being

digital and clocked, may also implement high-level, logical control operations. The success dance in the stick pulling experiment is a typical example: its duration is triggered by a sensory stimulus and systematically lasts until an incremental timer of the robot reaches a pre-established time-out. Since we are not interested in modeling the details of the success dance, a time-discrete model would capture this duration precisely without extra computation in between. This is the main reason we believe time-discrete models are the most adequate solution for the level of description we are aiming for, although at the microscopic level they will force state transitions of the PFSMs to happen at the end of time steps rather than completely asynchronously. Emulation of time continuity would simply add simulation time (even macroscopic models, being, in general, nonlinear, must be solved numerically) without increasing prediction accuracy and prevent the description of microscopic and macroscopic models under the same temporal framework.

Time Discretization Interval Consistent with previous publications [31, 9, 4, 5, 33, 34], each iteration of our models corresponds to a time step of a finite duration in real time. The duration of a time step is equivalent to the time needed for a robot, moving with a certain mean speed v and having a certain mean detection width w_i for the smallest object in the arena i (in our case, a stick), to cover the smallest object's detection area. Equation 2.2 shows how to compute the duration T of one time step in the modeling methodology:

$$T = \frac{A_s}{v w_s} = \frac{\pi R_s^2}{v 2 R_s} = \frac{\pi R_s}{2v}. \tag{2.2}$$

Choosing the smallest object is a way to ensure that the time granularity is high enough to capture every occurrence of the system's fastest detection event. This is an approximation that links time partitioning with probability-space partitioning and is consistent with the

Table 2.2: Duration of the different robot maneuvers described in Section 2.2.3.

Delay	Centering	Success Dance	Obstacle Avoidance	Interference
Mean measured value [s]	$\tau_c = 10$	$\tau_d = 6$	$\tau_a = 1$	$\tau_i = 2$
Discretized value [iterations]	$T_c = 8$	$T_d = 5$	$T_a = 1$	$T_i = 2$

fact that our models are nonspatial.[2] Although this method for choosing the time step is completely heuristic, it has provided, combined with the method of calculating the transition probabilities explained in Section 2.2.4.1, the best results so far not only in the stick pulling experiment but also in aggregation [5] and wall-building [4, 5] experiments performed with different robotic platforms. We will discuss the difficulties inherent to parameter calibration and time discretization more extensively in Section 2.6.2.1.

Measured and Discretized Delay Values Table 2.2 summarizes the values of delays used in all the models presented in this chapter. The measured mean delays are exactly the same used in Ijspeert et al. [9] The time step has been calculated with Equation 2.2: $T = 1.26$ s. Of course, any gripping time parameter τ_g will be also discretized to T_g iterations in the models.[3]

2.3 Model Construction: Modeling Key Sub-chains

Before describing implementation details and results of the full-system model, we introduce two examples of two-state PFSMs. Both examples not only represent key sub-chains of the full system's Markov chain we will describe in Section 2.4 but may also be used as basic

[2] An alternative explanation of this heuristic formula can be found in Martinoli's Ph.D. thesis [62], Chapter 4. Ijspeert et al. [9] used a slightly different robot detection width (a value that can be considered as a weighted sum of all the different detection widths specific to different objects in the arena).

[3] As the time step is different from Ijspeert et al. [9], the corresponding discretized delays used in these models may be slightly different after rounding.

elements in constructing models of other swarm robotic experiments.

The first example is concerned with a delay component affecting each robot's behavior. The delay could be defined by an internal timer (e.g., the success dance or any finite period required for processing information) or by a specific interaction with the environment (e.g., gripping a stick or any finite period needed for sensing or acting in the environment) or a teammate (e.g., obstacle avoidance or any finite period needed for information broadcast). The corresponding difference equations (DEs) at the macroscopic level are linear since we assume that there is no coupling between agents, either directly (agent-to-agent) or indirectly (through the environment): the specific action generating the delay lasts always for a pre-established period independent of the state of the system.

The second example can be considered as a simplified model of the stick pulling experiment, which involves the overlapping of the two nonlinear coupling mechanisms among the agents and therefore among the DEs: *environmental modification* and *triggering of mutual actions*. In contrast to other distributed manipulation experiments concerned with, for instance, aggregation and segregation of objects based on stigmergic[4] mechanisms, the environmental modification in the stick pulling experiment does not have plastic properties as result of the action of a single robot. In fact, as soon as a robot releases a gripped stick because no other robots came to help, the *environmental modification* performed (the lifting of the stick) is automatically reversed by the force of gravity as the stick falls back into its hole. In the version of the stick pulling experiment presented here, the *triggering of mutual actions* is obtained by the sequence of movements encoded in each robot controller, in essence, a primitive handshaking protocol that uses the stick as the communication medium.

[4]The concept of stigmergy was introduced for the first time by P.P. Grassé [63]: it comes from the Greek stigma (sting) and ergon (work); it describes a phenomenon in which a plastic modification of the environment introduced by the work of an individual is perceived as a stimulating configuration by other agents (or by the same agent at a later time).

Nothing will prevent us, however, from implementing this handshaking protocol with local wireless communication and obtaining, at the model level, the same description, with states simply characterized by different delays.

Unless otherwise stated, experiments using the microscopic model have been repeated 100 times and error bars represent standard deviation among runs. At the macroscopic level, of course, one run suffices, since only system-level performance (in our case the mean swarm performance) can be predicted.

2.3.1 Search and Obstacle Avoidance

One key element of the full-system model representing the stick pulling experiment is a delay state. . A delay state simply represents a behavior the robot will perform for a certain duration T_a with a probability p_a. The probability of leaving the delay state after T_a is one and is independent of the robot's interaction with the environment or with the teammates. Constructing a model for a delay state is not only an important step for modeling the stick pulling experiment; such a model may also be used as a building block in modeling other self-organized robotic systems.

As an example of a delay state in the stick pulling experiment, we have chosen a subchain of the system represented by the default search state coupled with an obstacle avoidance state. In reality, depending on the obstacle type, an avoidance maneuver may be characterized by different durations and different probabilities of performing it. Furthermore, depending on the coverage of the proximity sensors, the collision angle between two robots, and the type of motion executed by the robot, obstacle avoidance behavior may or may not be mutually triggered (resulting in a nonlinear coupling in the DEs, as shown in Lerman and Galstyan [28]). For sake of simplicity, in this subsection we will use a single

obstacle type and no mutual triggering of obstacle avoidance.[5] Values for probabilities and durations are derived from Table 2.1 and Table 2.2.

Figure 2.3 graphically represents the state diagram of this simple PFSM. N_s and N_a represent the numbers of robots in the search state and obstacle avoidance states, respectively. In the microscopic model, these are binary values at the level of each PFSM since an agent can be in one single state at the time; they are integer values corresponding to the actual numbers of robots in each state at the team level. In the macroscopic model, N_s and N_a are real values representing the average numbers of robots in a certain state, at a given time step, and over multiple runs.

Since the state variables of the macroscopic model are represented by continuous quantities, the PFSM of Figure 2.3 representing the whole swarm can be described by the following DE system:

$$N_s(k+1) = N_s(k) - p_a N_s(k) + p_a N_s(k - T_a) \tag{2.3}$$

$$N_a(k+1) = N_0 - N_s(k+1) \tag{2.4}$$

The current iteration is represented by k. $N_x(k)$ represents the value of the state variable N_x at time kT; the notation $N_x(k)$ instead of $N_x(kT)$ is standard in the automatic control literature. Equation 2.3 states that the average number of robots in the search state at iteration $k+1$ is equal to the average number of robots in search at iteration k minus those that left for an obstacle avoidance maneuver plus those that have terminated their avoidance. Equation 2.4 simply exploits the conservation of the total number of robots for

[5] As shown in Section 2.6.2, this linear approximation is quite faithful for a platform such as the Khepera robot, which is endowed with a proximity sensory belt affected by relevant blind angles on the sides, around the wheels. Namely, this implies in turn that, depending on the angles of approach, an interaction between robots may involve only one of them exhibiting obstacle avoidance behavior.

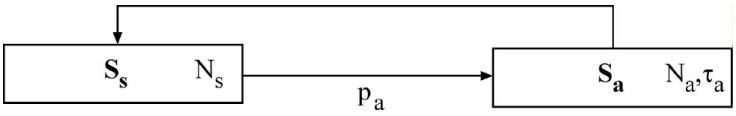


Figure 2.3: A simple sub-chain consisting of a search and an obstacle avoidance state. The numerical values used in this example have been derived from the values of Table 2.1 and Table 2.2 using parameters for a generic obstacle: $T_a = 2$ iterations and p_a is a function of the setup (for example, $p_a = 0.63$ for the mean probability of encountering an obstacle in an arena of 40 cm, 4 robots, and 4 sticks; in this case teammates, walls, and sticks are all considered as obstacles).

calculating the average number of robots in obstacle avoidance.

Unless otherwise stated, we assume that no robots exist before iteration $k = 0$ (a standard convention for this type of time-delayed DE). Mathematically speaking:

$$N_s(k) = N_a(k) = 0 \text{ if } k < 0. \tag{2.5}$$

As all robots are in the search state at the beginning of the experiment, the initial conditions for the DE system are $N(0) = [N_s(0)\ N_a(0)]^\mathsf{T} = [N_0\ 0]^\mathsf{T}$.

2.3.1.1 Microscopic and macroscopic results

Figure 2.4(a) shows a comparison between microscopic and macroscopic predictions for the mean number of searching robots at steady state and different team sizes. As it could be easily demonstrated mathematically by summing up and averaging individual quantities, in a linear system the microscopic-macroscopic transformation is straightforward and no approximation is required for obtaining a closed form. As a consequence, both models' predictions perfectly coincide on the stationary mean number of robots in search mode, also for small teams, while the standard deviation among runs and over time of the microscopic model is about inversely proportional to the swarm size. It is worth noting that the mean

number of robots in obstacle avoidance mode at steady state can be easily derived with Equation 2.4.

2.3.1.2 Steady state analysis

Since the DE system in Equations 2.3-2.4 is linear, we can analyze the steady state of the system either using a z-transform (frequency domain) or in time domain. Both analyses bring the same result.

Frequency domain Equation 2.3 can be transformed using the right shift and left shift theorems in the z-space as follows:

$$zN_s(z) - zN_0 = N_s(z) - p_a N_s(z) + p_a N_s(z) z^{-T_a}. \tag{2.6}$$

Solving for $N_s(z)$ and applying the limit theorem we obtain:

$$N_s^* = \lim_{k\to\infty} N_s(k) = \lim_{z\to 1} (z-1) N_s(z) = \frac{N_0}{1 + p_a T_a}. \tag{2.7}$$

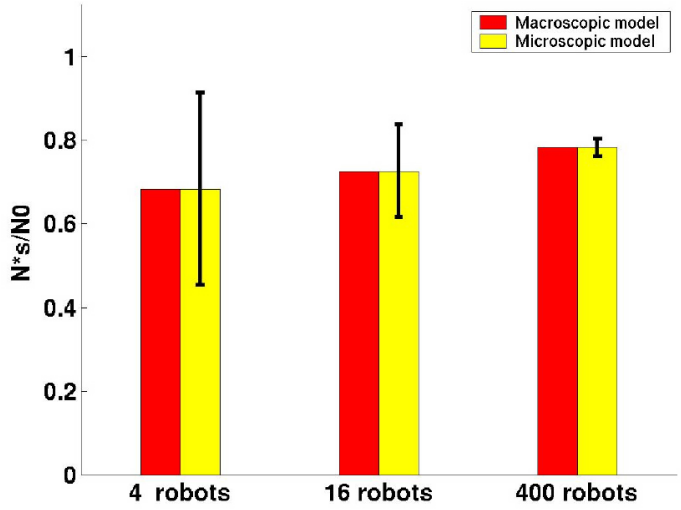
And therefore through the robots' conservation law:

$$N_a^* = N_0 - N_s^* = \frac{N_0 p_a T_a}{1 + p_a T_a}. \tag{2.8}$$

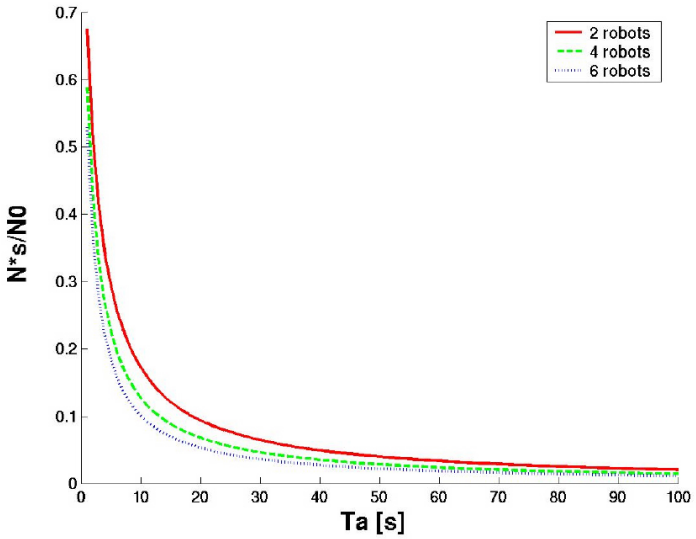
$N^* = [N_s^*\ N_a^*]^{\mathsf{T}}$represents the state vector of the DE system in the steady state regime.

Time domain Equation 2.4 could have been written also in the same form as Equation 2.3, i.e., as a DE instead of an equation:

$$N_a(k+1) = N_a(k) + p_a N_s(k) - p_a N_s(k - T_a). \tag{2.9}$$



(a)



(b)

Figure 2.4: (a) Comparison of the steady state N_s^* obtained with microscopic and macroscopic models for different team sizes and different arenas (density of teammates per surface unit was kept constant). The height of the microscopic column represents the mean value of N_s^* over a 1000 s time window and over 100 runs. The slight increase in the mean value for different team sizes is due the fact that the wall detection surface becomes proportionally smaller in bigger arenas. The error bars represent a mean standard deviation calculated as an average of the standard deviations measured on each run over the same time window. (b) Graphical representation of Equation 2.7 for different delay durations and different probabilities of encountering an obstacle in the 40 cm arena (the greater the swarm size, the larger p_a is).

Equation 2.9 represents a delay state. A first, intuitive step for calculating the steady state of this equation could be to set $N_a(k+1) = N_a(k) = N_a^*$ and $N_s(k) = N_s(k-T_a) = N_s^*$. However, we can immediately see that in this case Equation 2.9 will be underdetermined, i.e., $0 = 0$.

A workaround step for this problem is as follows: we can say that after T_a iterations, every robot that enters the delay state must leave it with probability one. Since no robots exist for negative iterations (see Equation 2.5), the only time during which N_a can be increased is during the first T_a iterations (full inflow with probability p_a and zero outflow). Although $N_s(k)$ may vary during the first T_a iterations, we can approximate it as a constant (i.e., as if in steady state) if T_a is small:

$$N_a^* = \int_0^{T_a} p_a N_s^* dk = p_a N_s^* \int_0^{T_a} dk = p_a N_s^* T_a. \tag{2.10}$$

By combining Equation 2.10 with the robots' conservation law it is easy to demonstrate that N_s^* and N_a^* can be expressed as we calculated via the z-transformation (Equation 2.7 and 2.8). This second method of calculating the steady state vector for delay states is extremely useful if the whole system involving this type of state is nonlinear. This will be the case in the full-system model describing the dynamics of the stick pulling experiment. Figure 2.4(b) shows graphically how the normalized average number of searching robots at the steady state is influenced by the time delay T_a and different probabilities p_a.

2.3.2 Simplified Stick Pulling System: Search and Grip

A second key sub-chain of the full-system model captures the two nonlinear couplings among agents mentioned before: environmental modification and triggering of mutual actions. The former mechanism, although it does not generate plastic modifications of the environment,

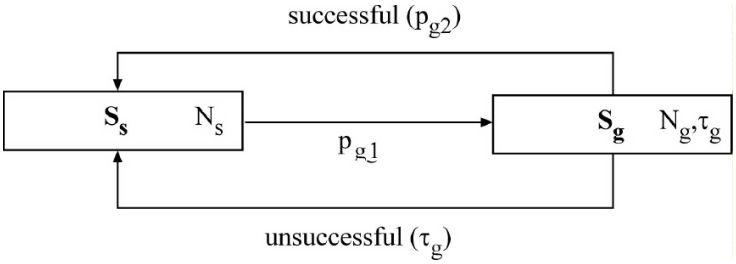


Figure 2.5: The key sub-chain representing the dynamics of collaboration in the stick pulling experiment. The numerical values used in this simplified model have been derived using Table 2.1.

has indirect consequences on the action of the teammates since the gripping of a stick reduces the opportunities for other teammates to find free sticks for grip1. The latter mechanism is instead due to the strictly collaborative nature of the stick pulling experiment and the way robots communicate through sticks' manipulation.

If we look carefully the PFSM of Figure 2.2, we notice that several states can be thought of as simple delay states. The durations of the delays do not exceed a few seconds and are therefore much shorter than most of the values of the gripping time parameter considered in our experiments, simulations, and models (up to 600 s). As a consequence, not only does the sub-chain described in this section represent an example of modeling swarm robotic experiments that involve both distributed manipulation and local communication, it also represents the core of the collaboration dynamics in the stick pulling experiment. Indeed, by neglecting all the minor delays, the full-system model reduces to the very same two states: search and grip (see Figure 2.5).

The following system of DEs represents the macroscopic model for this search-grip sub-chain:

$$N_s(k+1) = N_s(k) - \Delta_{g1}(k)N_s(k) + \Delta_{g2}(k)N_s(k) + \Delta_{g1}(k-T_g)\Gamma(k-T_g;k)N_s(k-T_g) \quad (2.11)$$

$$N_g(k+1) = N_0 - N_s(k+1) \tag{2.12}$$

In words, Equation 2.11 tells us that the mean number of robots in search state at any time is decreased by the robots transitioning to a gripping state (Δ_{g1}) and is increased by the robots coming back from a successful collaboration (Δ_{g2}) and those coming back from an unsuccessful collaboration ($\Delta_{g1}\Gamma$). Equation 2.12 again exploits the conservation of the total number of robots for calculating the mean number of robots in the grip state. The nonlinear coupling between equations is achieved with Δ- and Γ-functions as follows:

$$\Delta_{g1}(k) = p_{g1}[M_0 - N_g(k)] \tag{2.13}$$

$$\Delta_{g2}(k) = p_{g2}N_g(k) \tag{2.14}$$

$$\Gamma(k - T_g; k) = \prod_{j=k-T_g}^{k} [1 - p_{g2}N_s(j)] \tag{2.15}$$

The Δ_{g1}-function characterizes the environmental modification mechanism for single robots and the Δ_{g2}- and Γ-functions characterize the triggering/non-triggering of mutual actions. Equation 2.13 indicates that the number of sticks free for gripping is equal to the total number of sticks M_0 minus those that are already "busy". Equation 2.14 tells us that we need another robot already gripping a stick, and a correct approaching angle for the searching robot in order to achieve successful collaboration, while Equation 2.15 represents the fraction of robots that abandon the grip state after the time spent in this state exceeds their gripping time parameter T_g. As explained more extensively in Lerman et al. [32], this

is equivalent to calculating the probability that no other robot came "to help" during the time interval $[k - Tg, k]$.

Finally, our team metric, the (average) collaboration rate $\overline{C_t}$, can be computed from the cumulative number of successful collaborations C over the maximal number of iterations T_e:

$$C(k) = p_{g2} N_s(k) N_g(k) \tag{2.16}$$

$$\overline{C_t} = \frac{\sum_{k=0}^{T_e} C(k)}{T_e} \tag{2.17}$$

As in the case of the two-state system described in Section 2.3.1, the initial conditions for the DE system are $N(0) = [N_s(0)\ N_a(0)]^\mathsf{T} = [N_0\ 0]^\mathsf{T}$.

2.3.2.1 Microscopic and macroscopic results

Figure 2.6 shows a comparison between microscopic and macroscopic predictions of the collaboration rate (left column) and of the steady state number of robots in search (right column) for different arenas and swarm sizes. A first striking result is the existence of two different system dynamics as a function of the ratio between number of robots and number of sticks. When there are more robots than sticks, the collaboration rate increases monotonically with the gripping time parameter and eventually saturates in a plateau corresponding to the optimal collaboration rate. In other words, under these conditions, it is a good strategy for a robot gripping a stick to wait a very long time for another robot to help, because there will always be at least one "free" robot available. In contrast, when there are fewer robots than sticks, waiting a very long time becomes a poor strategy, because all the

robots lose time holding different sticks while no other robots are available to collaborate. As an extreme example, an infinite gripping time parameter would lead to a null collaboration rate with all robots eventually holding a different stick permanently. Although this intuitive explanation is correct for the physical setup considered here, it is not *generally* true since the sticks-to-robots ratio at which the system bifurcates is dependent on the R_g parameter, as we will show in the next subsection and as we have also demonstrated in the full-system model in a recent publication [34].

A second observation we can make about Figure 2.6 is that, much like the two-state example presented in Section 2.3.1, the smaller the swarm size is, the larger the standard deviation is among runs using the microscopic model (visible in both the collaboration rate and steady state variables). Here we notice, however, that for very small swarm sizes for which there is at most the same number of robots as sticks, (see Figure 2.6(a), first row) macroscopic and microscopic models' predictions diverge quantitatively. Other, much smaller discrepancies can be observed between microscopic and macroscopic predictions with larger swarm sizes (400 and 600 robots, Figure 2.6, last row). This is counterintuitive since one would think that the problem shown in the first row of Figure 2.6 would only have to do with the small number of robots and sticks used in this scenario, quantities not large enough to satisfy the law of the large numbers on which macroscopic models base their calculations of central tendencies. A more careful inspection of the state variables in steady state, much like what was done in Section 2.3.1.1, reveals that the collaboration rate is not correctly predicted when, at the source, the steady state variables used for calculating it are different (see Figure 2.6, right column). In the case of nonlinear systems, we can no longer simply add and average equations defining individual agent's PFSMs for obtaining the PFSM representing the whole team. Approximations for obtaining closed formulas representing

the macroscopic PFSM depend on the specific nonlinearity involved in the coupling and this, in combination with the fact that in smaller swarm sizes, the continuous quantities of the state variables representing the macroscopic level are in stronger contrast with the integer quantities representing the microscopic level, generates discrepancies between these two types of models. A quantitative analysis of the discrepancies of prediction between microscopic and macroscopic models is beyond the scope of this work, but is partially addressed by Correll and Martinoli [64].

2.3.2.2 Steady state analysis

Since the system of DEs 2.11-2.15 is nonlinear, we cannot analyze it in z-space, as we did with the previous two-state sub-chain. Much as Lerman et al. have proposed for a time-continuous version of the same sub-chain [32], we perform a steady state analysis of the system in time domain.

By setting $N_s(i) = N_s^*$ and $N_g(i) = N_g^*$ for all i between $k - T_g$ and $k + 1$ in Equations 2.11-2.16 and substituting Equations 2.13 and 2.14 in Equation 2.11, we obtain:

$$0 = -p_{g1}(M_0 - N_g^*)N_s^* + p_{g2}N_g^*N_s^* + p_{g1}(M_0 - N_g^*)N_s^*\Gamma^* \tag{2.18}$$

$$N_g^* = N_0 - N_s^* \tag{2.19}$$

$$\Gamma^* = (1 - p_{g2}N_s^*)^{T_g} \tag{2.20}$$

$$C^* = p_{g2}N_s^*N_g^* \tag{2.21}$$

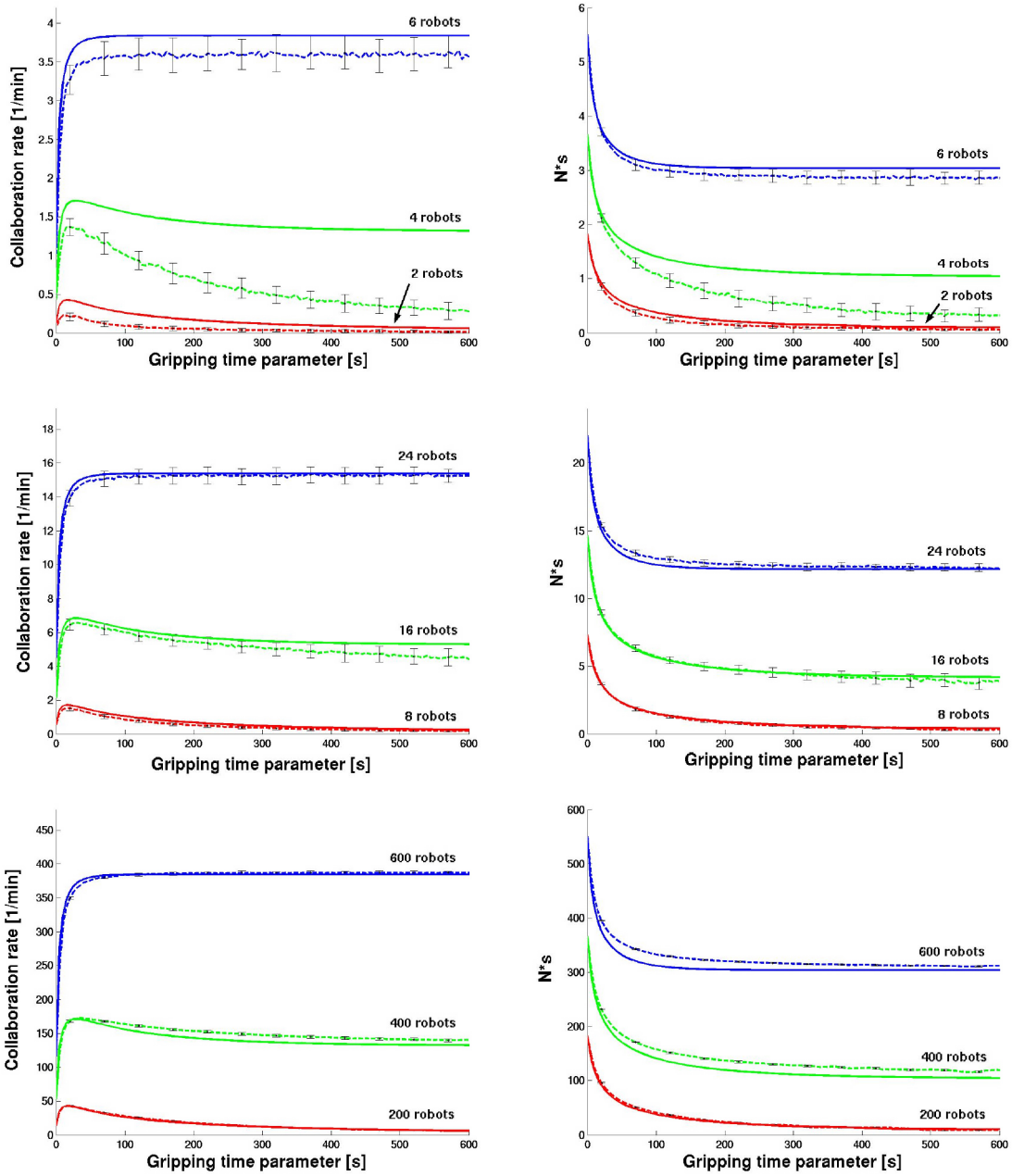


Figure 2.6: Comparison between microscopic and macroscopic predictions for the simplified stick pulling model depicted in Figure 2.5. The density of robots and sticks in each arena is invariant. Rows one to three represent results obtained in arenas of 40, 80, and 400 cm in radius, respectively. *Left column:* the collaboration rate as a function of the gripping time parameter for different swarm sizes. *Right column:* the steady state values of the average number of robots in the search state for different swarm sizes. Steady state values have been calculated over a window of 1000 s after at least $5\tau_g$ s from the beginning of the experiment. The error bars represent a mean standard deviation calculated as an average of the standard deviations measured on each run over the 1000 s time window.

First, we determine when the number of collaborations is maximized as a function of the number of robots in the search state (or in the grip state, respectively). To do this, we insert Equation 2.19 in Equation 2.21, perform a partial derivative over N_s^* and set the result equal to zero. C^* is maximal when $N_s^* = N_0/2$.

By inserting this result, $p_{g2} = R_g p_{g1}$, Equations 2.19 and 2.20 in Equation 2.18, we obtain the following transcendental equation:

$$0 = -(M_0 - N_0/2) + R_g \frac{N_0}{2} + (M_0 - \frac{N_0}{2})(1 - p_{g1} R_g \frac{N_0}{2})^{T_g^{opt}}. \tag{2.22}$$

Introducing $\beta = N_0/M_0$ and solving the equation for T_g^{opt}, we obtain

$$T_g^{opt} = \frac{1}{\ln(1 - p_{g1} R_g \frac{N_0}{2})} \ln \frac{1 - \frac{\beta}{2}(1 + R_g)}{1 - \frac{\beta}{2}}. \tag{2.23}$$

Equation 2.23 tells us that an optimal T_g exists if all the arguments of the logarithms are greater than zero. While this condition is met for the first logarithm in all our scenarios, the argument of the second logarithm depends on β and R_g. It can be demonstrated[6] that an optimum exists if and only if

$$\beta < \beta_c = \frac{2}{1 + R_g}. \tag{2.24}$$

For all other cases, the collaboration rate is a monotonically increasing and eventually saturating function of T_g. Figure 2.7 graphically demonstrates the meaning of Equation 2.24.

Equation 2.24 tells us that the bifurcation of the system (defined by whether an optimal

[6]While for $2/(1 + R_g) \leq \beta \leq 2$ the second logarithm of Equation 2.23 does not exist, more generally, if $\beta \geq 2/(1 + R_g), N_s^*$ in Equation 2.18 will be always greater than $N_0/2$, i.e., a suboptimal value. This can be easily demonstrated by introducing $\beta = N_0/M_0$ in Equation 2.18, solving for β and comparing with Equation 2.24.

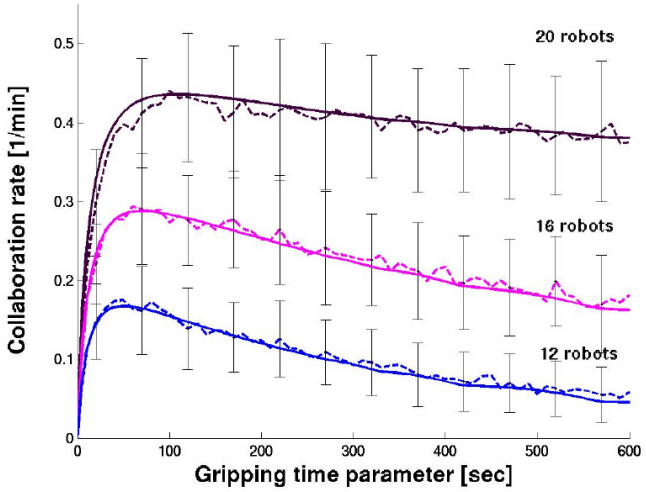
T_g exists) is a function of the collaboration parameter R_g. Notice that, for instance, if the collaboration is very difficult (i.e., R_g is very small), there could be situations where although we have a greater number of robots than sticks (see Figure 2.7(a), the 20-robot line), the optimal collaboration rate may still be achieved only with a specific T_g. In other words, when it is difficult to collaborate, in order to enhance the number of collaborations, it is worth abandoning the sticks after a while and probabilistically increasing the critical mass of robots working in another area of the arena. Although the precise team size at which the bifurcation happens in the real system cannot be correctly computed with Equation 2.24, this equation allows us to better situate intuitive considerations such as those presented in Section 2.3.2.1.

2.4 Model Construction: Full Stick Pulling System Model

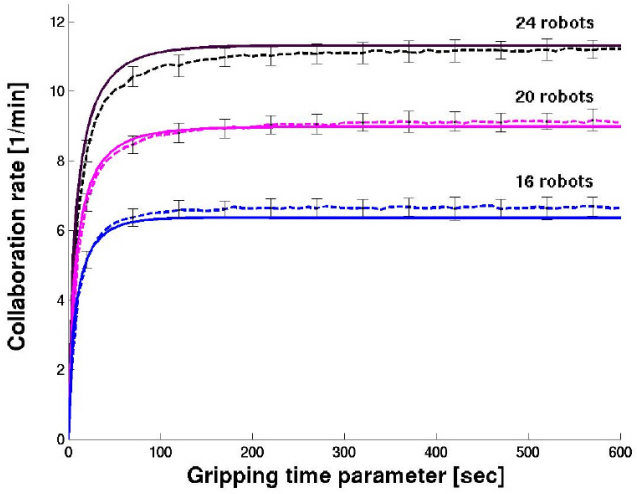
Having introduced the two key sub-chains characterizing the dynamics of the stick pulling experiment, we are now ready to analyze the full-system model. We can already imagine that the description of the full system at the macroscopic level will involve a system of nonlinear, coupled, time-delayed DEs.

As mentioned above, Figure 2.2(b) shows the PFSM of the full system. The macroscopic model of the full system can be formalized as follows[7]:

[7]The system of DEs presented here is consistent with the description presented in previous publications [33, 34] with one exception: the number of sticks available for grip1 has been decreased by a factor N_d since actually, as explained in Section 2.2.3, the experimenter replaces the stick in the hole only when the success dance of the robot has terminated. Although the collaboration rate changes only minimally with this correction, the current model is more faithful to the real experiment.



(a)



(b)

Figure 2.7: Graphical illustration of Equation 2.24 for an arena of 80 cm and 16 sticks (microscopic and macroscopic predictions overlapped). (a) $\beta < 2/(1+R_g)$, with $R_g = 0.035$. (b) $\beta < 2/(1+R_g)$, with $R_g = 1.0$.

$$\begin{aligned} N_s(k+1) &= N_s(k) - [\widetilde{\Delta}_{g1}(k) + \Delta_{g2}(k) + p_w + p_R]N_s(k) \\ &\quad + \widetilde{\Delta}_{g1}(k - T_{cga})\Gamma(k - T_{ga}; k - T_a)N_s(k - T_{cga}) \\ &\quad + \Delta_{g2}(k - T_{ca})N_s(k - T_{ca}) \\ &\quad + \Delta_{g2}(k - T_{cda})N_s(k - T_{cda}) \\ &\quad + p_w N_s(k - T_a) + p_R N_s(k - T_{ia}) \end{aligned} \tag{2.25}$$

$$\begin{aligned} N_a(k+1) &= N_a(k) - \widetilde{\Delta}_{g1}(k - T_{cg})\Gamma(k - T_g; k)N_s(k - T_{cg}) \\ &\quad + \Delta_{g2}(k - T_c)N_s(k - T_c) + \Delta_{g2}(k - T_{cd})N_s(k - T_{cd}) \\ &\quad + p_w N_s(k) + p_R N_s(k - T_i) - \widetilde{\Delta}_{g1}(k - T_{cga})\Gamma(k - T_{ga}; k - T_a)N_s(k - T_{cga}) \\ &\quad - \Delta_{g2}(k - T_{ca})N_s(k - T_{ca}) - \Delta_{g2}(k - T_{cda})N_s(k - T_{cda}) \\ &\quad - p_w N_s(k - T_a) - p_R N_s(k - T_{ia}) \end{aligned} \tag{2.26}$$

$$N_i(k+1) = N_i(k) + p_R N_s(k) - p_R N_s(k - T_i) \tag{2.27}$$

$$\begin{aligned} N_c(k+1) &= N_c(k) + \widetilde{\Delta}_{g1}(k)N_s(k) + \Delta_{g2}(k)N_s(k) \\ &\quad - \widetilde{\Delta}_{g1}(k - T_c)N_s(k - T_c) - \Delta_{g2}(k - T_c)N_s(k - T_c) \end{aligned} \tag{2.28}$$

$$N_d(k+1) = N_d(k) + \Delta_{g2}(k - T_c)N_s(k - T_c) - \Delta_{g2}(k - T_{cd})N_s(k - T_{cd}) \tag{2.29}$$

$$N_g(k+1) = N_0 - N_s(k+1) - N_a(k+1) - N_i(k+1) - N_c(k+1) - N_d(k+1) \tag{2.30}$$

where $T_{xyz} = T_x + T_y + T_z$, N_s represents the mean number of robots in the search state, N_a those in the obstacle avoidance state, N_i those in the interference state, N_c those in the stick-centering state, N_d those in the success dance state, and N_g those in the grip state. The Δ_{g2}- function can be calculated with Equation 2.14 and Γ-functions with Equation 2.15, while the Δ_{g1}-function is modified as follows:

$$\widetilde{\Delta}_{g1}(k) = p_{g1}[M_0 - N_g(k) - N_d(k)]. \tag{2.31}$$

Equations 2.25-2.30 can be interpreted in a way similar to that shown for Equations 2.11-2.12. For instance, Equation 2.25 indicates that the mean number of robots in search state at any time is decreased by the number of robots that transition to a grip state (grip1 and grip2) and by the number that start avoiding a wall or a teammate; N_s is increased by the number of robots that come back from a successful collaboration either as first or second robot (the success dance now has a duration greater than zero), those that come back from an unsuccessful collaboration, and those that finish their wall or robot avoidance maneuver. It is interesting to notice that all states other than search (the default behavior) and grip (calculated with the robots' conservation law) are characterized by an N_s factor

(either at the current iteration k or delayed) since they are simple delays like those used in the two-state system described in Section 2.3.1. However, much like the simplified stick pulling model described in Section 2.3.2, the coefficients with which the state variable N_s is multiplied are time-variable and functions of other state variables or N_s itself (see Γ and Δ-functions), thus generating nonlinear coupling between the equations.

As in the case of the two-state systems described in Section 2.3, all robots are in the search state at the beginning of the experiment, so the initial conditions for the DE system are $N(0) = [N_s(0)\ N_a(0)\ N_i(0)\ N_c(0)\ N_d(0)\ N_g(0)]^\mathsf{T} = [N_0\ 0\ 0\ 0\ 0\ 0]^\mathsf{T}$.

2.4.1 Results

We can now compare the results of our microscopic and macroscopic abstraction with those gathered with lower implementation levels, i.e., embodied simulations and real robots. Real robot experiments lasted about 20 minutes (duration of the on-board batteries) and were repeated three times each, while those carried out in the embodied simulator lasted 30 minutes (simulated time) and were repeated 10 times.

Figure 2.8(a) compares results obtained with real robots and embodied simulations. Although the real robot experiments were repeated only three times, as compared to the ten runs used for embodied simulations, real robots appear to achieve a slightly lower collaboration rate than their corresponding simulated agents. Differences between simulated and real robots' gripper modules are at the origin of this discrepancy. We will discuss this type of problem in detail in Section 2.6.1.

Furthermore, Figure 2.8(b) clearly shows that, in contrast to the macroscopic results, predictions delivered by the microscopic model are in good quantitative agreement with the data collected using the embodied simulator for all the team sizes.

Much like the results presented in Section 2.3.2.1, these problems are drastically attenuated as soon as we increase the number of robots and sticks, maintaining the same density of both items per unit area. Figure 2.9(a) shows good quantitative agreement among the three simulation levels simply by multiplying the robot and stick quantities by four and increasing the arena size to maintain the same object density, but without changing any implementation details. However, as soon as we multiply the quantities by 100 (see Figure 2.9(b)), we notice that the problem mentioned in Section 2.3.2.1 arises again, but even more accentuated. Section 2.6 will address again these problems more in detail.

2.4.2 Steady State Analysis

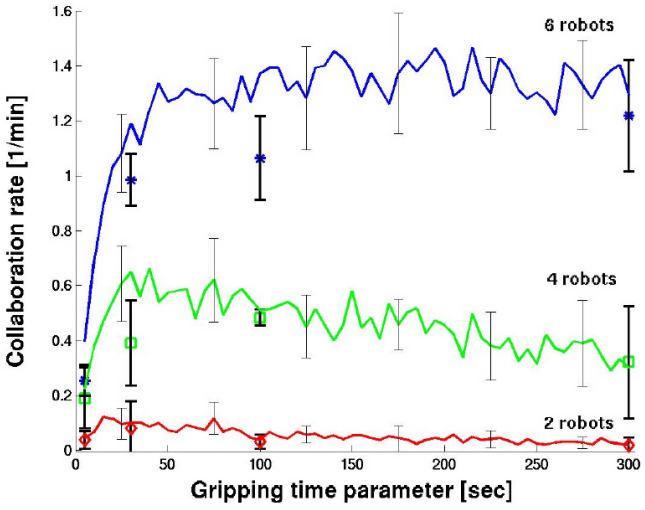
Like the simplified two-state system described in Section 2.3.2, the DEs describing the full system are nonlinear and, therefore, we must perform the steady state analysis in the time domain. Using the same method we adopted in Section 2.3.2.2 for Equations 2.25 and 2.30 and the approximation in time domain introduced in Section 2.3.1.2 for all the delay states of the full system (Equations 2.26-2.29), we obtain

$$0 = -p_{g1}(M_0 - N_g^* - N_d^*) + p_{g1}(M_0 - N_g^* - N_d^*)\Gamma^* + p_{g2}N_g^* \tag{2.32}$$

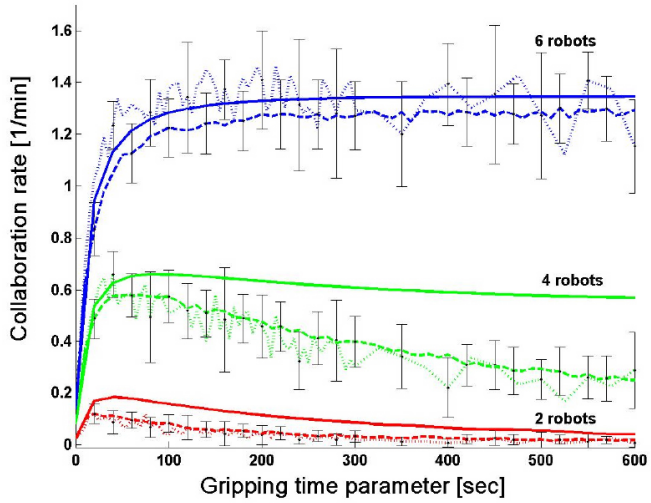
$$N_a^* = T_a N_s^*[p_{g1}(M_0 - N_g^* - N_d^*)\Gamma^* + 2p_{g2}N_g^* + p_w + p_R] \tag{2.33}$$

$$N_i^* = T_i p_R N_s^* \tag{2.34}$$

$$N_c^* = T_c N_s^*[p_{g1}(M_0 - N_g^* - N_d^*) + p_{g2}N_g^*] \tag{2.35}$$

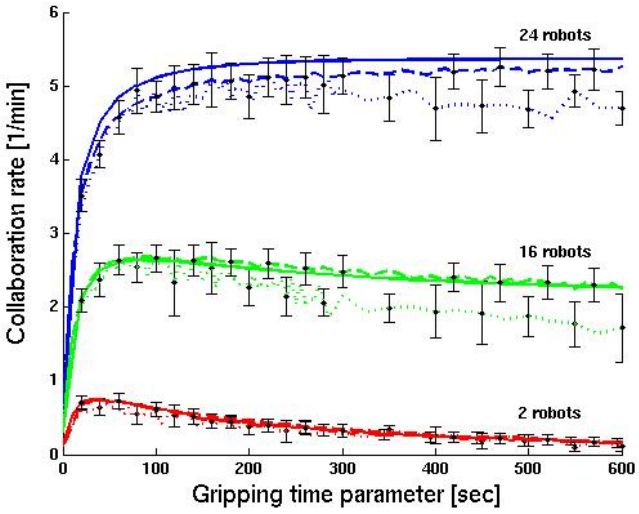


(a)

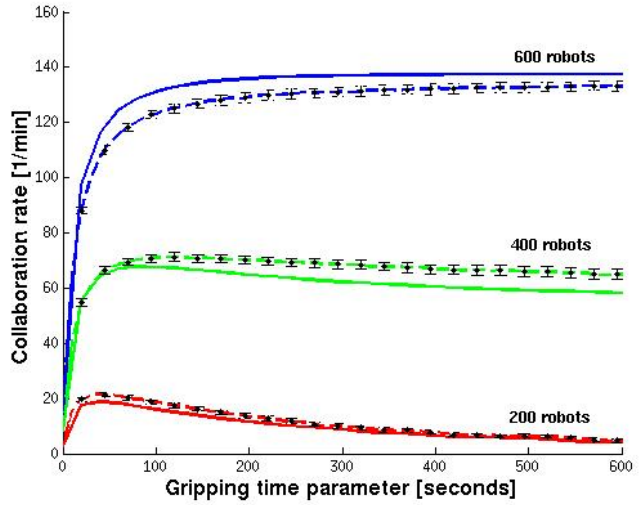


(b)

Figure 2.8: Collaboration rate as a function of the gripping time parameter for group sizes of two, four, and six robots in a 40 cm radius arena. (a) Results gathered using real robots (data for $\tau_g = [5, 30, 100, 300]$ s) and embodied simulations (solid line with $\tau_g = [0 : 5 : 300]$ s). (b) Microscopic (dashed line) and macroscopic (solid line) models' predictions overlapped with the embodied simulations' results (dotted line) for $\tau_g = [0 : 5 : 600]$ s.



(a)



(b)

Figure 2.9: (a) Results of embodied simulations, microscopic (dashed line) and macroscopic (solid line) models for 8, 16, and 24 robots, 16 sticks, and an arena 80 cm in radius. (b) Predictions obtained using microscopic (dashed line) and macroscopic (solid line) models for swarms of 200, 400, and 600 robots in an arena 400 cm in radius.

$$N_d^* = T_d p_{g2} N_s^* N_g^* \tag{2.36}$$

$$N_g^* = N_0 - N_s^* - N_a^* - N_i^* - N_c^* - N_d^* \tag{2.37}$$

The collaboration rate in steady state becomes

$$C^* = p_{g2} N_s^* N_g^* = p_{g2} N_s^* (N_0 - N_s^* - N_a^* - N_i^* - N_c^* - N_d^*) \tag{2.38}$$

Simulating the DE system 2.25-2.30 long enough until a stationary regime is reached and solving the equation system 2.32-2.37 are two alternative options for obtaining the full system's steady state vector $N^* = [N_s^* \ N_a^* \ N_i^* \ N_c^* \ N_d^* \ N_g^*]^\mathsf{T}$. Both of these operations must be performed numerically. Indeed, if we try to solve the equation system 2.32-2.37 analytically by substitution and introducing all the results in Equation 2.32 we obtain the following transcendental equation:

$$P_0(N_s^*) + P_1(N_s^*)\Gamma^* + P_2(N_s^*)\Gamma^{*2} = 0, \tag{2.39}$$

where P_0, P_1, and P_2 are second order polynomials in N_s^* whose coefficients are a function of all the system parameters, i.e., $p_{ij} = p_{ij}(M_0, N_0, p_{g1}, p_{g2}, p_w, p_R, T_a, T_i, T_c, T_d)$, i being the polynomial index ($i = 0, 1, 2$) and j being the term number in a given polynomial ($j = 0, 1, 2$).

Like Equation 2.18, Equation 2.39 can be solved for Γ (a quadratic equation here) and therefore also for T_g. However, unlike the analysis described in Section 2.3.2.2, determining the value of N_s^*, which achieves an optimal collaboration rate, implies we know what values

the delay state variables assume in steady state, as shown by Equation 2.38. Unfortunately, $N_a^*, N_i^*, N_c^*, N_d^*$, and N_g^* are, in turn, nonlinearly coupled with N_s^*, preventing us from finding an optimal value without solving 2.39 for N_s^*. Although we can approximate Γ with a McLaurin series ($\Gamma^* \approx e^{-T_g p_{g2} N_s^*}$), Equation 2.39 can be solved, to our knowledge, only numerically. This prevents us from formulating analytical expressions for the system bifurcation points (such as that reported in Equation 2.24) or superlinear-linear and linear-sublinear regime transitions (such as those reported in Ijspeert et al. [9] for the relative collaboration rate, i.e., the collaboration rate normalized over the total number of robots used in the experiment).

2.5 A Case Study: The Aggregation Experiment

To demonstrate the versatility of the methodology developed in [36] and presented above, I will briefly present our application of the methodology to a case study concerned with multi-robot aggregation. The task is to collect small objects, referred to as "seeds", in a square arena and to gather them in a single cluster[8]. In this case, the model was used for assessing the efficiency of a worker allocation algorithm. In contrast to most aggregation work to date [6, 4, 5], in which the size of the working team was kept constant during the whole aggregation process, this method allows workers to estimate demand and retire based on that estimate. The experiment uses three primary team performance measurements: the average cluster size, the average number of clusters, and the average number of active workers in the environment. Full details and an incremental construction of the model may be found in [37].

[8] Although these experiments are not intended to reproduce a biological system, they present several similarities with the nest cleaning and the clustering of dead ants performed by some ant colonies [65].

As the robots have only local sensing capabilities and do not exploit a fully connected communication network, there is neither central nor global coordination among robots. Collective coordination is purely probabilistic and happens based on local interactions, strictly following SI principles. The only communication available to the robots is implicit, stigmergic communication via the aggregation process. Because the robots do not have a global perception of the environment, they do not know when the task is finished. This motivates the need for a distributed task allocation mechanism to allow each individual to stop working based on its own estimate of the availability of work. Robots that decide to stop working can rest in a dedicated parking zone adjacent to the main arena.

The specific interest for modeling this particular aggregation experiment lies in the fact that, although the control algorithms used for aggregation and individual activity regulation are both deterministic (and therefore easy to model), the whole system depends on and exploits randomness to achieve its final state, i.e., a single cluster with all the robots resting. Indeed, the stochastic nature of the robot-environment interactions combined with noisy sensory readings automatically generates probabilistic transitions between system (i.e., the robots' and environment's) states. Randomness combined with irreversible elimination of isolated seeds prevents the aggregation process from getting stuck in multiple-cluster configurations. The irreversible decision to rest achieves a self-regulation of the swarm activity that gracefully follows the evolution of the aggregation process.

In the aggregation experiments presented here, the arena consists of an inner working zone of 80 x 80 cm^2, in which the cluster formation takes place, and a surrounding parking zone of 40 cm in width, where the robots that decide to stop working go and stay in an idle state to save energy. At the beginning of the experiment, 20 seeds, 1.7 cm in diameter and 2.5 cm in height, are randomly scattered in the working zone. The picture presented

Figure 2.10: A close-up of a simulated robot equipped with a gripper, ready to grip a seed; the corresponding setup in the embodied simulator (10 robots, 20 seeds, 178 x 178 cm^2 arena), in which the inner area represents the working zone and the surrounding area is the parking/resting zone where robots that decide to stop working stay idle; and a typical end state of the aggregation experiment, e.g., 5 hours of simulated time, in a 178 x 178 cm^2 arena.

in Figure 2.10 shows a 178 x 178 cm^2 working zone; this is chosen for clarity, as robots, seeds, working zone, and parking zone are more easily distinguishable in this picture than in a picture of an 80 x 80 cm^2 working zone.

Groups of one to ten embodied agents equipped with gripper turrets are used to collect and cluster the seeds. Agents can distinguish seeds from walls and other robots because of their thinness, simply using their six frontal proximity sensors. A floor-color sensor allows each agent to distinguish between the working and parking zones.

2.5.1 Aggregation Experiment without Worker Allocation

In most previous aggregation experiments [3, 4, 5, 6], the size of the working team was kept constant during the experiment. Furthermore, the experiments were monitored and terminated when a single cluster arose in the arena. Thus, the (destructive) effect of the lack of a mechanism to allow the robots to stop working was never clearly studied. Here we do not end the experiment when the robots create a single cluster of seeds, instead we let the experiment run for 10 hours. In keeping with previous works in cluster formation

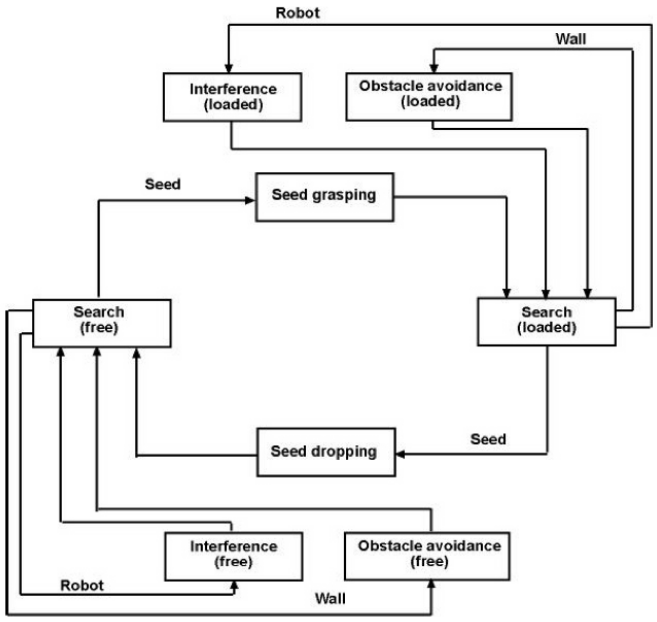
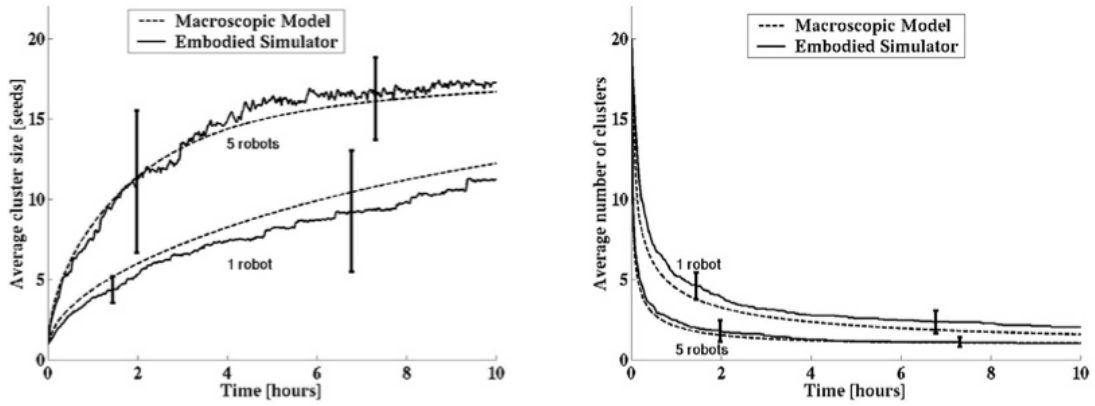
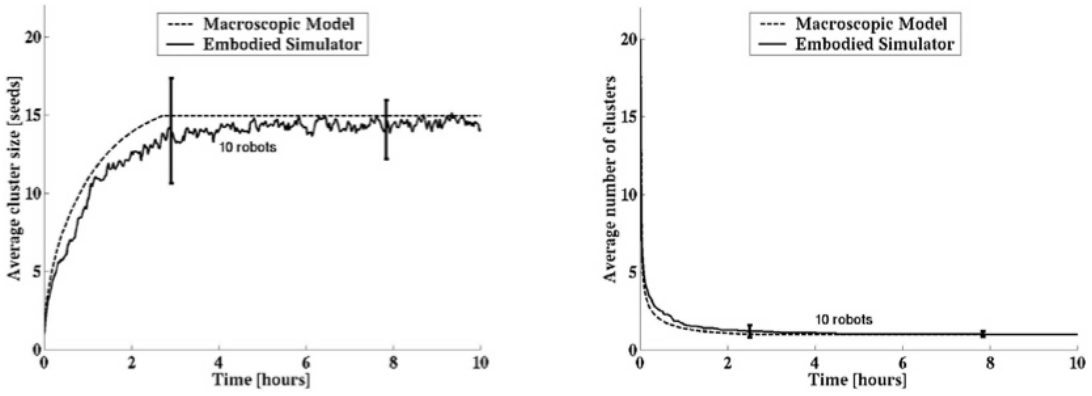


Figure 2.11: FSM representing the robot controller. Transitions between states are deterministically triggered by sensory measurements. This representation does not include the distributed worker allocation mechanism.



(a) (b)

Figure 2.12: Results of aggregation experiment with groups of 1 and 5 robots and 20 seeds in an 80 x 80 cm^2 arena. Team sizes are constant. (a) Average cluster size over time. (b) Average number of clusters over time.



(a) (b)

Figure 2.13: Results of aggregation experiment with groups of 10 robots and 20 seeds in an 80 x 80 cm^2 arena. Team sizes are constant. (a) Average cluster size over time. (b) Average number of clusters over time.

and aggregation [3, 4, 5, 31], two primary team performance measurements are used: the average cluster size and the average number of clusters.

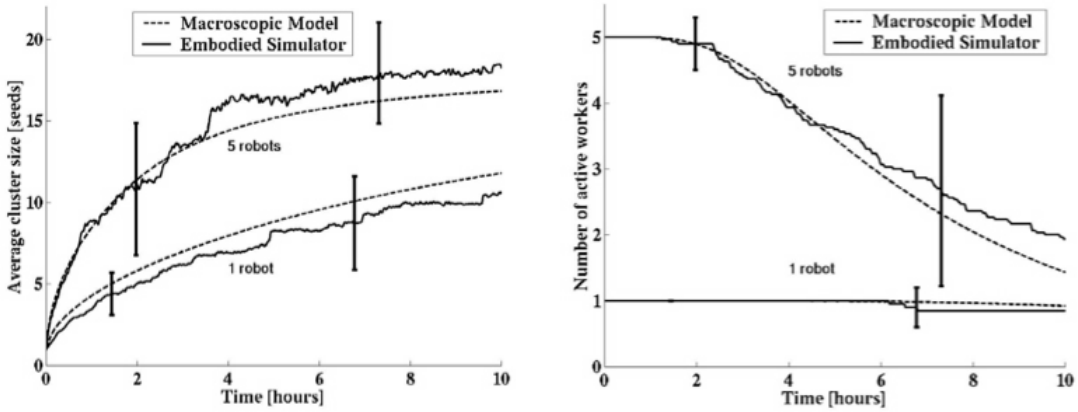
The behavior of each robot is determined by a simple hand-coded program that can be represented with a standard flow chart or a Finite State Machine (FSM), as depicted in Figure 2.11. The behavioral granularity shown in Figure 2.11 is arbitrary and is chosen by the experimenter so that the FSM captures all the details of interest. Each agent's behavior can be summarized with the following simple rules. In its default search behavior, the agent moves straight forward within the working zone looking for seeds. As in the stick pulling experiment, when at least one of its six frontal proximity sensors is activated, the agent attempts to identify whether the object it has encountered is a seed or a larger obstacle, such as another robot or a wall. If the agent is in front of a large object the agent avoids it with an appropriate maneuver. In the object is a seed, then if the agent is not already carrying a seed (if it is "free"), it grasps this one with the gripper, otherwise (if it is "loaded") it drops the seed it is carrying close to the one it has found; then in both cases, the agent resumes searching for seeds. With this simple individual behavior, the team is

able to gather objects in clusters of increasing size. A cluster is defined as a group of seeds whose neighboring elements are separated by at most one seed diameter. We note that, since agents can identify only the two ends of a cluster as seeds (as opposed to obstacles), clusters are built in a line.

Figures 2.12 and 2.13 show the model predictions and the simulation results of the aggregation experiment without the use of any worker allocation algorithm with groups of 1, 5, and 10 robots in an 80 x 80 cm^2 arena. Figure 2.12(a) and Figure 2.13(a) present the increasing average size of the clusters over time while Figure 2.12(b) and Figure 2.13(b), show the decreasing average number of clusters over time. Each experiment characterized by a different team size has been repeated for 30 runs and the depicted error bars represent the standard deviation over runs. Good agreement between the results collected at both implementation levels illustrates the reliability of the macroscopic model's predictions.

With a team of 10 robots (Figure 2.13(a)), the aggregation process clearly has two phases. In the first phase, the mean cluster size increases steadily from 1 seed to about 15 seeds and in a second phase, the mean cluster size remains, on average, constant around 15 seeds. Similarly, during the first phase, Figure 2.13(b) shows that the average number of clusters decreases asymptotically from 20 to about 1 and then remains close to 1 during the second phase of the aggregation process. This is clearly a side effect of the lack of a mechanism to allow workers to stop performing the aggregation task once a single cluster arises, which is confirmed by the fact that 10 hours into the aggregation process, the average cluster size built by the team of 5 robots (see Figure 2.12(a)) is larger than that obtained with the team of 10 robots.

The results in Figure 2.12 and Figure 2.13 can be explained by the fact that, once a single cluster arises only two manipulation sites remain in the environment (i.e., the two



(a) (b)

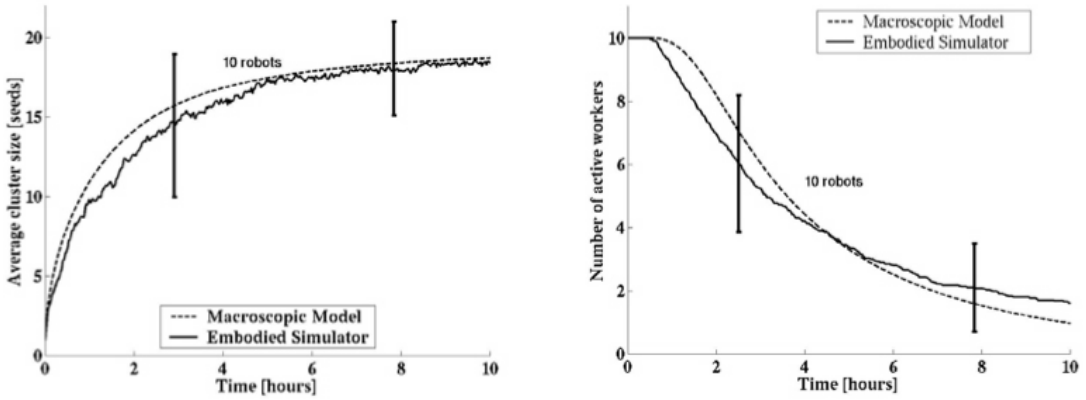
Figure 2.14: Results of aggregation experiment with worker allocation and groups of 1 and 5 robots and 20 seeds in an 80 x 80 cm^2 arena. (a) Average cluster size over time. (b) Average number of active workers over time.

tips of that cluster). Because the seed pick-up and drop-off probabilities are empirically very close, at any given time during the last phase of the aggregation process, on average, half of the active workers will be carrying a seed and the other half will not.

The results presented in Figures 2.12 and 2.13 justify the need for a distributed mechanism that enables each robot to switch off when the aggregation task is finished.

2.5.2 Aggregation Experiment with Worker Allocation

In threshold-based worker allocation systems, the "propensity" of any agent to act is governed by a response threshold. If the demand is above the agent's threshold then that agent continues to perform the task; conversely, if the demand is below its threshold then the agent stops performing that particular task. In the algorithm presented here, the time an agent spends before finding some work to accomplish (i.e., picking up and placing a seed) represents the agent's estimation of the demand stimulus associated with the aggregation task.



(a) (b)

Figure 2.15: Results of aggregation experiment with worker allocation and groups of 10 robots and 20 seeds in an 80 x 80 cm^2 arena. (a) Average cluster size over time. (b) Average number of active workers over time.

The worker allocation algorithm is as follows. When a robot has not been able to work for a reasonable amount of time, its propensity to accomplish that particular task is decreased. If the stimulus falls below a certain threshold (i.e., if the amount of time spent in search of work is above a given search time-out T_s), a deterministic switching mechanism prompts the robot to leave the working zone and rest in the surrounding parking zone for the remaining duration of the experiment. In other words, our worker allocation algorithm has been specifically designed for an irreversible process such as aggregation: robots become inactive from an active mode and not vice versa as seeds are irreversibly gathered in clusters of increasing size. Furthermore, a seed-carrying robot that decides to become inactive cannot do so until it finds an appropriate spot (i.e., one tip of a cluster) to drop the seed. Thus, with this simple algorithm characterized by a single threshold, each robot is able to estimate the aggregation demand locally and decide whether to work or rest with no need for a centralized supervisory controller. This task allocation mechanism is similar to that observed in some ant colonies [25] for which it has been shown that an individual performs a

task as long as the demand stimulus for the task, e.g., a pheromone concentration, exceeds the individual's threshold for that particular task.

The threshold distribution among agents could be performed in several ways. In this case, we use the simplest possible distribution: we assign the same response threshold to all the agents, obtaining in this way a homogeneous and fully scalable team from a control point of view. However, the resulting agents' behavior (rhythm of activity) is not identical since it is based on the local, private assessment of the aggregation demand, which is represented by the seed distribution in the environment. In other words, diversity in activity is created by exploiting the intrinsic noise of the system due to local perception and additional noise sources on simulated (miniature) sensors and actuators.

Figure 2.14 and Figure 2.15 present the outcome of the aggregation experiment using the proposed distributed worker allocation algorithm with teams of 1, 5, and 10 robots in an 80 x 80 cm^2 arena. Each aggregation run lasted 10 hours. All error bars represent the standard deviations among 30 runs. For all the results presented in this subsection, we hand-coded $T_s = 25$ minutes. This value of T_s is very close to the optimal threshold value obtained using a systematic search shown in Section 2.6.5. Figure 2.14 and Figure 2.15 show that, in contrast to the case without worker allocation, the average cluster size remains an increasing function of time for all three group sizes within the 10-hour experimental time window. When using a single worker, the average cluster size over time is the same as that obtained in the scenario of Section 5 and the robot usually does not leave the working zone; with a group of 5 workers, although the average cluster size over time is similar to that obtained in the scenario without worker allocation, here the average number of workers necessary to obtain the same results is substantially lower, as on average, only about half of the agents remain in the arena after 10 hours. The most important improvement comes

with a large team of 10 robots. In fact, with 10 workers, during the second phase of the aggregation, the average cluster size remains an increasing function of time, eventually reaching 19 seeds, nearly the largest value possible, while the number of active workers in the environment decreases such that after 10 hours only about 2 workers remain active.

2.6 Discussion

This section discusses the problems and subtle effects that arise in moving between levels of abstraction and explores the usefulness and limitations of this modeling methodology as a tool for optimization.

2.6.1 From Real Robots to Embodied Simulations

Although in general, as demonstrated in several other tasks [4, 5, 66, 17], the embodied, sensor-based simulator Webots has provided very faithful results, simulation is never reality. Several effects due to nonlinear physical laws, noise, and small heterogeneities among robots and components are simply neglected in simulation in order to reduce computational cost. In particular, when (collaborative) manipulation is performed with miniature robots like the Khepera, these effects can play a major role: grippers are usually crude, endowed with only a few degrees of freedom, and sensors are affected by a high level of noise.

In the specific case of the stick pulling experiment, we have observed (but never quantified) that the discrepancies between results obtained using the embodied simulator and those using real robots shown in Figure 2.8(a) have to be attributed to differences in the real gripper module and its corresponding embodied simulation. These differences between simulated and real grippers are twofold. First, reliability: while the simulated gripper never releases the stick unintentionally (in other words, the trigger for releasing is always either a

teammate's help or the internal time-out), the real gripper sometimes drops the stick early due to the noise in measuring the elevation of the arm. Since the presence of a teammate is assessed based on the measurement on the arm elevation sensor[9], if this measurement is noisy, the robot performing grip1 may erroneously conclude that a teammate is helping and release the stick, allowing it to drop into its hole again. The error in elevation is correlated with the arm PI controller and more probable when the arm is in full swing. Although the decision to release the stick is based on redundant sensory samples and checks, it still happens from time to time, but does not occur in Webots. Second, body solidity: in Webots 2, collision detection routines are 2-D and grippers never get entangled, instead they actually pass through each other, whereas entanglement is an occasional problem in the real robotic experiment. Possible fixes for these discrepancies include implementing more realistic sensor noise in the gripper in Webots and using a newer version of the Webots simulator (version 3 or higher), which implements more computationally intensive 3-D collision detection routines and should therefore eliminate the problem of gripper penetration at the price of a slower simulation.

2.6.2 From Embodied Simulations to Probabilistic Models

The method of abstraction presented here transforms an embodied, sensor-based agent in a much simpler agent. In the nonspatial, probabilistic models, an agent assumes a random position in the arena at each new simulation step instead of being characterized by a trajectory. The probabilistic agent has a perfectly centered, uniform, and precise range of detection for each object it may encounter in the arena, in contrast to the individual, heterogeneously distributed, noisy sensors available to the real robots and in the embodied

[9] A systematic positive error in the pre-established position is detected by the first robot when the second robot pulls up on the stick. See Section 2.2.3.

simulation. The probabilistic agent is characterized by an average speed instead of having a more complex kinematic controller: at lower levels of implementation, accelerations are tightly coupled to sensory readings, whereas the probabilistic agent is endowed with a behavior-based controller that can be represented by a precise PFSM whose state-to-state transitions follow either precise durations or are triggered by external events, but never happen with shorter or longer duration due to interrupts or because an event was detected by the sensors earlier or later.

In the next three subsections we will discuss the role of some of these important approximations in the accuracy of the predictions delivered by the probabilistic models.

2.6.2.1 Parameter calibration and behavioral granularity

As seen in Section 2.2.4, the models' parameters characterize the microscopic robot-to-robot and robot-to-environment interactions. They include probabilities of encountering specific objects, sometimes even considering a preferred angle of approach, and delays required by specific maneuvers whose details are uninteresting for the metric considered. This implies, for instance, an effort to summarize belts of noisy, perhaps unevenly spaced sensors as well as their relative detection and reactive control algorithms with an average detection area or capture an obstacle avoidance maneuver with a mean duration.

To calculate the probability of encountering an object or robot in the case studies presented, two different methods were used. The first, used in the stick pulling case study, bases the parameter calibration on simple geometrical considerations and systematic experiments with one or two real robots. The second method, similar to one proposed by Lerman and Galstyan in [28], used in the aggregation case study [37], is based on the area swept out by the robots' sensors in a given time step.

While these heuristic methods are certainly steps in the right direction, none of them has, thus far, taken into account the fact that model parameters, measured with systematic tests at lower levels of implementation, may also be characterized by measurement error and should, therefore, be introduced in the models as a mean value and corresponding distribution instead of an average value. To do this, we must take a step beyond our current mean-field approximation and develop an exact statistical formulation, describing the system a stochastic master equation instead of a rate equation [67]. We note that, considering the complexity of the models and the propagation of errors in nonlinear, time-delayed systems, this step will not be trivial.

Further difficulties may arise because of the behavioral granularity captured in the microscopic model. The robot controller used in the both case studies, developed prior to the modeling methodology presented here, can be approximated as an FSM, though certain routines (obstacle avoidance and interference) have been implemented with proximal controllers. Proximal controllers, in our case neural network-based controllers, tightly couple actuators with sensors without passing through a distal representation as, for instance, is the case for behavior-based implementations. Parameters used to describe the states corresponding to such routines (in our case, the duration of obstacle avoidance and interference as well as the probability of detecting an obstacle and a teammate) can still be measured in systematic tests with one or two robots, as mentioned above, even if this implies some inaccuracy. For predicting the collaboration rate, our chosen metric for the stick pulling task, and for predicting "high-level" metrics such as the average cluster size, this approximation is quite sufficient. Consider, however, a simpler system, for example, a controller consisting only of a searching mode and the obstacle avoidance and interference routines used in our proximal controllers. The description of a state as proposed here would probably no longer

be adequate for such controllers; temporal attractors in the state space rather than static state definitions may achieve better results.

2.6.2.2 Nonspatial models

The macroscopic modeling method proposed in this chapter is nonspatial, as it does not make use of the trajectories of the agents, the correlation between the positions occupied over consecutive time steps, or the spatial distribution of the agents resulting from their movement pattern and the environmental configuration. Therefore, this model cannot accurately describe swarm-based systems that are sensitive to the trajectories of the agents or their positions over time. An example of such a system is a team of rovers carrying out a coverage task in a planetary exploration mission. In such a case, the lifetime of each robot is crucial and sets a maximal boundary on the time within which the mission must be accomplished. In this case, the current model would have to be extended to take into account states defined by the position (and orientation) of the robot in order to be quantitatively accurate. Depending on the metric chosen, spatial distributions rather than detailed trajectory information may suffice to achieve this goal.

The modeling methodology assumes a uniform distribution of objects on the arena and no relevance of trajectories or robot spatial distribution to the chosen metric. As long as detection areas do not overlap between the objects placed in the arena (in this case, walls and sticks) and the metric does not specifically address spatiality, this assumption is correct. In order to verify this, we ran several stick pulling experiments characterized by different stick distributions (see Figure 2.16) using the embodied simulator. In each of these cases, the predictions of both models were as good as those shown in Figure 2.8(b), the microscopic model reaching quantitative agreement with the embodied simulations. If

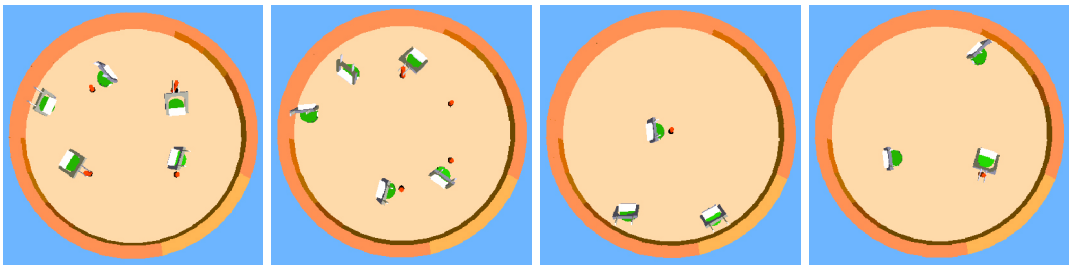


Figure 2.16: Four examples of implemented stick distributions.

needed, the current modeling methodology could be also easily adapted in order to take into account overlapped detection areas, especially for nonmobile objects. However, subtle effects due to robot clustering and mutual influence in search and manipulation activities could arise in densely populated scenarios and, as we will show in Section 2.6.2.3, these effects are more difficult to incorporate in the models. Specific robots' distributions (e.g., a specific pattern of movements at the arena's boundary) could also be easily introduced at the price of additional complexity (i.e., more states) in the PFSM(s).

In addition to a robot's position, its orientation may also play a role in the metric considered, particularly when either a specific sensor or actuator has a range comparable to the dimensions of the arena. For instance, in the stick pulling experiment, the introduction of directional communication for attracting robots [9, 68] and vision capabilities in a limited cone of view [68] generates additional difficulties for a nonspatial modeling methodology such as the one presented here. Usually, quantitatively correct predictions using nonspatial, probabilistic models can still be achieved, but not without using free parameters [9].

2.6.2.3 Overcrowded arenas

The modeling methodology achieves quantitatively correct predictions of nonspatial metrics based on the assumption that robots are, on average, homogeneously distributed in the

arena. As soon as this assumption is no longer valid, such as in an overcrowded scenario, the current methodology reaches its limitations and predictions are no longer quantitatively correct. Another way to explain why an overcrowded scenario breaks the methodology's assumptions is that the models as generated by the methodology are no longer Markovian in this situation. As the arena becomes crowded with robots, though the robots can still move around and collaborate, the next transition between controller states is contingent on each robot's trajectory and the trajectories of the surrounding robots.

In an overcrowded scenario, the robots form clusters with overlapping detection areas, which frees space around sticks and seeds that can, in turn, be exploited by other robots. To maintain Markovian assumptions in this situation, we would have to consider states relative to robot trajectories, which would result in an explosion of complexity of the models, if formulated in the same way we proposed in the methodology presented here.

Figure 2.17(a) illustrates this effect for the microscopic model in the stick pulling experiment. The plot shows a clear discrepancy between the microscopic model's and embodied simulation's optimal collaboration rate for group sizes greater than ten robots. While the embodied simulation results reflect this continuing collaboration mitigated by crowding, the microscopic model predicts ever-increasing performance. The continual increase here is due to the fact that the interference area represented by the robots cannot expand into sticks' and wall's detection areas and will therefore saturate to a maximal value corresponding to the free space available in the arena (see Equation 2.1). An alternative option, which allows the robots' detection area to grow until the calculated probability of encountering another robot is one, will also fail to deliver correct predictions since, at a certain point, the collaboration rate disappears [9, 33] (in the model, the robots can only execute interference maneuvering) while in the embodied simulation performance continues to be greater than

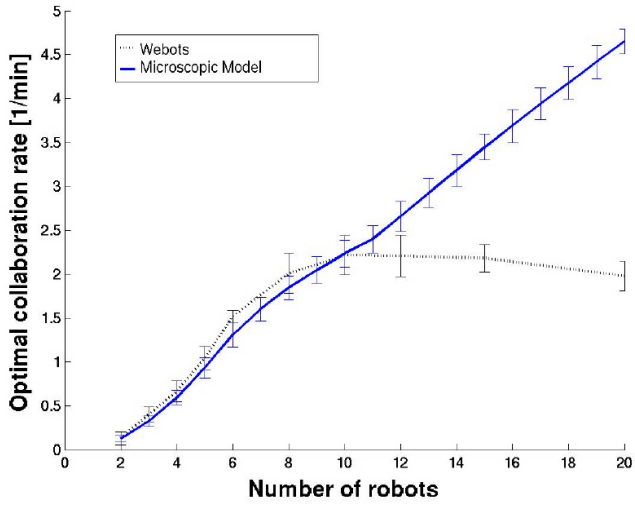
zero, as shown in Figure 2.17(a).

Figure 2.17(b) shows the problem from another perspective. The piecewise linear approximation we do in the model for the probability of encountering another robot per time step is also approximately correct for teams of up to ten robots, but then, in the saturation phase, the model's curve diverges from the likelihood measured using embodied simulations.

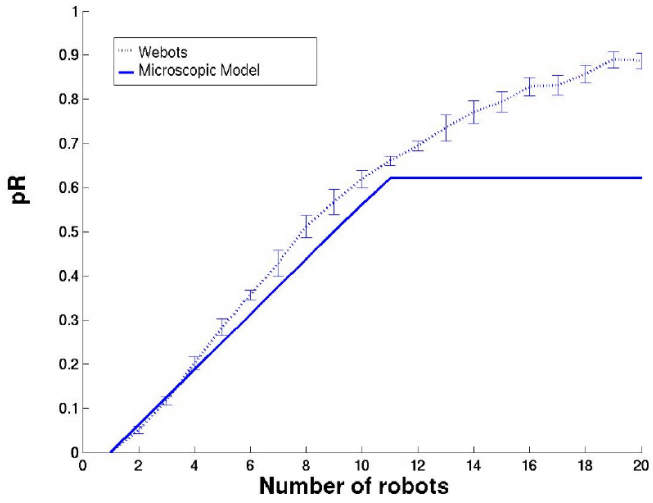
2.6.3 From Microscopic to Macroscopic Models

The microscopic and macroscopic models of a distributed manipulation experiment, although they rely on the same abstraction of the individual agent, may deliver slightly divergent predictions. The discrepancies are fundamentally due to the fact that typical models of a distributed manipulation experiment are constituted by nonlinear, often time-delayed DEs and that the average quantities predicted by the macroscopic models cannot simply be calculated from the linear combination of the individual PFSMs constituting the microscopic model. This already affects the predicted mean value of state variables, which in turn introduces more or less relevant discrepancies in the metric used for evaluating the swarm performance. For instance, the discrepancies between microscopic and macroscopic models in the collaboration rate (based on N_s and N_g state variables) appear to be slightly more important in the full-system model (Figure 2.8 and Figure 2.9) than in the simplified one (Figure 2.6, left column) because in the simplified model, inaccuracies in the N_s state variable can be directly compensated by the N_g variable, while this is only partially possible in the full-system model.

In the mathematical description of systems consisting of discrete numbers of agents, numerical effects can emerge in predictions. For instance, in the calculation of the size and the number of clusters in the aggregation experiment, using real numbers can result in



(a)



(b)

Figure 2.17: (a) Comparison of the prediction obtained using embodied simulations and the microscopic model in an overcrowded arena (up to 20 robots in an arena of 40 cm in radius). For each group size, the collaboration rate achieved after optimization of τ_g (systematic search) is plotted. (b) Probability of encountering a robot in the embodied simulator as compared to the linear approximation used in the modeling methodology.

obtaining cluster sizes or numbers of clusters smaller than unity but still greater than zero. Similarly, in the stick pulling experiment, the macroscopic model predicts a collaboration rate that is greater than zero for a single robot. These numerical effects may reduce the meaningfulness of the model's predictions, in particular with small swarm sizes.

Further discrepancies may have a combinatorial origin: the microscopic model correctly calculates the allocation of robots at limited, shared manipulation sites by keeping track of each modification of the environment generated by the actions of the robots. The macroscopic model instead summarizes the states of the robotic swarm and environment with expected value quantities and closed form expressions, a representation that is often only an approximation of combinatorial series.

Unfortunately, it is difficult to draw general quantitative conclusions on discrepancies between microscopic and macroscopic models at this point. Only after having carefully analyzed how errors propagate as a function of the nonlinearities characterizing a given system, will we be able to evaluate more precisely the effects of this further level of abstraction. Moreover, it is currently unclear whether or not we will be able to identify a set of nonlinearities systematically arising in distributed manipulation or, even more generally, swarm robotic experiments for which we could develop ad hoc but accurate error estimation procedures.

2.6.4 Capturing Randomness with Macroscopic Deterministic DEs

Both the stick pulling and aggregation experiments are permeated by randomness. Without randomness, robots would not spontaneously collaborate to extract a stick, aggregation of a single cluster could not be achieved, nor could such a simple distributed worker allocation algorithm be implemented to regulate team activity.

Qualitatively speaking, randomness decreases the efficiency of the robotic system but increases its robustness to deadlock situations and unfavorable initial conditions. System randomness has its origin in the local perception and in the concurrent, autonomous action of each robot as well as in the intrinsic noise of miniature sensors and actuators faithfully mimicked in the Webots simulator. Thus, a hand-coded, deterministic robot behavior is unavoidably transformed through the interactions between the robot and the environment and between the robot and its teammates into a stochastic behavior. Our macroscopic models, which at first glance may appear deterministic, capture the randomness of the system by using probabilities and probabilistic rates of occurrence of different events. In the rest of this subsection, we will present a few concrete examples of randomness-based mechanisms exploited in, and essential to, the distributed manipulation experiments described here.

The first example is the randomness in the robots' trajectories in both case studies. If the trajectory of each robot were predetermined and noise-free, a central supervisor with a view of the entire arena would be required to coordinate the movements of the robots in such a way that from each possible starting condition, the robots could carry out their task without getting entangled. Just having deterministic trajectories without a central planner with a global view of the arena or a powerful on-board navigation system (e.g., GPS) may allow some solutions, but robot coordination without a central supervisor becomes problematic.

A second example is concerned with the creation of a single cluster at the end of the aggregation experiment. Due to the mechanical constraints of the robots, the approaching angles for incrementing and decrementing the size of a cluster are almost the same. Therefore, without randomness, we could have deadlock situations (e.g., 2 large clusters consisting of half of the seeds initially scattered in the arena) in which one seed is picked up

from one cluster and dropped into the other, picked up again from this latter cluster and dropped, once again, into the original cluster, and so on. We could see this as a form of dynamic equilibrium different from the single-cluster situation. Such a scenario has never appeared in any of our aggregation experiments. This is explained by the fact that the independent (or uncoordinated) actions of all the robots in the arena generates relevant variations on the cluster sizes so that, each time any cluster is reduced to a single seed, that single-seed cluster is irreversibly (and deterministically) removed from the arena. Of course, this achievement was facilitated by the fact that the number of seeds was never much larger than the number of robots (the worst case studied is a 20 to 1 ratio). With a much higher seed-to-robot ratio we would probably have significantly greater difficulty systematically achieving the single-cluster result.

The third example is concerned with the core principle of the worker allocation algorithm presented in Section 2.5.2. Although all the teammates are endowed with the same activity threshold (i.e., a homogeneous team), the noise inherent in individual estimation of the aggregation demand due to local perception of the environment guarantees that robots deactivate at different times during the aggregation process. Thus, local perception and other noise sources create diversity in the decision to quit; as a consequence, the occurrence of deactivation increases gracefully over time as opposed to the sudden, collective withdrawal that would be seen if all agents were characterized by the same threshold and had a global perception of the environment. Randomness in combination with an irreversible, deterministic deactivation algorithm leads to the targeted situation in which all the robots are eventually inactive.

2.6.5 Modeling as Tool for Optimization

The modeling methodology presented here can certainly be useful also as an optimization tool. The models presented here deliver results in time lapses that are at least four orders of magnitude shorter than a corresponding embodied simulation. The abstraction in general allows researchers to understand the role of key system parameters, generalize and analyze underlying principles, and sometimes even enables mathematical tractability and resituates intuitive considerations.

In assessing performance in the stick pulling experiment, the models are used as an optimization tool as the gripping time parameter τ_g is exhaustively explored.

In the aggregation experiment, although we have not investigated whether or not an optimal threshold T_s can be derived analytically, we used the full macroscopic model described in [37] to systematically search for the optimal activity threshold. The advantage of using the full model is that, although the complexity of the system may render the model analytically intractable, the optimization results obtained using the model are still quantitatively correct. The results are presented in Figure 2.18.

Figure 2.18 presents the average cluster size obtained after 10 hours of aggregation for different values of the activity threshold T_s. This figure illustrates that for a group of 10 robots, an activity threshold value of about 27 minutes provides the optimal cluster size at time $kT = 10$ hours. On one hand, when the threshold value is lower than 27 minutes, the workers stop performing the task too early, before all the seeds have been gathered in a single cluster. On the other hand, when the threshold value is larger than 27 minutes, the robots keep working even after a single cluster has been created and, as a consequence, after 10 hours some seeds are still being carried around by some active agents, decreasing the efficiency of the team.

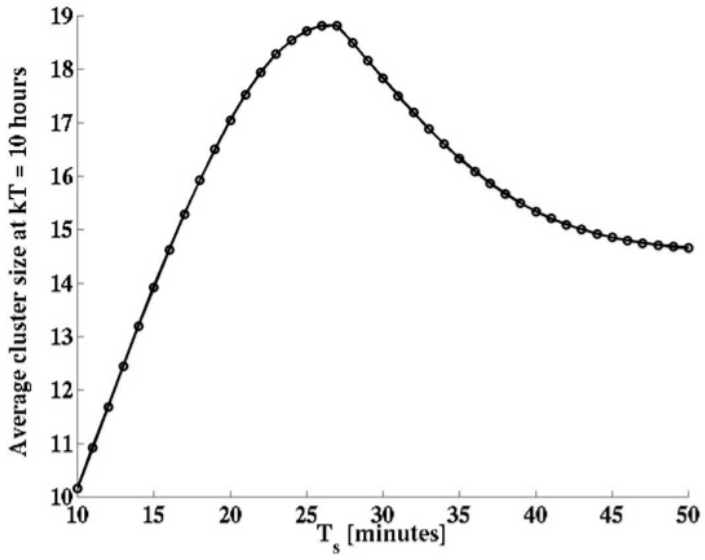


Figure 2.18: Macroscopic model's prediction of the average cluster size at time $kT = 10$ hours as a function of the activity threshold T_s. Experiments conducted in an 80 x 80 cm^2 arena with a group of 10 robots and T_s varying from 10 to 50 minutes with a 60-second incremental step.

Although the objects manipulated (fixed stick vs. transportable seeds), the metric used for assessing the swarm performance (average collaboration rate vs. average cluster size), and the significance of the threshold (a time-out regulating how long a robot waits for help vs. a time-out regulating the whole robot activity), the stick pulling experiment and the aggregation experiment with worker allocation share a few important similarities in their dynamics that are outlined, in particular, at the macroscopic model level. Some of them have been pointed out in Section 6 but we believe that the similarities between the profile of Figure 2.18 and, for instance, Figure 2.7(a) are striking. Describing the two experiments at a high level, we can say that in both of them there is a dynamical process taking place in which a specific robot's control parameter characterizing a threshold-based mechanism plays a crucial role. We can say that in both experiments the optimization goal is to find the right threshold (or the right distribution of thresholds) so that the robotic dynamics of the system

can be optimally coupled with the dynamics of the task accomplishment, which in turn is constrained by the robot and environmental features. In both cases, the axis defined by all possible threshold values presents an optimum and, therefore, the model-based optimization procedure for both experiments is also similar.

The models' scope as a tool for optimization is constrained, however, by the initial system design. In swarm systems, which base their principles of self-organized, collective behavior on multiple interactions among individuals and between individuals and environment, low-level system parameters such as the sensory configuration, body shape, proximal control parameters characterizing reactive behavior, and individuals' heading and positions may also play an important role, which we may neglect if we try to optimize the system at higher level. In other words, searching the optimal solution in a larger parameter space, including also low-level hardware and software parameters, may achieve better results than those explored with a more or less abstract model. Two concrete examples may illustrate this point.

A first example is concerned with collective plume tracing [17]. In this experiment, due to the highly stochastic nature of robot-plume interactions, the unavailability of suitable plume models for our specific task conditions, and the relevance of robots' headings and positions, we have thus far only been able to create models of the whole system characterized by several free parameters (see Hayes [69] for details), although all of them have a clear intuitive meaning. Despite being limited to only qualitative system optimization, Hayes was remarkably successful in optimizing the system using an ad hoc reinforcement learning algorithm combined with embodied simulations [18].

A second example comes from additional experiments we performed in the framework of the stick pulling experiment. As Ijspeert et al. [9] have shown using systematic search and

embodied simulations and as Li et al. [70] have shown using a simple learning algorithm combined with the same microscopic model presented here, a homogeneous swarm may not necessarily achieve the best performance. Subswarms of specialists in grip1 and grip2 may, for instance, outperform an optimized homogeneous swarm, depending on the ratio of robots to sticks. Generally speaking, at the macroscopic level, either we know the number of castes in a heterogeneous team in advance and we introduce a new set of DEs for each new type of agent caste involved in the system, or it would be impossible to explore heterogeneous solutions. Therefore, to explore heterogeneous solutions, microscopic models combined with machine learning algorithms appear to be a more efficient approach than macroscopic models.

Finally, as shown in Section 2.4.2, even if we are able to produce quantitatively correct analytical models, nonlinearities and the complexity of a real system, even a simple one as that used in the stick pulling experiment, often prevent us from going further in the analysis (and indirectly in the optimization process) with the mathematical tools currently available.

2.7 Collaborators' Continued Work

Since the publication of the SI work on which I was co-author [33, 34, 36, 37, 68], collaborators have made several notable improvements to the modeling methodology presented here.

In [42], Correll and Martinoli apply the models to the turbine inspection problem mentioned in Section 1.2 and present an improved method for the characterization of the models' parameters based on encountering rates, as suggested by Lerman and Galstyan in [28]. In another advance in calibration methods, Correll and Martinoli present a technique for sys-

tem identification [64], in which the system dynamics are derived from the individual robot controller and model parameters are estimated, then optimized, based on experimental data, eliminating the need for additional orthogonal calibration experiments. The result is an accurate predictive macroscopic model for systems that might otherwise be underdetermined, giving insight as to which parameters in the system govern the swarm dynamics.

Collaboration was introduced to the SI turbine inspection task in [44], allowing robots to serve as beacons for their teammates, biasing each others' inspection routes. In seeking an optimal collaboration policy for minimizing time to completion of inspection and minimizing energy consumption, Correll and Martinoli find a bifurcation similar to that seen in the stick pulling experiment, showing that a communication-based collaboration policy used towards the end of the inspection process can decrease the time needed for task completion, but only if the ratio of robots to turbine blades is greater than or equal to 1 [45]. This paper also applies methods from optimal control theory to optimize robot controller parameters.

2.8 Conclusion

This chapter presented a methodology for generating nonspatial, probabilistic microscopic and macroscopic models of swarm robotic systems. The methodology was explicitly designed for distributed manipulation experiments and is supported with two case studies, the stick pulling experiment and the aggregation experiment, belonging to this class. We have shown that models can deliver not only quantitatively correct predictions in periods at least four order of magnitude shorter than other popular simulation tools, such as sensor-based, embodied simulation, but can also allow for better understanding of the system properties and, in some cases, even enable mathematical analysis of the system.

We have also discussed several difficulties one may encounter in moving from one level of

implementation (real robots, embodied simulations, microscopic and macroscopic models) to a more abstract one. It is worth noticing that all these abstraction steps were hand-coded, exploiting a combination of engineering, heuristic, and systematic tests. Although in this chapter we did not compare the results of the models with those obtained at lower-level implementations using nonparametric statistical tests (something done in previous, related publications [62, 4]), we do not believe that we will gain further insight into how to resolve the difficulties in moving from one level of implementation to another in this way. However, statistical tests combined with algorithms that can systematically explore possible abstractions and verify their statistical impact on a chosen metric may allow us to achieve fundamental breakthroughs in the modeling methodology.

Chapter 3

Multi-robot Boundary Coverage

This chapter explores the *multi-robot boundary coverage problem* introduced in Chapter 1.2, wherein a group of k robots must inspect every point on the boundary of a two-dimensional environment. Section 3.1 describes the boundary coverage problem and the modeling assumptions. Using a simplified sensor model, this inspection problem is converted to an equivalent graph representation. Two methods for graph construction are presented. The first method, termed the *visibility region (VR) method*, is described in Section 3.2 and was presented in my first paper addressing the multi-robot boundary coverage problem [71]. It takes an intuitive approach to the graph construction problem, but results in an unnecessarily complex graph. The *boundary following (BF) method* is described in Section 3.5 and is the result of several improvements to the method described in Section 3.2. The BF method yields more efficient inspection paths, a more compact graph representation, and can handle environments containing nonconvex objects. Once a graph representation of the task has been constructed in which the undirected, connected graph G has a subset of edges E_R, every one of which must be traversed for inspection to be complete, the boundary coverage problem can be posed as the *k-Rural Postman Problem* (kRPP), a graph edge coverage problem. Section 3.3, presents a constructive heuristic to solve the kRPP. The robots use the solution to plan their inspection routes. I then discuss some simple bounds

on the path lengths traversed by the participating robots and prove the completeness of the boundary coverage algorithm. In Section 3.4, simulations illustrate the path planning algorithm's performance and characteristics using graphs generated with the VR method. Section 3.6 compares and contrasts the two graph construction methods and the resulting effects on the performance and capabilities of the path planning algorithm.

3.1 The Boundary Coverage Problem

The task is to be carried out in a bounded two-dimensional environment that is populated by N objects, $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$. Each object boundary is assumed to be a piecewise smooth, closed, convex curve. This convexity assumption will be relaxed in Section 3.5. The collective boundary of the objects is termed the "boundary" of the environment, $\partial\mathcal{O}$. The location and boundary geometry of each object to be inspected is assumed to be known a priori.

The inspection will be carried out by a group of k identical holonomic point robots, each equipped with an accurate scheme for localization as well as an omnidirectional "inspection sensor." (With an additional path planning step to determine the necessary sensor orientation at each robot pose, a steerable sensor with a limited angle of view could also be used.) The inspection sensor can measure phenomena of interest from a distance up to r_{max} from the boundary. A point p on the boundary is thus considered inspected when a robot's path intersects the imaginary line normal to the boundary at p and the distance from p to that point of intersection is less than or equal to r_{max}.

All robots involved in the inspection task deploy from a common "depot" location at the beginning of the inspection period; they will return to the depot when their inspection task is complete. At least one robot must inspect each point of the boundary at least

once during the group's inspection tour. The division of the task among the robots is determined by a centralized, supervisory agent. Once it has received its assignment, each robot calculates its inspection route, given by a series of waypoints determined by the constructive heuristic. Each robot has a limited communication capability so that it may receive its task assignments.

To develop a set of paths whose traversal will complete the inspection task, the sensor's visibility constraints are taken into account. Object $\mathcal{O}_j$'s *visibility region* consists of the set of individual robot poses in freespace for which the distance to the nearest point on the object boundary is less than or equal to r_{max}. The outer perimeter of each object's visibility region is a curve displaced a distance r_{max} normal to the object boundary, generally called an "offset curve."

The intersection of multiple objects' visibility regions are called *multiview regions*: from every pose within such a region the robot is a distance less than r_{max} from more than one point on $\partial\mathcal{O}$, and thus may inspect these points simultaneously. The routing strategy takes advantage of multiview regions when they arise, as simultaneous sensing of multiple object boundaries may reduce overall travel.

A *boundary segment* is a connected interval of $\partial\mathcal{O}$. Boundary segments for which each constituent point can be inspected from within a multiview region are called *interior boundary segments*, the union of which is $\partial\mathcal{O}_{int}$. All other intervals of $\partial\mathcal{O}$ are referred to as *exterior boundary segments*, the union of which is $\partial\mathcal{O}_{ext} = \partial\mathcal{O} \backslash \partial\mathcal{O}_{int}$.

A continuous sequence of poses within a visibility region is a *visibility path* for a boundary segment if and only if every point in the boundary segment will have been viewed when a robot has assumed every pose in the visibility path. For the sample environment shown in 3.1(a), this terminology is illustrated in Figures 3.1(b) and 3.1(c).

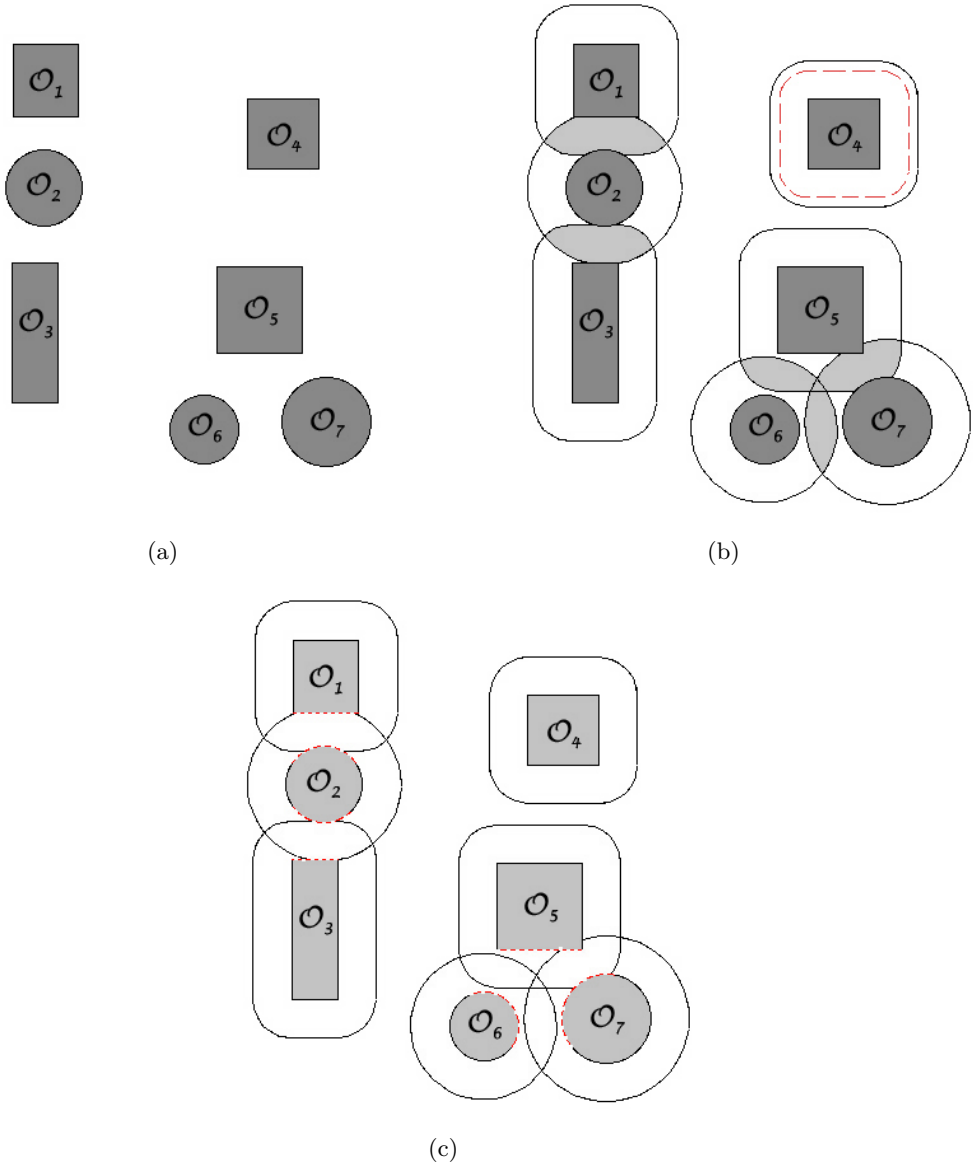


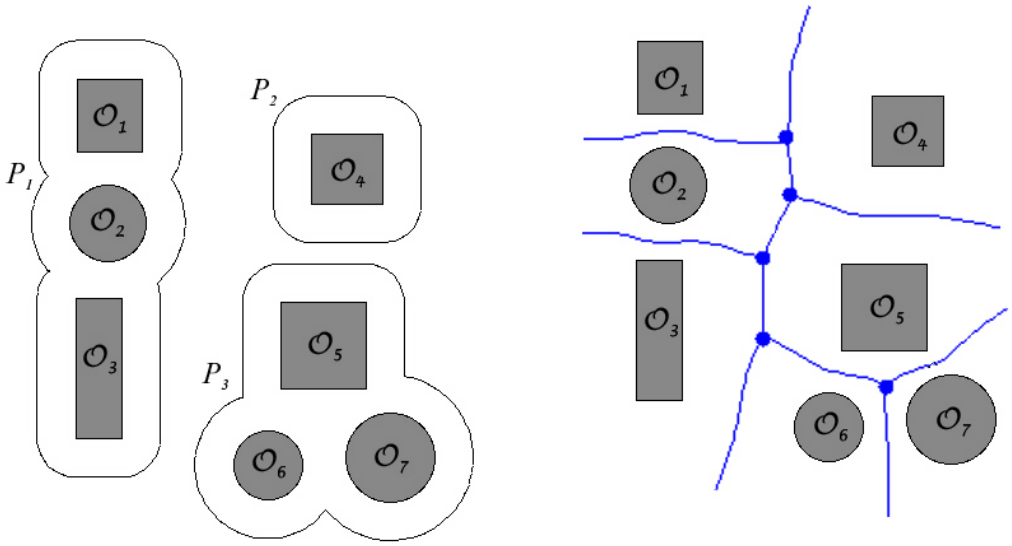
Figure 3.1: (a) Sample environment used to illustrate terminology introduced in Section 3.1 and the VR graph construction method in Section 3.2. (b) The boundaries of *visibility regions* are shown with a solid outline. *Multiview regions* are shaded. A possible *visibility path* for the inspection of the boundary of object $\mathcal{O}_4$ is shown with a dashed line. (c) *Exterior boundary segments*, $\partial\mathcal{O}_{ext}$, are indicated with a solid object boundary, while *interior boundary segments*, $\partial\mathcal{O}_{int}$, are indicated with a dashed object boundary.

3.2 A Graph Representation for Boundary Coverage: The Visibility Region Method

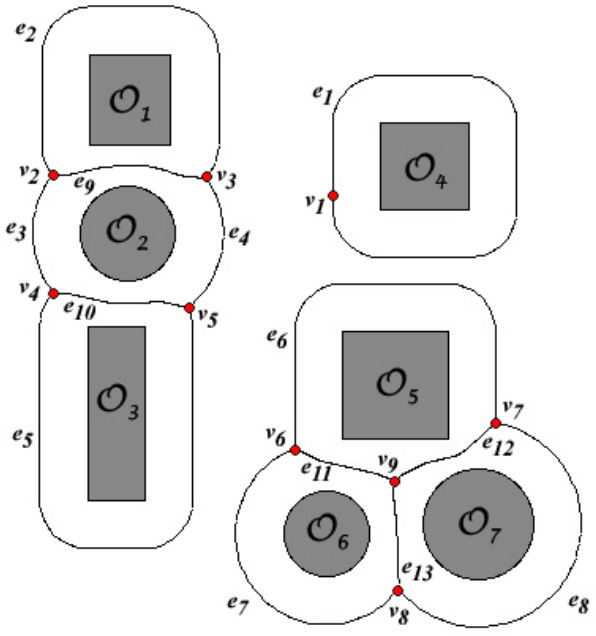
The graph's edges come in two varieties: required inspection edges, E_R, and connectivity edges, E_C. Each required edge represents an equivalence class of visibility paths, one of which the robot must traverse in order to inspect the associated boundary segment. This representation allows flexibility in navigation implementations, as individuals can avoid one another by taking different paths while "traversing" the same graph edge. The end-points of each edge are graph vertices, which physically correspond to terminal points for the equivalence class of visibility paths where a robot may transition locally from a path represented in one edge to a path represented in another. To simplify the construction of the graph representation a single, easily defined "preferred visibility path" for each edge is specified. The cost function assigns each edge a weight equal to the length of the preferred path along that edge, though other cost functions may be used.

The final result of the construction is an undirected, connected graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$, where V is the set of the graph's vertices, E is the set of its edges, and the cost function c assigns weights to the edges $E_R \subseteq E$. The remaining edges $E_C = E \backslash E_R$ represent a collection of paths that provide routes to the depot location and routes between remote areas where inspection is required.

For simplicity, this method assumes a preferred sensing distance of $r = r_{max}$. This constraint is imposed to make the construction of G easier by limiting the geometry of initially considered paths to well-defined offset curves.



(a) (b)



(c)

Figure 3.2: (a) Inspection regions $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$ for the sample environment. (b) Local components of the GVG. (c) Construction of edges in E_R.

3.2.1 Determining E_R: Edges Required for Inspection

Let S_r denote the set of offset curves that are displaced a distance r normal to each object boundary. For $r = r_{max}$, these curves represent the perimeter of each object's visibility region, as illustrated in Figure 3.1(b). Let $\mathcal{P}$ denote the union of the regions in freespace that are bounded by these curves. The region $\mathcal{P}$ will consist of one or more disjoint regions, $\mathcal{P}_i, i \in \{1, \ldots, m\}$, termed *inspection regions*. This is illustrated in Figure 3.2(a) for the sample environment. The boundary of $\mathcal{P}$, $\partial\mathcal{P}$, is comprised of m constituent closed curves, $\partial\mathcal{P}_i, i \in \{1, \ldots, m\}$. By construction, every point in $\partial\mathcal{O}$ is contained within the region $\mathcal{P}$. Every point in every object's visibility region and also every point in every object's visibility region within distance r_{max} of the boundary is contained within $\mathcal{P}$.

The edges in E_R are constructed as follows:

3.2.1.1 Edges outside multiview regions

Lemma 3.2.1. *Traversal of all curves in $\partial\mathcal{P}$ will result in the inspection of all exterior boundary segments, $\partial\mathcal{O}_{ext}$.*

Proof. Given the preferred sensing distance r_{max}, the preferred visibility path for an exterior boundary segment is an offset curve displaced a distance r_{max} normal to that boundary segment, which is, by construction, an interval of one of the curves in S. Exterior boundary segments cannot be inspected from inside a multiview region, therefore the portions of the offset curves defining the preferred visibility paths for exterior boundary segments consist of those curve segments that are a distance d, $r_{max} \leq d$ from every object boundary. By construction, the visibility paths for the exterior boundary segments are the portion of $S_{r_{max}}$ that is described by $\partial\mathcal{P}$. A robot traversing $\partial\mathcal{P}$ will pass within distance r_{max} of every point in each exterior boundary segment, thus the traversal of $\partial\mathcal{P}$ will result in the

inspection of each point in $\partial\mathcal{O}_{ext}$. □

Required edges are constructed such that every point in $\partial\mathcal{P}$ is included in a visibility path corresponding to an edge in E_R. To construct these edges, each disjoint region $\mathcal{P}_i$ is analyzed in turn.

If $\mathcal{P}_i$ incorporates no multiview regions, $\mathcal{P}_i$ must contain a single object, $\mathcal{O}_j$. The preferred visibility path for $\mathcal{O}_j$ is the entirety of $\partial\mathcal{P}_i$. In this case, a vertex is placed on the preferred visibility path at an arbitrarily chosen point. A single self-looping required graph edge of weight equal to the length of the object's preferred visibility path, $\partial\mathcal{P}_i$ is added to G. In Figure 3.2(c), vertex v_1 and edge e_1 illustrate this type of vertex and edge.

The offset curves $S_{r_{max}}$ intersect at points that are a distance r_{max} from two or more objects, allowing a robot to simultaneously inspect points on multiple boundaries. Each of these intersection points is, therefore, part of a multiview region. When these intersection points fall on $\partial\mathcal{P}$, they should be included in the preferred visibility path for an exterior boundary segment, as they represent points from which a robot may access a multiview region and move on to inspect an interior boundary segment. As these are the only points on $\partial\mathcal{P}$ that are part of a multiview region, they present a convenient choice for placement of graph vertices joining edges representing the inspection of exterior boundary segments with those representing interior boundary segments.

Thus, if $\mathcal{P}_i$ incorporates one or more multiview regions, consider $\partial\mathcal{O}_{\mathcal{P}_i}$, the union of the object boundaries contained within $\mathcal{P}_i$, in terms of the constituent exterior and interior boundary segments. Vertices are added to G at the points on $\partial\mathcal{P}_i$ that coincide with points where curves in S intersect one another. Required edges are added between these points representing the preferred visibility paths for the exterior boundary segments, the interval of $\partial\mathcal{P}_i$ between the vertices. In Figure 3.2(c), vertices v_2 through v_8 and edges e_2 through e_8

illustrate this type of edge.

3.2.1.2 Edges inside multiview regions

The inspection of interior boundary segments contained in $\mathcal{P}_i$ that also incorporate a multiview region remains to be considered. A convenient choice for the preferred paths through the multiview region is the local components of the *Generalized Voronoi Graph (GVG)* [72], a graph whose edges consist of points that are equidistant from the two nearest objects. Vertices in the GVG arise where edges meet, i.e., at points that are equidistant from three or more objects. Local components of the GVG for the sample environment are illustrated in Figure 3.2(b). Using these local components, robots are routed through multiview regions with a preference for paths whose constituent poses are equidistant from the boundaries currently being inspected. For every point on the GVG within a multiview region, at least two boundary points can be sensed from a single robot pose.

The graph vertices placed at the terminal points of the visibility paths for the exterior boundary segments in $\mathcal{P}_i$ lie on the GVG by definition, as they are equidistant from two or more points on $\partial\mathcal{O}$. Edges within multiview regions are established by adding to G all edges and vertices of the GVG contained within the union of all multiview regions. Weights equal to the length of the corresponding GVG path are assigned to each edge. Edges only partially contained in a multiview region are truncated where they exit the multiview region. In Figure 3.2(c), vertex v_9 and edges e_9 through e_{13} illustrate this type of vertex and edge.

Lemma 3.2.2. *Traversal of all edges of the GVG contained within the multiview regions results in the inspection of all interior boundary segments, $\partial\mathcal{O}_{int}$.*

Proof. Every point on every edge of the GVG is equidistant from the two nearest points on the boundary $\partial\mathcal{O}$. For points on the GVG within a multiview region, this distance is

less than or equal to r_{max}. By definition of an interior boundary segment, for every point, p_{int}, on every interior boundary segment, there exists a point, p_{GVG}, on a local GVG edge contained within a multiview region that falls within distance r_{max} of it. Thus, if all edges of the GVG that lie within multiview regions are traversed, a robot will pass within distance r_{max} of every point in every interior boundary segment, resulting in the inspection of each constituent point. □

3.2.2 Determining E_C: Edges Providing Connectivity

The group of required edges associated with boundary inspection in each inspection region $\mathcal{P}_i$ forms a distinct connected subgraph of G. The possibly disjoint components of this construction must be connected to allow for robot transit to all inspection regions.

3.2.2.1 Addition of vertices

To generate these connecting edges, the existing required edges are subdivided into a sequence of edges according to a user-selected *subdivision step size parameter*, d_{step}, while not changing the underlying graph geometry. These added *access vertices* offer opportunities to divide the inspection of the associated boundary segment among multiple robots, and some will also allow for more efficient transit to other parts of the graph. A vertex, v_{depot}, is added at the depot location at this point in the development of the graph, as shown in Figure 3.3(b).

3.2.2.2 Addition of connecting edges

A *complete graph* is then induced on the vertices of G by adding edges connecting each vertex in G to every other vertex in G if such an edge is not already in place. These new

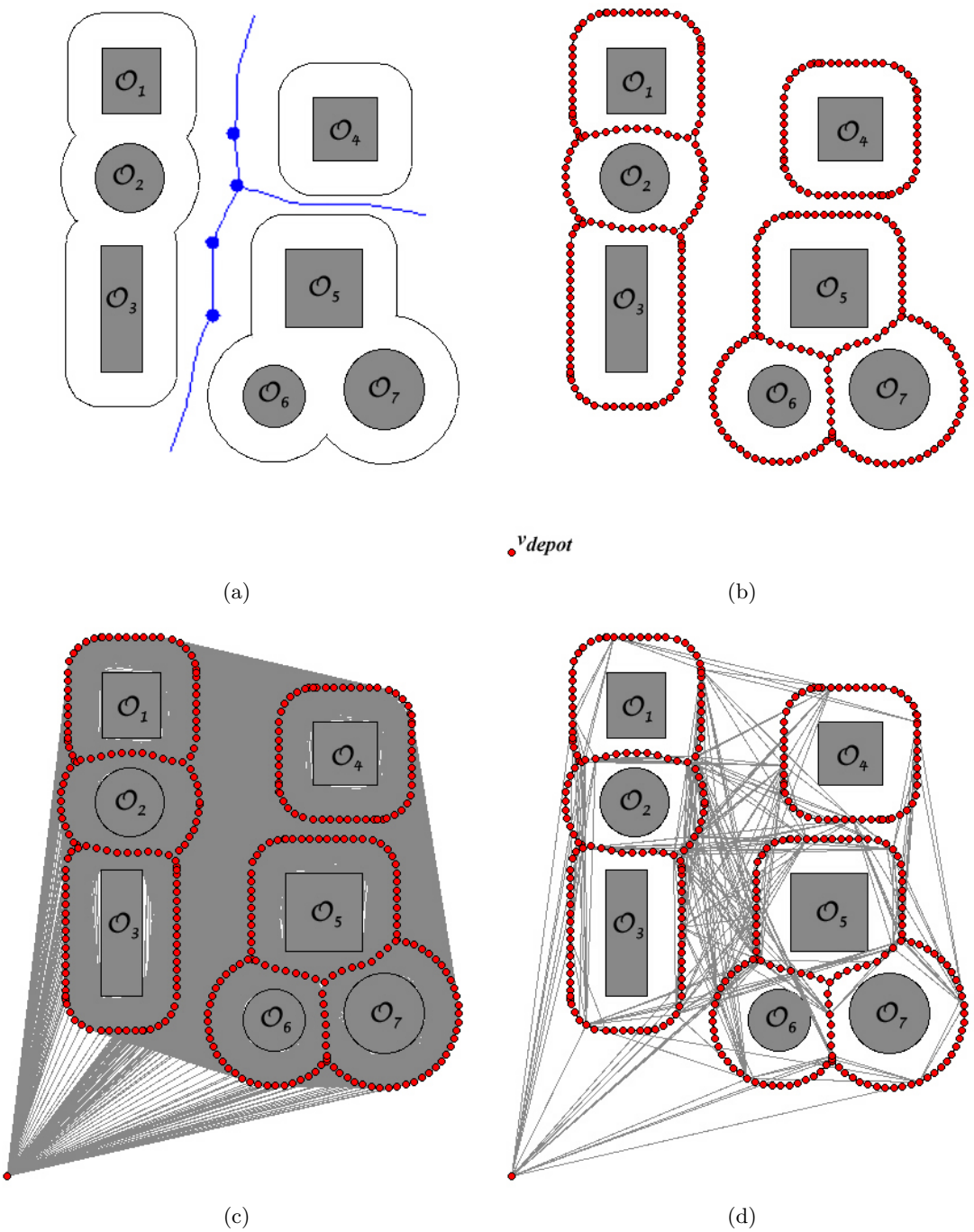


Figure 3.3: (a) Inspection regions are contained within inspection cells, defined by a subset of local components of the GVG; adjacent cells must contain at least one mutually visible vertex. (b) Placement of *access vertices* and v_{depot}, which represents the point of robot deployment. (c) Without the graph weeding procedure, the resulting final graph may have a prohibitively large number of edges. (d) A possible final graph representation for the sample environment from Figure 3.1(a).

edges are referred to as *induced edges*. Each induced edge represents a direct path through the environment from one vertex to another, and is assigned a weight equal to the distance between those vertices. If an induced edge's end vertices define a line segment that intersects a boundary, that edge is excluded from G, as it represents a path that does not lie entirely in freespace. All other induced edges are added to G and comprise the set of connectivity edges, E_C.

To ensure routes between different inspection regions are created, d_{step} must be small enough that if inspection regions $\mathcal{P}_i$ and $\mathcal{P}_j$ are in line of sight of one another, $\mathcal{P}_i$ will contain a vertex v_i that is in line of sight of a vertex v_j in $\mathcal{P}_j$. This guarantees that E_C will include at least one edge joining each disjoint component of G to every other graph component within line of sight, thus guaranteeing the final graph will be connected.

Each object in the environment is contained within a cell of the GVG. We refer to the union of the cells containing the objects within an inspection region as an *inspection cell*. Figure 3.3(a) illustrates the inspection cells for the sample environment. To ensure routes between different inspection regions are created, d_{step} must be small enough that if inspection regions $\mathcal{P}_i$ and $\mathcal{P}_j$ have adjacent inspection cells, $\mathcal{P}_i$ will contain a vertex v_i that is in line of sight of a vertex v_j in $\mathcal{P}_j$. This guarantees that E_C will include at least one edge joining each disjoint component of G to every other graph component within line of sight, thus guaranteeing the final graph will be connected.

3.2.2.3 Reducing graph size: the weeding parameter

As the number of edges in G becomes cumbersomely large, a possibility illustrated in Figure 3.3(c), the constructive heuristic will require a long time to run to completion. The user may elect to use an optional heuristic step to reduce the graph size while still achieving graph

connectivity. This step, which incorporates a user-selected *weeding parameter*, d_{weed}, results in the inclusion of only a subset of vertices $V_{induce} \subseteq V$ in the edge induction process. The first vertex added to the initially empty set of vertices, V_{induce} is the depot vertex, v_{depot}. For every other vertex $v \in V$, if the shortest paths from v to every vertex in V_{induce} is greater than the distance specified by d_{weed}, v is added to V_{induce}. Because the shortest path from a vertex within a disconnected graph component to any vertex outside the component is infinite, the construction of V_{induce} will result in the inclusion of at least one vertex from each disconnected component of G. To guarantee the graph will be connected by edges associated with paths lying in freespace, d_{weed} is subject to the same upper bound as d_{step}.

A possible final graph representation of the sample environment from Figures 3.1(a) is illustrated in Figure 3.3(d).

This is the graph construction method used in [71]. Because the construction of the graph is directly related to the geometry of the visibility regions for a given instance of the task, this method of graph construction is termed the *Visibility Region (VR) method*.

3.3 A Graph Algorithm to Solve the Boundary Coverage Problem

Based on the graph representation outlined in the previous section, the k-robot boundary coverage problem can be posed as a graph problem that I term the *k-Rural Postman Problem* (kRPP). This problem is concerned with finding a set of $k \geq 1$ tours $T = \{T_1, ..., T_k\}$ within an undirected, connected weighted graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$ such that each edge in a required subset of edges $E_R \subseteq E$ is traversed in at least one tour.

This nomenclature is adapted from a group of *Postman Problems* in the graph literature.

In 1962, Kwan posed a problem concerned with finding the minimum length tour of a graph guaranteeing every edge in the graph is traversed at least once [73]. Due to the routing applications suggested by Kwan and that paper's Chinese origin, it is widely referred to as the *Chinese Postman Problem* (CPP). Edmonds showed that the undirected CPP is efficiently solvable in 1965 [74] and Edmonds and Johnson presented an algorithm for solving the CPP on directed and undirected graphs in 1973 [75].

Many variations of the CPP have been explored, of which the two most closely related to the graph problem at hand are:

1. The *Rural Postman Problem (RPP)* [76], an $\mathcal{NP}$-complete problem in which the task is to find a minimum weight tour that traverses every edge in a required subset of edges in a connected, undirected graph G, and

2. The *Min-Max k-Chinese Postman Problem (MMkCPP)*, an $\mathcal{NP}$-hard problem in which the task is to find a set of k tours of a connected, undirected graph G such that the weight of the longest tour is minimized, each tour starts and finishes at the same "depot" vertex, and every edge in G is traversed in at least one tour [77, 78].

Observe that for $k = 1$, the kRPP reduces to the RPP. Because the kRPP can be reduced to a known $\mathcal{NP}$-complete problem [79], one can deduce that the kRPP is $\mathcal{NP}$-hard, and that a constructive heuristic approach to its solution is appropriate.

Further note that if $E_R = E$, the kRPP reduces to the kCPP; with the additional aim of minimizing the longest of the k tours, the MMkCPP is an instance of the kCPP. In [77], Ahr and Reinelt address this minimization both within the constructive heuristics they present and with post-construction tour improvement algorithms. While the formulation of the kRPP does not specifically require minimization of the longest tour, my constructive

heuristic does attempt to balance the inspection load spatially across the k tours by grouping neighboring edges into the same tours and seeks to avoid unnecessary redundant coverage.

3.3.1 A Modular Constructive Heuristic to Solve the kRPP

3.3.1.1 Definitions

Given an undirected, connected graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$ set of edges on the shortest path between vertices u and v, $\{u, v\} \in V$ is denoted $SP(u, v)$. The length of this path through G is denoted $C_G(SP(u, v))$.

The distance through graph G between a vertex v and an edge $e = \{x, y\}$ is defined as

$$d(v, e) = max\{C_G(SP(v, x)), C_G(SP(v, y)))\}. \tag{3.1}$$

The distance between two edges $e = \{x, y\}$ and $g = \{u, v\}$ through graph G is defined as

$$\begin{aligned} d(e, g) = \quad & max\{C_G(SP(u, x)), C_G(SP(u, y)), \\ & C_G(SP(v, x)), C_G(SP(v, y))\}. \end{aligned} \tag{3.2}$$

3.3.1.2 Partition the required edges E_R

The algorithm begins by partitioning the required edges E_R into k groups, one for each robot, $F_1 \cup ... \cup F_k = E_R$, using a farthest-point clustering method similar to that used in the "Cluster Algorithm" heuristic for the Min-Max k-Chinese Postman Problem presented by Ahr and Reinelt [77]. The method is based on an algorithm that aims to minimize the maximum intra-cluster distance [80].

Given a set of k *representative edges* $(f_1, ..., f_k)$, f_i is the first edge assigned to the cluster F_i that will eventually form the i^{th} robot's tour. The $|E_R| - k$ remaining edges $e \in E_R$ are assigned to the group F_j that minimizes $d(e, f_j)$. The k representative edges are chosen such that the first representative edge, f_1, is the edge $e \in E_R$ that maximizes $d(v_{depot}, e)$. Subsequent representative edges $f_i = e$ are chosen to maximize $\sum_{j=1}^{i-1} d(e, f_j)$.

After this partitioning step, which is carried out by a centralized, supervisory agent, the remainder of the planning procedure can take place online, with each robot calculating its own route.

3.3.1.3 Include edges for connectivity

The task remains to ensure each cluster of edges is connected and, thus, traversable. To that end, edges are added to each group to create k connected subgraphs of G, each of which must also include the depot vertex.

Given the subgraph $G_{R_i} = G[v_{depot} + F_i]$, which consists of v_{depot} and the edges in F_i, a graph G'_{R_i} is constructed with a vertex representing each connected component of G_{R_i}. An edge is added to G'_{R_i} between every vertex u and v. The list of vertices contained in the component of G_{R_i} represented by vertex u in G'_{R_i} is $V_u \subseteq V(G_{R_i}) \subseteq V(G)$. The weight $c(e)$ of edge $e = \{u, v\}$ is equal to $min_{q \in V_u, r \in V_v}\{C_G(SP(q, r))\}$, the length of the shortest path on G between the component represented by u and component represented by v. The vertices in G that minimize $c(e)$ are denoted $q_{u,v}$ and $r_{u,v}$. A minimum spanning tree is computed on G'_{R_i}; the edges in the shortest path in G, $SP(q_{u,v}, r_{u,v})$, associated with each edge in the spanning tree, $e = \{u, v\}$, are added to F_i, yielding a connected subgraph G_i.

3.3.1.4 Compute a tour of each subgraph G_i

A single-postman tour T_i is computed on each subgraph $G_i, i = \{1, ..., k\}$ using the Chinese Postman algorithm as presented by Gibbons in *Algorithmic Graph Theory*, page 164 [81]. Each tour $T_i \in \{T_1, ..., T_k\}$ originates and terminates at v_{depot}, and has length $C_G(T_i) = \sum_{e \in T_i} c(e)$.

3.3.1.5 Refine tours

Finally, for each tour T_i, sequences of edges that are retraced in the course of a tour are replaced with the shortest path in G, if a shorter path between the end vertices of the sequence exists. When a robot has traversed every required edge in its tour, it returns to the depot via the shortest path through G. Additional path refinement heuristics could certainly be implemented.

3.3.2 Path Planning Using the Graph Algorithm

The k robots' inspection paths are crafted from the graph solution to the kRPP. Because edges in G specify paths through freespace, the tours found by the graph algorithm each define a series of waypoints. Each robot visits its tour's waypoints in sequence. When a robot detects an unexpected object (i.e., another robot) blocking the way to its next waypoint, it carries out a brief obstacle avoidance maneuver, then continues to navigate to its next destination. Such encounters could easily be managed using other navigation methods, as well.

3.3.3 Lower Bounds on Tour Lengths

Shortest Path Tour Lower Bound The Shortest Path Tour Lower Bound described by Ahr and Reinelt [77] for the MMkCPP also applies to the kRPP with minor additional consideration. Because each required edge must be visited by a robot deployed from v_{depot}, the longest tour must be of length greater than or equal to the shortest path from v_{depot} to e_{max} and back, where $e_{max} = e \in E_R$ maximizes $d(v_{depot}, e)$. For $e_{max} = \{u, v\}$, this bound is $C_G(SP(v_{depot}, u)) + c(e_{max}) + C_G(SP(v, v_{depot}))$. Solutions will come closest this bound when k is well-suited to the size of the graph.

Fair Division Lower Bound Consider the length of the tour of E_R computed by the algorithm for $k = 1$, termed C_{RPP}, as this tour is a solution to the RPP posed on G. A lower bound on the length of the longest tour implying an equal division of labor among the k robots is given by C_{RPP}/k.

3.3.4 Completeness of Boundary Coverage

Lemma 3.3.1. *Traversal of every edge in E_R will result in the inspection of every point in $\partial\mathcal{O}$.*

Proof. Recall that every point in $\partial\mathcal{O}$ is also contained in $\mathcal{P}$, i.e, $\partial\mathcal{O}_{\mathcal{P}_1} \cup ... \cup \partial\mathcal{O}_{\mathcal{P}_m} = \partial\mathcal{O}$. By construction, every point in $\partial\mathcal{O}$ is contained either within an exterior or an interior boundary segment. If every edge constructed and added to E_R is traversed, then by Lemma 3.2.1, every exterior boundary segment in $\mathcal{P}$ will be inspected and by Lemma 3.2.2, every interior boundary segment will be inspected. It follows that traversal of every edge in E_R will result in the inspection of every point in $\partial\mathcal{O}$. □

Lemma 3.3.2. *The set of k tours generated by the constructive heuristic guarantees each edge in the required subset of edges E_R is traversed in at least one tour.*

Proof. By construction, every edge $e \in E_R$ is assigned to a cluster F_i. Additional edges are added to the clusters to ensure each is a connected subgraph of G. A tour of each cluster is then calculated with the Edmonds and Johnson CPP algorithm, an algorithm that guarantees every edge in the cluster will be traversed [75]. As every edge in E_R is contained in a cluster, and every edge in every cluster will be traversed by a tour, it follows that every edge in E_R will be traversed in at least one tour. □

Proposition 3.3.3. *Given a connected, undirected graph G constructed with the VR method, the paths planned from the kRPP graph solution provide complete coverage of $\partial\mathcal{O}$.*

Proof. In Section 3.5.1, each inspection region is considered in turn. By Lemma 3.3.1, if every edge in E_R is traversed (i.e, a robot travels along a visibility path represented by that edge), every point on the boundary $\partial\mathcal{O}$ will be inspected. By Lemma 3.3.2, the kRPP heuristic gives a graph solution that guarantees the traversal of every edge in E_R, thus complete boundary coverage is guaranteed. □

3.4 Results and Discussion: Inspection Planning

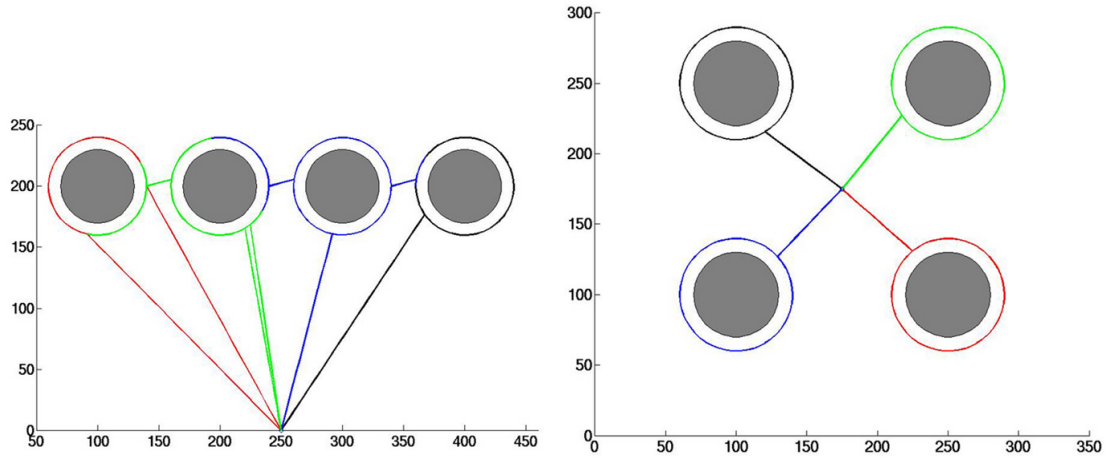
This section provides three examples to illustrate some of the key characteristics of this algorithm and some of the main issues related to its practical use. The distance units used in simulations are arbitrary, and are intended to show relative distance.

3.4.1 Division of Labor by Edge Partitioning

This first example illustrates the division of labor imposed by the kRPP approach to routing the robots. In this example, four robots are tasked to inspect an environment containing four identical circular boundaries, and each robot's preferred sensing distance r is small enough that no multiview regions arise. Applying the algorithm to the configuration of the disks shown in Figure 3.4(a), results in a counterintuitive division of the task where one robot is not assigned to individually inspect each disk boundary. This somewhat unusual division arises from the nature of the farthest-point edge-partitioning method used to divide the edges among the k tours. While there are four distinct boundaries, the partitioning algorithm may group the edges in seeming unnatural ways. When the boundaries are arranged in an equidistant pattern from the depot and are sufficiently separated so that their inspection edges cluster in a natural way, then an "intuitive" division of labor arises, as illustrated in 3.4(b).

While the routes are of equal length in this case, the scenario in Figure 3.4(a), in comparison, yielded a longest route 38% longer than the shortest route and 24% longer than the average length of the four routes. Though the paths are not intuitive in that scenario, the load is still roughly balanced among the robots. Because the partitioning algorithm seeks to minimize the intra-cluster distance of each cluster, it does not "fairly" distribute the weight of the edges in each group; how evenly the inspection load is balanced across the robots will be dependent on the weight and concentration of required edges in proximity to the k representative edges. For an inspection with a relatively homogeneous distribution of required edges, the routing will, however, certainly be "fair" enough to offer a tour length advantage with $k > 1$ robots.

Further tour weight optimization with alternative partitioning methods and post-construction



(a) (b)

Figure 3.4: The environments in (a) and (b) contain four identical circular boundaries, but placement has a large effect on the "fairness" of the planned routes. The routes planned for $k = 4$ robots are illustrated with waypoints marked by route-specific colors of black, green, red, and blue.

tour improvement heuristics is certainly possible.

3.4.2 Multiview Regions and More Complex Environments

Consider the following two illustrative examples of routing through more complex environments. The first "scattered" environment contains several objects of various sizes and shapes, while the second "packed" environment contains a closely spaced set of identical objects. For each environment, consider two values of r, one small enough that no multiview regions arise, the other large enough that every boundary has an associated multiview region. The scattered environment is shown in Figure 3.5(b), while the packed environment is illustrated in Figure 3.6(b).

The major difference between the two environment examples is the fraction of the required edges that lies inside multiview regions. As r increases, the total length of each curve in S_r increases. Due to the assumption of a preferred visibility path, the weight

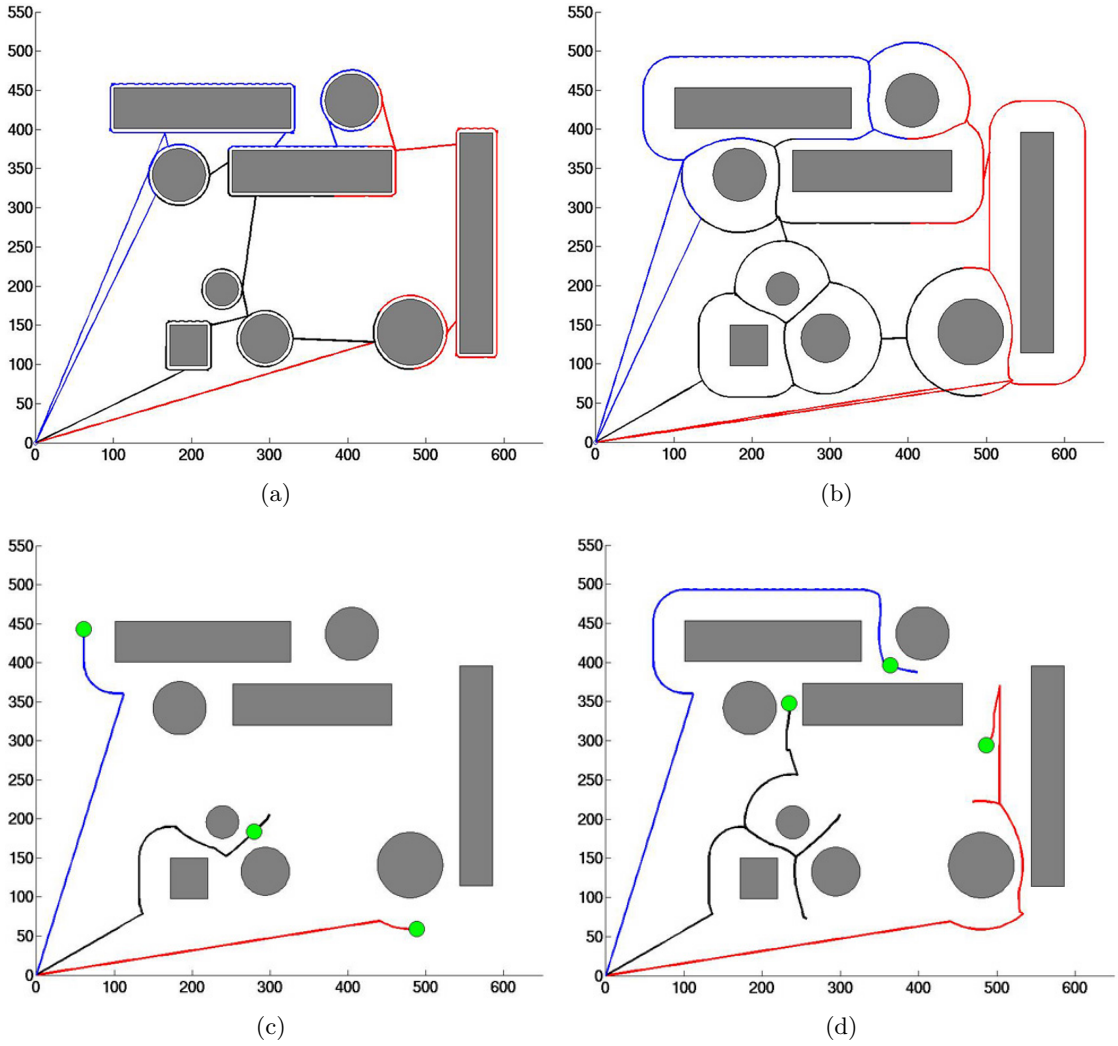


Figure 3.5: In an environment with several objects of various sizes and shapes, routes planned for $k = 3$ robots with (a) $r = 5$, small enough that no multiview regions arise, and [(b), (c), and (d)] $r = 40$, large enough that every boundary is associated with a multiview region are illustrated. Routes are marked by a route-specific color of blue, black, or red. Two "snapshots" of the $r = 40$ inspection in progress are shown at time steps (c) 50 and (d) 100. A total of 605 time steps is required for the inspection to be completed and for all robots to return to the depot. The green dots in (c) and (d) represent the robots' positions at that given time step.

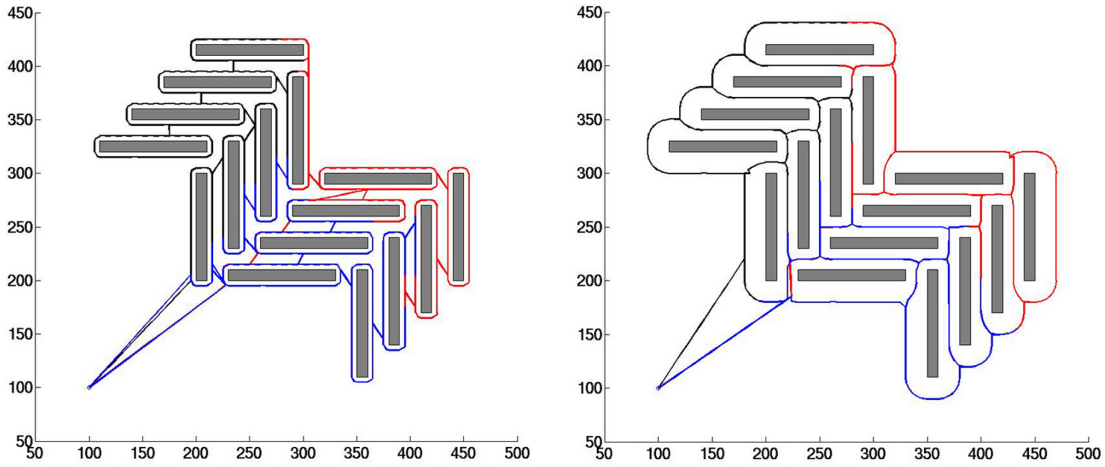
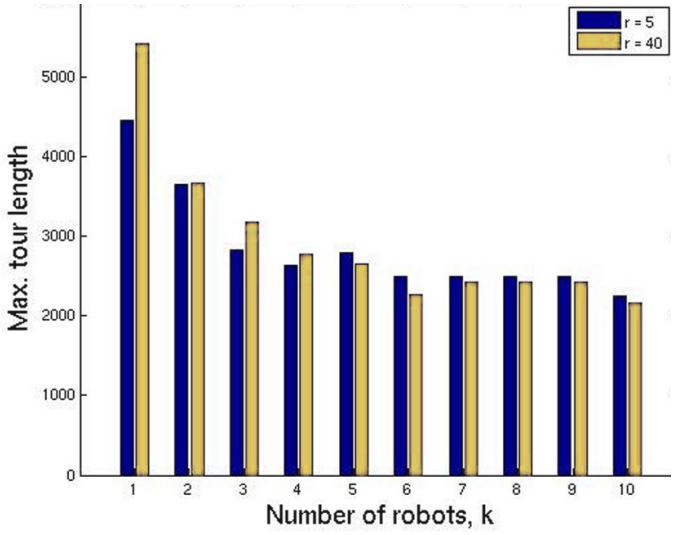


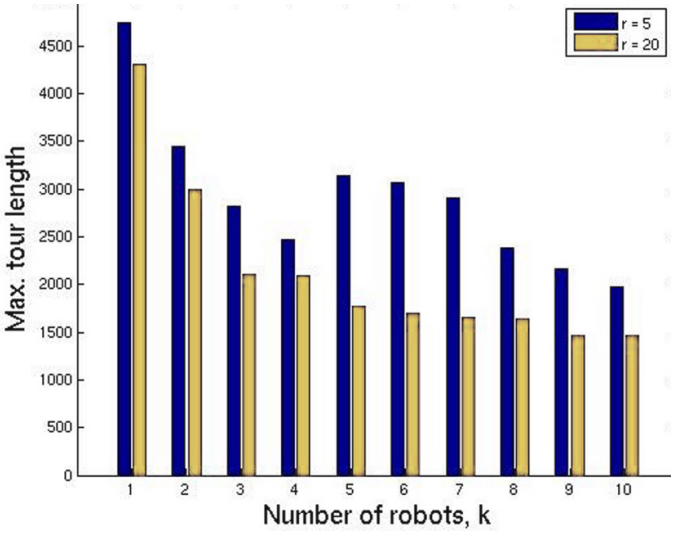
Figure 3.6: In an environment with identical, closely packed objects, routes planned for $k = 3$ robots with (a) $r = 5$, small enough that no multiview regions arise, and (b) $r = 20$, large enough that most edges in E_R are part of a multiview region are illustrated. Routes are marked by a route-specific color of black, blue, or red.

of external required edges is directly related to the path length along curves in S. The weight of required edges within multiview regions will also increase as r increases, but not as significantly; once the inspection region is saturated by the increase in r (the multiview regions contain a single connected component of the local GVG), the internal edge weights will only increase marginally where the GVG components meet external edges.

In the scattered environment, the increase in length of the preferred paths as r increases outweighs the benefit in path reduction of traversing a multiview region. The tours in the large-r scattered environment example tend to be longer than in the small-r case because the total weight of edges in E_R was greater than in the small-r case, inflation resulting from the relatively small fraction of edges in E_R associated with multiview regions. A large part of this disadvantage due to inflation is artificially imposed by the constraint of the preferred path's distance r_{max} from the boundary. The relaxation of this constraint is addressed in Section 3.5.



(a)



(b)

Figure 3.7: (a) Comparison of the longest tour lengths vs. number of robots k for $r = 5$ and $r = 40$ in the "scattered" environment in Figure 3.5. (b) Comparison of the longest tour lengths vs. number of robots k for $r = 5$ and $r = 20$ in the "packed" environment in Figure 3.6. For $r = 20$, the tour lengths are significantly shorter than for the $r = 5$ case, illustrating the advantage of exploiting multiview regions in a packed environment.

The packed environment is reminiscent of the turbine-blade environment (Figure 1.1) that motivated this inspection task. In this example, the path-length advantage of exploiting multiview regions in the routing task is evident. For large enough r in a crowded enough environment, a large portion of the edges in E_R will be within multiview regions, allowing the routing algorithm to avoid multiple passes through the same region. The weight of edges within multiview regions will not change as significantly as external edges' weights will as r increases. The packed environment in Figure 3.6 illustrates the advantage of exploiting multiview regions in a packed environment, as the large-r tour lengths are significantly shorter than the small-r tours, as shown in Figure 3.7(b).

In both examples, despite the difference in graph structure between the two r-value cases, the resulting routes are remarkably similar in their division of the task. This is evident in Figs. 3.5 and 3.6; each route in the small-r scenario has a corresponding route in the large-r scenario that inspects approximately the same set of boundary segments.

The effects of team size k on tour lengths are shown in Figs. 3.7(a) and 3.7(b). Despite the pronounced difference in graph structure introduced by the different r values, the performance is qualitatively similar for both example environments. With the addition of a few robots, the longest tour length drops dramatically, but the tour length benefit decreases exponentially with k. As k increases, the average tour length levels off. This is due to the partitioning method's requirement that every tour contain at least one required edge. As a result, as k increases, the heuristic begins developing nearly identical tours for multiple robots.

For a cost function based on time to completion, a choice in which the path length of the longest tour will dominate, dividing the task among two or more robots offers a marked benefit. If, in addition to time concerns, the inspection process itself is more

time consuming and costly than traveling between inspection regions, dividing the task and avoiding redundant coverage is particularly beneficial. However, if the cost function is primarily based on total robot energy output, e.g., distance traveled, using a single robot will likely offer a lower cost strategy.

3.5 A Graph Representation for Boundary Coverage: The Boundary Following Method

The graph construction procedure introduced in this section is motivated by the VR method described in Section 3.2, but key differences provide significant improvements in overall graph complexity and inspection path length.

While the multiview regions' potential for reducing the path length of the robots' inspection tours should be exploited, the length of the visibility paths for those boundary segments not visible from a multiview region could be minimized by following the boundary as closely as possible. This can be accomplished by removing the constraint that robots carry out inspection at a preferred sensing distance. Instead, the planning process considers the entire inspection sensing range, from distance r_{min} to r_{max} away from the boundary. All boundary points in the environment are assumed to accessible to the robot, thus disjoint object boundaries must be spaced at least $2r_{min}$ apart. Previously, a point p on the boundary was considered inspected when a robot passed within distance r_{max} orthogonal to the boundary. To allow for inspection of concave corners (described in Section 3.5.1.2), this constraint is relaxed. Though the orthogonal sensing orientation is still preferred, a point p is considered inspected when a robot passes within distance r_{max}.

In contrast to the VR method, the approach in this section allows for the inspection

of objects with nonconvex boundaries. Each object boundary is still assumed to be a piecewise smooth, closed curve. Nonconvex objects are partitioned into convex objects for the purposes of calculating the local GVG and constructing the graph representation of the inspection task. This partitioning creates a new type of boundary segment, the *inaccessible boundary segment*, which cannot be and does not need to be inspected. To reflect this, $\partial\mathcal{O}$, originally defined as the collective boundary of all objects in the environment in Section 3.1, is redefined as the boundary of the union of all objects in the environment. By this definition, $\partial\mathcal{O}$ consists of all exterior boundary segments, $\partial\mathcal{O}_{ext}$, and interior boundary segments, $\partial\mathcal{O}_{int}$, and excludes all inaccessible boundary segments.

As in Section 3.2, the inspection graph G is comprised of required inspection edges, E_R, and connectivity edges, E_C. Again, a single, easily defined "preferred visibility path" will be constructed for each edge and its length will be assigned as that edge's weight. Because this method is motivated by following the boundary as closely as possible while still making efficient transitions to the inspection of neighboring boundaries, this graph construction method is called the *Boundary Following (BF) method*.

Note that the graph searching and decomposition algorithm presented in Section 3.3 is independent of the method used for the construction of G; it will accept any connected, undirected graph G as an input.

3.5.1 Determining E_R: Edges Required for Inspection

Let r_{min} denote the closest distance to an object that yields adequate inspection performance ($r_{min} < r_{max}$, and r_{min} may approach 0). Let $S_{max,j}$ and $S_{min,j}$ denote the set of offset curves that are displaced a distance r_{max} and r_{min} normal to the jth object boundary, $\partial\mathcal{O}_j$, respectively. The *configuration space for inspection*, $\mathcal{C}$, the union of freespace bounded

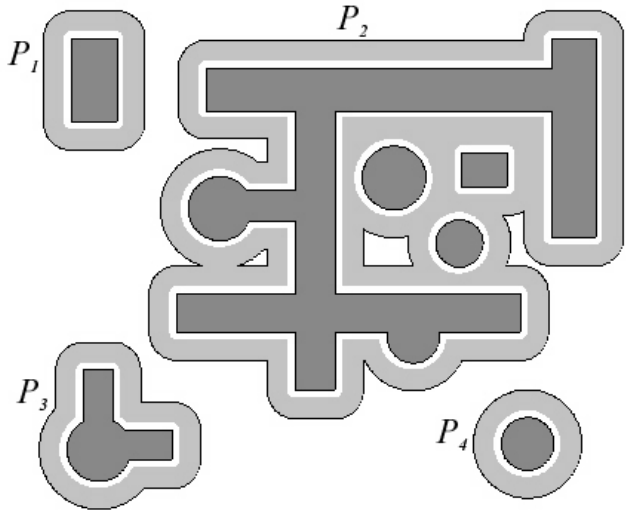
by $S_{min,j}$ and $S_{max,j}$ for all $j \in \{1, ..., N\}$ is the set of all robot poses from which a boundary point may be inspected. Let $\mathcal{P}^{BF}$ denote the union of the regions in freespace that are bounded by $S_{max} = \bigcup_{j=1,...,N} S_{max,j}$ and let $\mathcal{P}_i^{BF}, i \in \{1, \ldots, m\}$, denote the constituent disjoint *inspection regions*. Figure 3.8(a) shows a sample environment incorporating non-convex objects and a value of $r_{max} = 4r_{min}$. The boundaries of the inspection regions are labeled and $\mathcal{C}$ is shown with shaded regions.

The objective in this method of graph construction is to find an efficient way to transition from paths that closely follow the boundary to paths that exploit multiview regions while guaranteeing that the path lies within $\mathcal{C}$ and passes within r_{max} of every point in $\partial\mathcal{O}$.

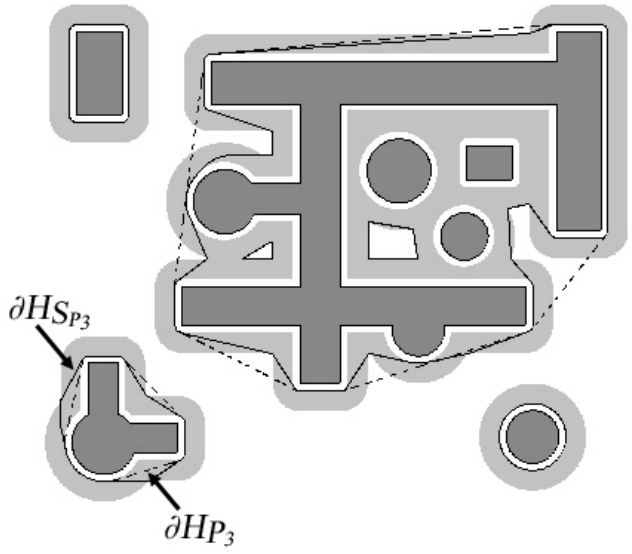
Each disjoint inspection region $\mathcal{P}_i^{BF}$ is examined in turn. Consider every object $\mathcal{O}_j$ whose visibility region is contained in $\mathcal{P}_i^{BF}$ and denote the list of those objects' indices with l_i. The required edges in G that closely follow the exterior boundaries, termed *boundary following edges,* are constructed first, then required edges that lie within multiview regions, *multiview edges*, are added. Figure 3.9(a) illustrates the required edges as constructed with the BF method.

3.5.1.1 Boundary following edges

In order to ensure that the exterior boundary segments for each object $\mathcal{O}_j, j \in l_i$ are completely inspected, each object is considered individually. To construct the preferred visibility path for the exterior boundary segments of $\mathcal{O}_j$, first consider the outer extremes of the multivew regions adjacent to the object. These extreme points, $\mathcal{O}_j$'s *multiview extrema*, occur where where $S_{max,j}$ and $S_{max,k}$, $j \neq k$, intersect. By definition, these points lie on the GVG, presenting a good opportunity to transition from a path tracing an exterior boundary segment to a path exploiting a multiview region.



(a)



(b)

Figure 3.8: (a) Illustration of the inspection regions, $\mathcal{P}_1^{BF}$ through $\mathcal{P}_4^{BF}$. The configuration space for inspection, $\mathcal{C}$, is shown with shaded regions. (b) For each inspection region $\mathcal{P}_i^{BF}$, $\partial\mathcal{H}_{S_{\mathcal{P}_i^{BF}}}$ is shown with a solid line. $\partial\mathcal{H}_{\mathcal{P}_i^{BF}}$ is shown with a dashed line.

For each $j \in l_i$, the region $\mathcal{H}_{S_j}$ bounded by the convex hull of $S_{min,j}$ and $\mathcal{O}_j$'s multiview extrema is calculated. The intervals of $\partial\mathcal{H}_{S_j}$ lying outside multiview regions define visibility paths for $\mathcal{O}_j$'s exterior boundary segments. Let $\mathcal{H}_{S_{\mathcal{P}_i^{BF}}} = \bigcup_{j \in l_i} \mathcal{H}_{S_j}$. The construction of $\mathcal{H}_{S_{\mathcal{P}_i^{BF}}}$ preserves the intervals of each $\partial\mathcal{H}_{S_j}$ that define the exterior boundary segments' visibility paths. The construction also discards the intervals that fall within multiview regions (corresponding to interior boundary segments, which will be considered below when multiview regions are examined) or intersect other objects' boundaries (corresponding to inaccessible boundary segments, which need not be inspected). Solid lines denote $\partial\mathcal{H}_{S_{\mathcal{P}_i^{BF}}}$ in Figure 3.8(b).

If r_{max} is large relative to the size of the objects to be inspected, for objects at the edge of $\mathcal{P}_i^{BF}$, $\partial\mathcal{H}_{S_{\mathcal{P}_i^{BF}}}$ may describe paths from the boundary following curve to the edge of the multiview region that take the robot farther away from the objects in the inspection region than if the robot simply proceeded along a tangent path to an adjacent object. To avoid this situation, those tangent paths that are part of the convex hull, $\partial\mathcal{H}_{\mathcal{P}_i^{BF}}$, of $\bigcup_{j \in l_i} S_{min,j}$ are found. This convex hull is illustrated with a dashed line in Figure 3.8(b) and the region bounded by this convex hull is denoted $\mathcal{H}_{\mathcal{P}_i^{BF}}$. In that example, two segments of $\partial\mathcal{H}_{\mathcal{P}_3^{BF}}$ provide a more direct path through $\mathcal{C}$ than $\partial\mathcal{H}_{S_{\mathcal{P}_3^{BF}}}$ does.

To take advantage of these "shortcuts," consider $\mathcal{H}_i = \mathcal{H}_{\mathcal{P}_i^{BF}} \bigcap \mathcal{H}_{S_{\mathcal{P}_i^{BF}}}$. The boundary $\partial\mathcal{H}_i$ describes the preferred visibility paths for the inspection of all *hull boundary segments* within $\mathcal{P}_i^{BF}$, the subset of boundary segments that will *not* be inspected from within a multiview region. As such, the set of hull boundary segments includes all exterior boundary segments in $\mathcal{P}_i^{BF}$ and, where $\mathcal{H}_{\mathcal{P}_i^{BF}}$ offers a more direct path, the outermost portions of some interior boundary segments. Graph vertices are placed where the GVG intersects $\partial\mathcal{H}_i$, then edges are added representing the interval of $\partial\mathcal{H}_i$ between those vertices. This method

of vertex addition ensures the continuity of inspection between interior hull boundary segments, which will be inspected in the course of traversing $\partial\mathcal{H}_i$, and the remaining interior boundary segments, which will be inspected from within a multiview region.

If $\mathcal{P}_i^{BF}$ incorporates no multiview regions, $\mathcal{P}_i^{BF}$ contains a single object, $\mathcal{O}_j$. The preferred visibility path for $\partial\mathcal{O}_j$ is $\partial\mathcal{H}_i = S_{min,j}$. In this case, a vertex is placed on the preferred visibility path at an arbitrarily chosen point. A single self-looping required graph edge of weight equal to the length of $S_{min,j}$ is added to G.

Let $\mathcal{H} = \bigcup_{i=\{1,\ldots,m\}} \mathcal{H}_i$. Because the regions $\mathcal{H}_i$ are disjoint, $\partial\mathcal{H} = \bigcup_{i=\{1,\ldots,m\}} \partial\mathcal{H}_i$. Edges in E_R representing intervals of $\mathcal{H}$ are termed *boundary following edges* and their traversal will result in the inspection of all hull boundary segments.

3.5.1.2 Multiview edges

Edges corresponding to the preferred visibility paths for the remaining boundary segments in $\partial\mathcal{O}$ are established by adding to G all edges and vertices of the GVG contained within $\mathcal{H}$. These edges are assigned a weight equal to the length of the corresponding GVG path. GVG edges only partially contained in $\mathcal{H}$ are truncated at the vertex that was placed at the point of intersection during the construction of the boundary following edges.

Recall that GVG edges describe the locations that are equidistant from the nearest object. Because nonconvex objects have been partitioned into abutting convex objects, the GVG will incorporate edges that allow for the simultaneous inspection of interior boundary segments belonging to the same nonconvex object prior to partitioning, ensuring that all boundary points in such "interior regions" of a nonconvex object are inspected. This partitioning also leads to GVG edges that approach concave corners, terminating in a single degree vertex where the objects meet. Because inspection of points in these "corner regions"

from a distance of r_{max} is sufficient, corner GVG edges are truncated a distance r_{max} from the point of object intersection. Several examples of such corner regions can be seen in Figure 3.9(a).

Edges in E_R representing intervals of GVG $\bigcap \mathcal{H}$ (which defines the preferred visibility paths for all remaining boundary segments in $\partial\mathcal{O}$) are termed *multiview edges*.

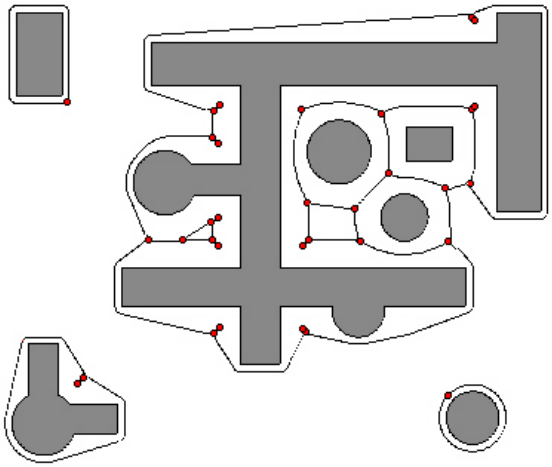
The collection of preferred visibility paths represented by edges in E_R is denoted $R = \partial\mathcal{H} \bigcup (\text{GVG} \bigcap \mathcal{H})$.

Lemma 3.5.1. *Traversal of every edge in E_R will result in the inspection of every point in $\partial\mathcal{O}$.*

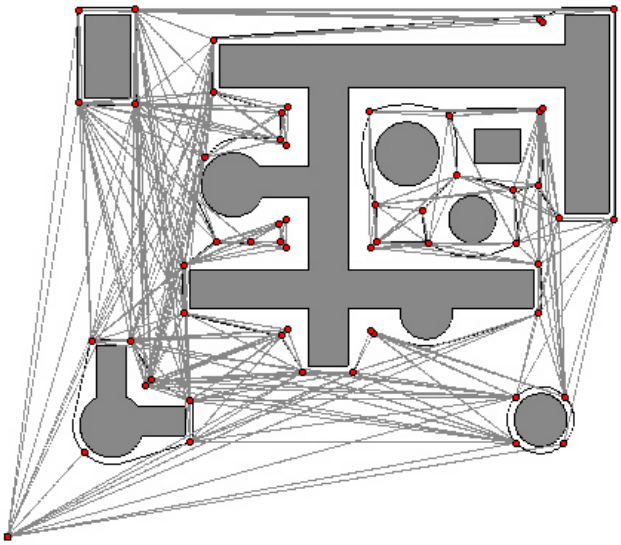
Proof. By construction, the traversal of the visibility paths $\partial\mathcal{H}$ associated with boundary following edges will result in the inspection of every hull boundary segment in $\partial\mathcal{O}$. Also by construction, the traversal of the visibility paths GVG $\bigcap \mathcal{H}$ associated with multiview edges will result in the inspection of all remaining boundary segments in $\partial\mathcal{O}$. Every preferred visibility path in $R = \partial\mathcal{H} \bigcup (\text{GVG} \bigcap \mathcal{H})$ is represented by an edge in E_R. Therefore, the traversal of every edge in E_R will result in the traversal of every interval of R, which will result in the inspection of every point in $\partial\mathcal{O}$. □

3.5.2 Determining E_C: Edges Providing Connectivity

To connect disjoint inspection regions and allow for more efficient travel within inspection regions, *access vertices* are added to G. As in the VR method, mutually visible vertices must exist between adjacent inspection regions to guarantee G's connectivity. To meet this requirement (as well as to add points of access on edges that are long relative to those constructed with the VF method), for each object $\mathcal{O}_j$ in the inspection region, the points



(a)



(b)

Figure 3.9: (a) Construction of edges in E_R. (b) A possible final graph representation for the sample environment from Figures 3.8(a) and 3.8(b).

tangent to $S_{min,j}$ from distant points in the four cardinal directions are found. For any of these points lying on a preferred visibility path, a vertex will be added to G at that point, subdividing the edge representing that visibility path.

As with the VR method, the group of required edges associated with boundary inspection in each inspection region $\mathcal{P}_i^{BF}$ forms a distinct connected subgraph of G. The possibly disjoint components of this construction must be connected to allow for robot transit to other parts of the graph.

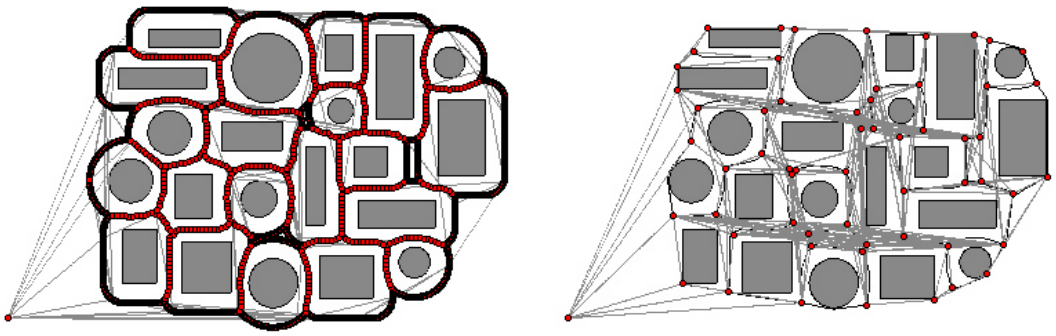
The vertex v_{depot} is added at the depot location. A *complete graph* is then induced on the vertices of G, adding edges connecting each vertex in G to every other vertex in G if an edge describing a direct path between those vertices does not already exist. These *induced edges* are assigned a weight equal to the distance between those vertices. If the path associated with an induced edge passes through an object, that edge is excluded from G. The induced edges comprise the set of edges connecting edges, E_C. The graph G is now connected, i.e, a path exists between every vertex in the graph.

A possible final graph representation of the sample environment is illustrated in Figure 3.9.

3.6 Results and Discussion: Impact of the Graph Representation

3.6.1 Reduction in Graph Complexity

To generate a typical graph representation with the VR method, the user selects a subdivision step size parameter, d_{step}, that directly affects the number of vertices $|V|$ in G and an edge-weeding parameter, d_{weed}, directly affecting the number of edges $|E|$ in G.



(a) (b)

Figure 3.10: (a) VR graph construction method (blue in Figure 3.11), shown for $r_{max} = 15$, $d_{step} = 1$, and $d_{weed} = 75$. G has 2595 edges and 2455 vertices. (b) BF graph construction method (yellow in Figure 3.11), shown for $r_{max} = 15$. G has 620 edges and 59 vertices.

In that construction technique, edges represent straight paths through freespace and the waypoints in the tours all correspond to vertices. In the BF method, waypoints along the paths represented by graph edges are simply a parameter associated with each edge. These waypoint lists contribute only to the storage space required for the graph and do not add to its complexity. The user may select the granularity of path information to suit system memory constraints; for examples presented here, a granularity equivalent to $d_{step} = 1$ (in arbitrary distance units) in the VR method was chosen.

A graph generated using the BF method will have far fewer vertices than a graph generated with the VR method for typical values of the user-selected parameters. The example graphs in Figure 3.10 were generated for typical fine granularity (i.e., small step size) values of these user-selected parameters. The BF method offers reduction in $|V|$ by more than an order of magnitude while offering an arbitrarily fine spatial resolution of waypoints. Such savings in $|V|$ were consistently seen in the examples explored.

3.6.2 Reduction in Computational Complexity

This reduction of graph complexity greatly improves the running time of the constructive heuristic. The most expensive steps in the algorithm are heavily dependent on the number of vertices in the graph, especially those incident to the edges in E_R:

1. The all pairs shortest path calculation used in the partitioning step (Section 3.3.1.2) has time complexity $O(|V_R|^3)$ [77], where V_R is the set of vertices incident to E_R.

2. The minimum spanning tree calculation used in the connectivity step (Section 3.3.1.3) has time complexity $O(|E_{F_i}|log|V_{F_i}|)$ [82] for each group F_i.

3. The single postman tour calculation (Section 3.3.1.4) has time complexity $O(|V_{G_i}|^3)$ [75] for each subgraph G_i.

Using the graph generated with the VR method for the example environment shown in Figure 3.10 for, and running on a Pentium-4 processor (3GHz CPU), the constructive heuristic runs to completion in 16 minutes 21.0 seconds for $r_{max} = 15$ and $k = 6$. The constructive heuristic runs to completion nearly 196 times faster using the graph generated with the BF method for the example environment, requiring only 5.007 seconds on the same machine.

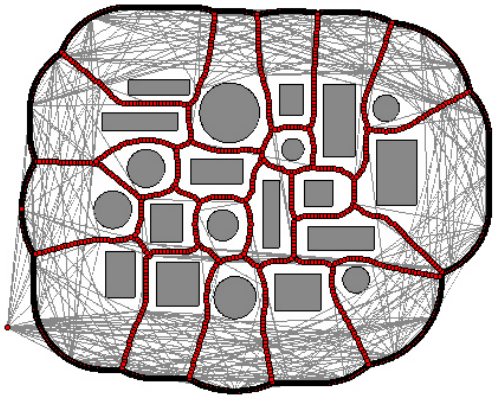
To further examine the explosion in running time for relatively small changes in user parameters, graphs were constructed with the VR method for $d_{step} = 1$ and $d_{step} = 2$; , keeping all other parameters fixed ($r_{max} = 15$, $d_{weed} = 75$, and $k = 6$). The resulting graphs had $|V| = 2455$ and $|V| = 1511$, respectively. For $d_{step} = 2$, the constructive heuristic ran to completion in 4 minutes 28.7 seconds, slightly more than a quarter of the time required for $d_{step} = 1$, showing the major impact of $|V|$ on the algorithm's running time.

For the values of k used in simulations ($k \leq 10$), team size did not play a large role in the running time for either construction method.

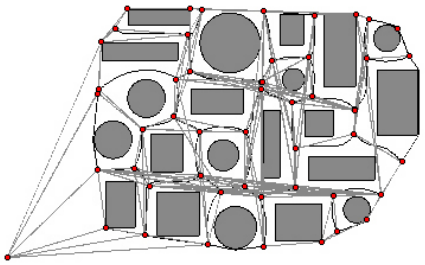
The vast reduction in G's complexity offered by the BF method allows the heuristic to solve the kRPP quickly enough that robots might use it online to revise their paths in light of changes to the environment or team. This brings about the opportunity to reuse this graph framework in a variety of related inspection tasks that will be explored in the next chapter.

3.6.3 Reduction in Total Length of E_R

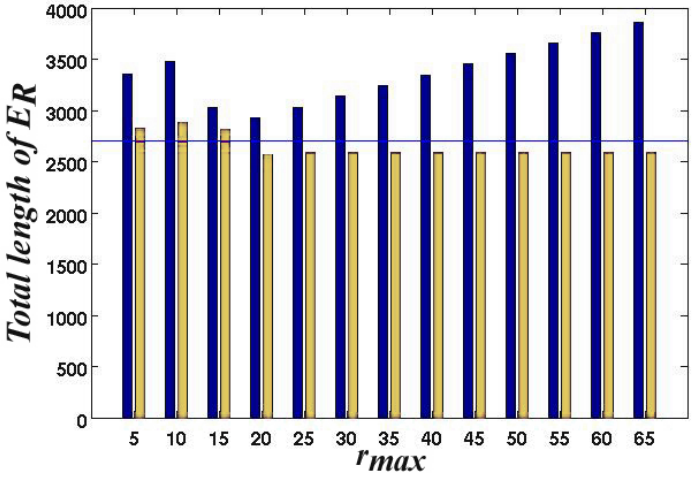
Using the BF method, as the constraint of the preferred sensing distance r is relaxed and more efficient boundary following paths are exploited, the total length of the preferred visibility paths represented by E_R is reduced. The extreme case of $r_{max} = 65$ for the example environment introduced in Figure 3.10 is shown in Figure 3.11(a) and (b). Figure 3.11(a) clearly illustrates the expanding exterior edges are the source of the monotonic increase in the total length of E_R in the VR method as r_{max} increases. Figure 3.11(b) illustrates how using $\mathcal{H}_i$ to describe efficient exterior edges curbs this growth in total E_R length. Figure 3.11(c) shows the total length of E_R for the VR example (in blue) and the BF example (in yellow) as a function of r_{max}. The total length of E_R is consistently smaller in graphs generated with the BF method. In contrast to the monotonic increase in total E_R length seen for graphs constructed with the VR method, the BF method constructs the same graph for any r_{max} greater than the r_{max} for which multiview regions saturate the interior of every inspection region's $\mathcal{H}_i$.



(a)



(b)



(c)

Figure 3.11: (a) VR graph construction method shown for $r_{max} = 65$. (b) BF graph construction method shown for $r_{max} = 65$. (c) Graph comparing the total length of E_R for VR (blue) and BF (yellow) methods shown in Figure 3.10 as a function of r_{max}. The horizontal line indicates the total length of $\partial\mathcal{O}$, the boundary to be inspected.

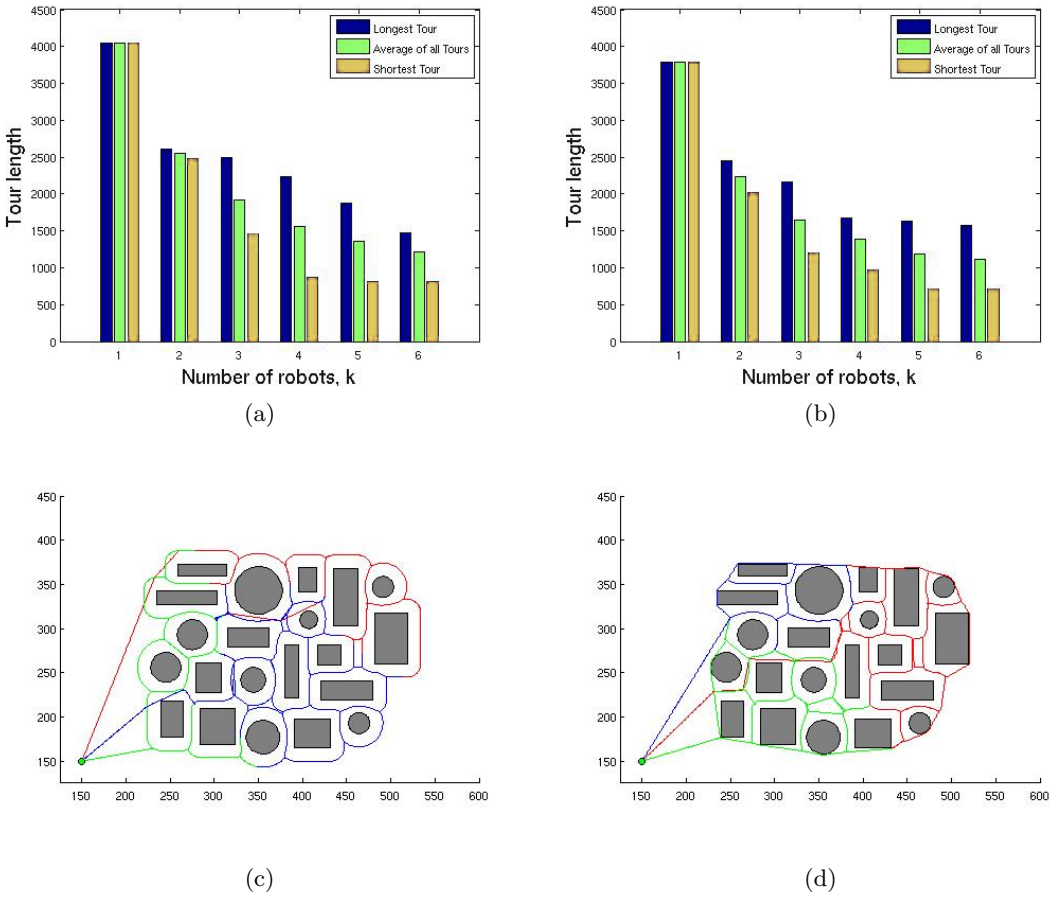


Figure 3.12: (a) Minimum, mean, and maximum tour lengths for the VR graph example shown in Figure 3.10(a) for various team sizes. (b) Minimum, mean, and maximum tour lengths for the BF graph example shown in Figure 3.10(b) for various team sizes. (c) Tours on VR graph calculated for $k = 3$. (d) Tours on BF graph calculated for $k = 3$.

3.6.4 Tour Length

When the kRPP heuristic is applied to graphs generated with both methods, both types of graphs yield comparable solutions. In general, as long as the graph edges are adequately subdivided, the graphs generated with the BF method yield somewhat shorter tours than VR-generated graphs with $r = r_{max}$ due to the more efficient preferred visibility paths. This subdivision (achieved in most cases with the addition of access vertices) allows the task to be divided among the robots; given too few edges to assign, the task may not be partitioned well. For the example environment shown in Figure 3.11, the methods come closest in total E_R length for $r_{max} = 15$. For that instance of the example, the tours planned for $k = 3$ robots are illustrated in Figure 3.12. While the spatial division of labor is not similar, the quality of the solution, as measured by the maximum, mean, and minimum tour lengths is comparable.

3.6.5 Division of Labor by Edge Partitioning

Another effect of the reduction of the degree of subdivision is notable in certain cases of labor division. Because the edges constructed with the BF method are not subdivided to the extent done in the VR method, the resulting division of labor is often more in line with intuition. Recall the row of circular boundaries in Figure 3.4(a). In contrast to the nonintuitive division of inspection in that example, when the graph for the same task is constructed using the BF method, the result is a much more intuitive division, shown in Figure 3.13 with one robot inspecting each circular object.

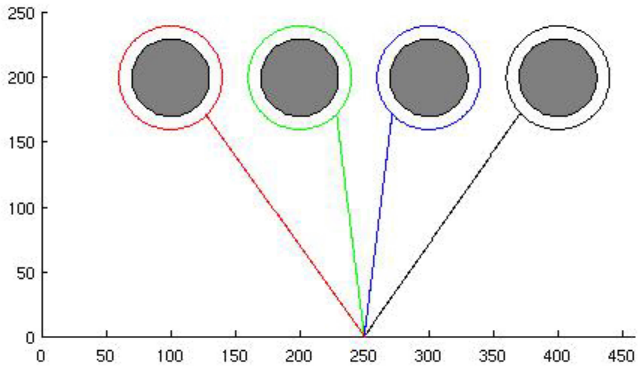


Figure 3.13: In contrast to the nonintuitive division of inspection achieved using a graph constructed with the VR method in Figure 3.4(a), when the graph for the same task is constructed using the BF method, the result is a much more intuitive division, with one robot inspecting each circular object.

3.6.6 Completeness of Boundary Coverage

The BF method, like the VR method, produces a graph representation that includes a visibility path for the inspection of every point in $\partial\mathcal{O}$.

Proposition 3.6.1. *Given a connected, undirected graph G constructed with the BF method, the paths planned from the kRPP graph solution provide complete coverage of $\partial\mathcal{O}$.*

Proof. By Lemma 3.5.1, if every edge in E_R is traversed (i.e, a robot travels along a visibility path represented by that edge), every point on the boundary $\partial\mathcal{O}$ will be inspected. By Lemma 3.3.2, the kRPP heuristic gives a graph solution that guarantees the traversal of every edge in E_R, thus complete boundary coverage is guaranteed. □

Chapter 4

Multi-robot Inspection with Plan Revision

This chapter revisits the multi-robot boundary coverage problem in which a group of k robots must inspect every point on the boundary of a two-dimensional environment. As a result of the considerable computational savings offered by the BF graph construction method, for graphs describing typical boundary coverage tasks, the kRPP heuristic requires only a few seconds to run to completion. This vast improvement in running time allows the kRPP to be solved quickly enough that robots might use it to revise their paths online in light of changes to the environment or team in a variety of related inspection tasks. In Section 4.1, modularity in the algorithm is exploited to propose a method for the revision of paths mid-task, allowing for adaptation of the inspection plan in reaction to changes in team size or the task assignment. Simulated examples and discussion are presented in Section 4.3 and open questions and possible extensions are summarized in Section 4.4.

4.1 Plan Revision

In the boundary coverage problem, detailed in Chapter 3, a group of k robots is required to completely inspect the boundary of all two-dimensional objects in a specified environment.

Additionally, one would practically seek inspection plans which balance the inspection load, as much as possible, across the cooperating robots. This section details a path-revision approach which exploits the reduced complexity of the graph representation and the modularity of the path planning algorithm to allow for plan revision in response to mid-task changes in robot team size and/or changes in the environment.

Such changes can occur in a number of practically important situations. The number of robots cooperating in the inspection task might change during the inspection task due to robot failure, the retasking of robots to another chore, or the addition of robots to the inspection team. Robots may encounter new objects that must be inspected, or the inspection assignment may be changed thereby changing the structure of G. In such cases, the remainder of the inspection task should be replanned to adapt to the characteristics of the newly resized inspection team. The replanning capability is a necessary step in making the approach more robust to failure and uncertainty and introducing complementary or collaborative subtasks to the problem. The first step in robustly handling these practical issues is to devise a method for plan revision in the course of task execution.

4.2 A Modular Algorithm to solve the kRPP

To revise the robots' paths mid-task, one can appeal to the modular structure of the planning heuristic presented in Section 3.3 to partially reuse prior calculations with modified inputs based on the current state of inspection.

4.2.1 Repartition Unvisited Required Edges

Let k' denote that number of cooperating robots that remain after one or more robots has failed, been retasked, or added. Similarly, let E'_R denote the set of unvisited required edges

at the time of revision. This set of edges includes edges that have not yet been visited in the original plan at the time of plan revision, as well as newly required edges that might arise from changes in the environment's geometry. The edges in E'_R must be repartitioned among the k' robots so that the inspection task can be completed.

The supervisory agent partitions the edges in E'_R into k' groups, $F'_1 \cup ... \cup F'_k = E'_R$ using the farthest point clustering algorithm described in Section 3.3.1.2. The k' representative edges used as the basis for partitioning are chosen based on the robots' current positions. For robot i already engaged in the inspection task and currently traversing the j^{th} edge of its tour, the representative edge f'_i will be edge $j+1$ in the robot's current tour. Representative edges for robots new to the task, which are deployed from the depot, are chosen from E'_R such that the sum of the minimum distance through G to the existing representative edges is maximized.

As in the initial planning process, after the robots are assigned their respective groups of required edges, the remainder of the planning procedure can take place online.

4.2.2 Make Subset of Edges in G Connected

At the time of path revision, because the robots will not necessarily start their new tours from the depot vertex, a new effective depot vertex must be specified for each robot. For robot i already engaged in the inspection task and currently traversing the j^{th} edge, the end vertex toward which the robot is moving will serve as the new depot $v'_{depot,i}$. For robots new to the task, which are deployed from the depot, $v'_{depot,i} = v_{depot}$ (this need not be the original depot, but for simplicity we assume it is). This step is executed for the groups $F'_1, ..., F'_k$, each connected to its respective $v_{depot,i'}$, as described for $F_1, ..., F_k$ and the single v_{depot} in Section 3.3.1.3.

4.2.3 Compute Tour of a Connected Subgraph

This step is executed for the subgraphs $G'_1, ..., G'_k$, as described for $G_1, ..., G_k$ in Section 3.3.1.4, yielding tours $T'_1, ..., T'_k$.

4.2.4 Refine a Tour in G

Because the new tours T'_i terminate at $v_{depot,i}$, if $v_{depot,i} \neq v_{depot}$, once every required edge in a tour has been visited, the rest of the tour is replaced with the shortest path to the original point of deployment, v_{depot}. As in Section 3.3.1.5, for each tour T'_i, sequences of edges that are retraced in the course of a tour are replaced with the shortest path in G, if a shorter path between the end vertices of the sequence exists.

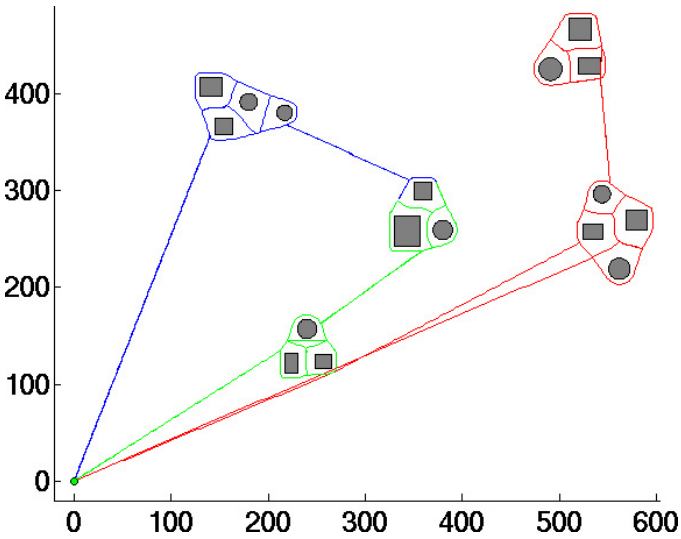
4.3 Simulation Results and Discussion

4.3.1 Revision: Change in Robot Team Size

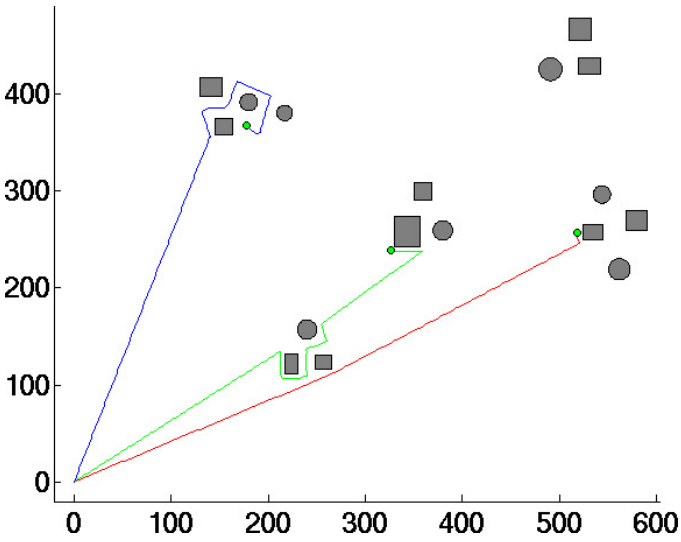
To illustrate the path revision method, consider the case of robot failure mid-task.

For the first example, consider the sparse environment with several distinct inspection regions, shown in Figure 4.1. The sensing parameters used are $r_{min} = 5, r_{max} = 25$. For $k = 3$, routes planned and executed successfully are shown in Figure 4.1(a), while Figure 4.1(b) illustrates the progress made at the time the robot carrying out the route shown in blue fails. The complete post-revision paths are shown in Figure 4.2. A second example environment, featuring a single, densely packed inspection region, with sensing parameters $r_{min} = 5, r_{max} = 50$, is shown for $k = 3$ in Figures 4.3 and 4.4.

In the first example, the edges that are "abandoned" by the failed robot are reassigned to the robot executing the route drawn in green, while in the second, the edges are divided



(a) Routes planned and carried out successfully, $k = 3$.



(b) Robot progress at the time of the blue robot's failure (Section 4.3.1) and at the time several edges are removed from E_R (Section 4.3.2).

Figure 4.1: Illustration of examples of path revision presented in Sections 4.3.1 and 4.3.2 for a sparse environment with several inspection regions.

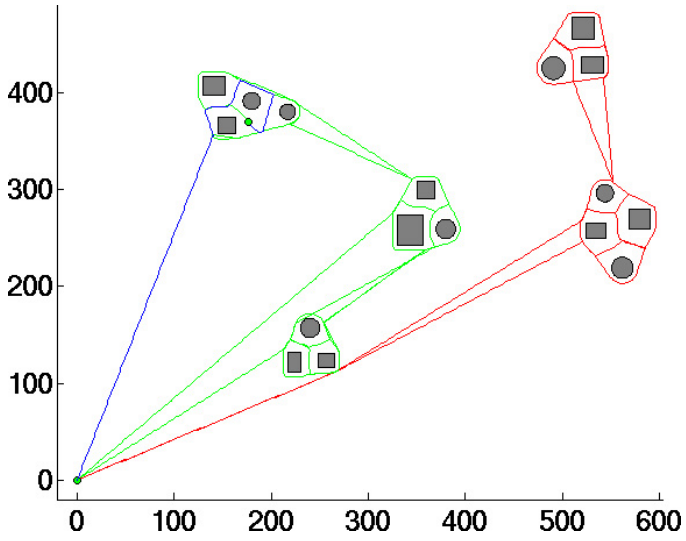
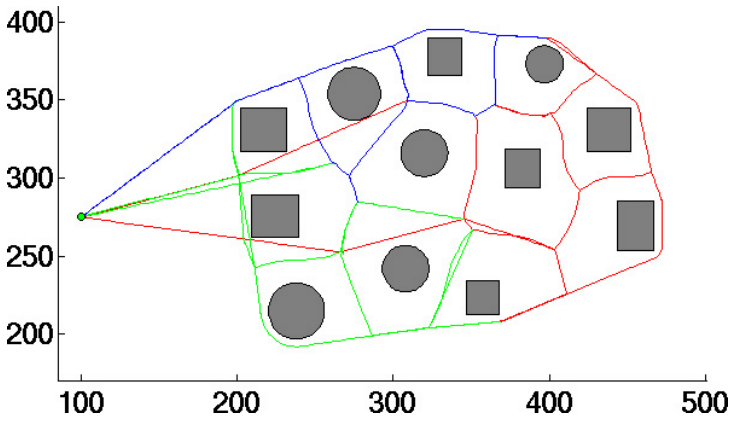


Figure 4.2: Routes carried out after path revision necessitated by change in k (robot on blue route fails).

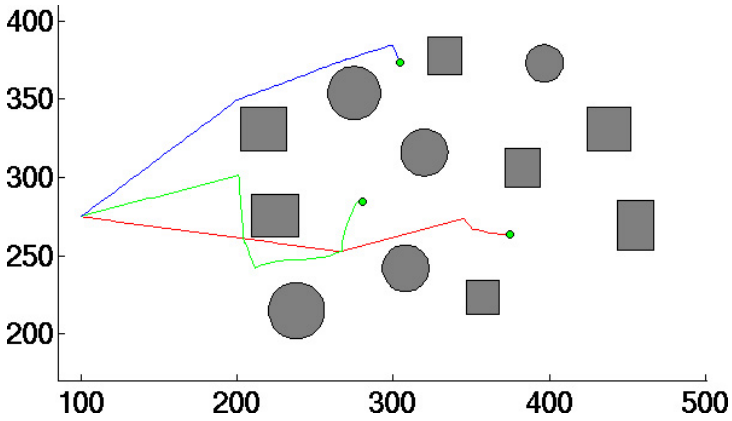
among both remaining robots. This is due to the nature of the partitioning process and to the structure of the environment: in the first example, at the time of failure, every required edge in the blue route is closer to the robot following the green route than the robot following the red. In contrast, the robots in the second example are routed through a more closely packed environment and the remaining robots share the abandoned edges. The repartitioning results in such cases depend more heavily on the robots' positions at the time of failure.

In the example of Figure 4.1, the differences in the pre- and post-revision red route are due to the fact that the route is replanned from a new "depot" point, even while the edges yet to be inspected remain the same. This replanning is not necessary, but is inexpensive when used with the new graph construction method and, as in this case, may improve upon the previously planned path.

In addition to the path revision method presented Section 4.1, a method for post-failure path revision which preserved the partitioning calculated in the initial execution of the



(a) Routes planned and carried out successfully, $k = 3$.



(b) Robot progress at the time of the blue robot's failure (Section 4.3.1) and at the time several edges are removed from E_R (Section 4.3.2).

Figure 4.3: Illustration of examples of path revision presented in Sections 4.3.1 and 4.3.2 for a more densely packed environment with a single inspection region.

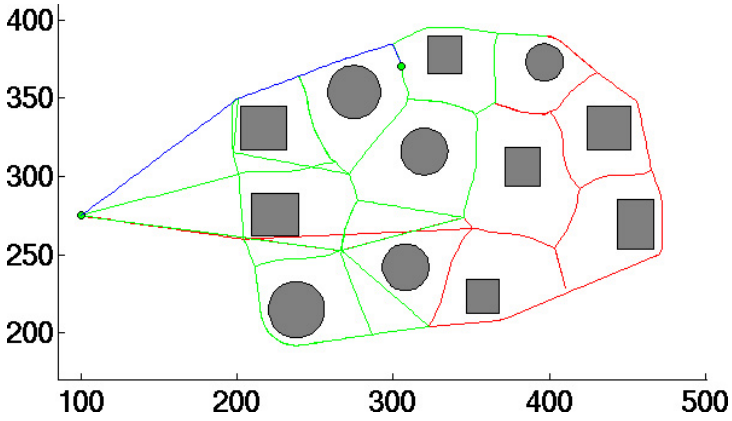


Figure 4.4: Routes carried out after path revision necessitated by change in k (robot on blue route fails).

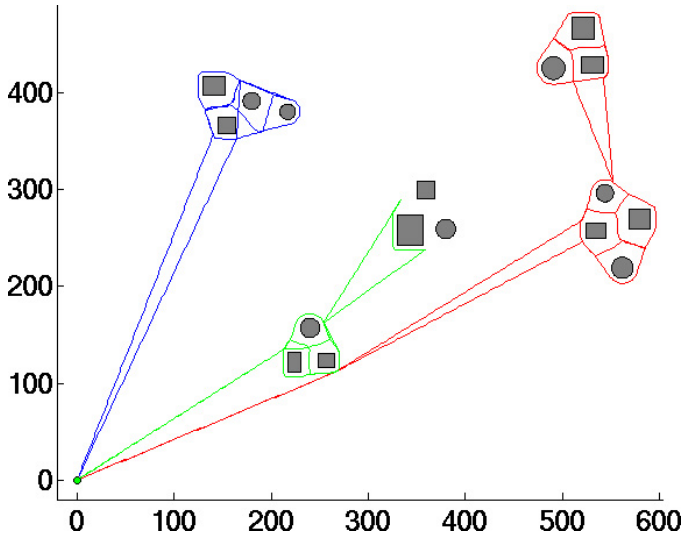


Figure 4.5: Routes carried out after path revision necessitated by change in E_R (several edges are removed from E_R).

constructive heuristic was also explored. This method repartitioned only the unvisited required edges in F_i initially assigned to the failed robot i. By reducing the number of edges being partitioned, the time complexity of the repartitioning step taken during path revision was significantly reduced. This method did not lend itself to extensions such as revision after addition of robots or changes to G, however, and when the savings in running time offered by this method offered were overwhelmed by the savings in running time due to the BF graph representation, the more computationally intensive but more flexible method presented in Section 4.1 was pursued instead.

4.3.2 Revision: Change in Inspection Task

Path revision can be a useful tool when information is acquired mid-task. Consider an example where, initially, every boundary point in our example environment must be inspected. The examples illustrated in Figures 4.1(a) and 4.3(a) are used again. Mid-task, again at the point illustrated in Figures 4.1(b) and 4.3(b), the planning system receives a

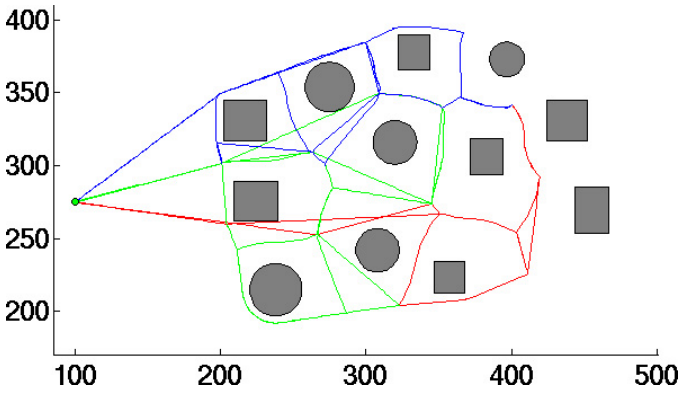


Figure 4.6: Routes carried out after path revision necessitated by change in E_R (several edges are removed from E_R).

command to ignore some of the boundary segments (for example, information about these segments comes from other sources, and thus they no longer need to be inspected). Figure 4.1 illustrates an example in which, mid-task, the supervisory agent instructs the team to ignore the boundary segments in the centermost inspection region. Figure 4.3 illustrates a case in which, mid-task, the supervisory agent instructs the team to ignore three rightmost objects. The edges associated with these objects are thus removed from E_R, and the traversal of the remaining unvisited edges in E_R is divided among the k robots by the path revision algorithm.

The complete post-revision paths, no longer incorporating inspection of these objects, are shown in Figures 4.5 and 4.6. In such cases, the revision of the robots' paths allows them to bypass now-unnecessary inspection and complete the task more quickly.

4.4 Conclusion

This chapter utilized the considerable computational savings offered by the new representation to explore plan revision. The path revision capability presented here is a step towards making the graph-based approach more robust to failure and open to the addition of com-

plementary or collaborative subtasks. The use of multiple robots offers the opportunity for transient, but potentially unplanned, collaborations between teammates, a task extension that may be explored in future work. For example, a robot with a complementary sensory suite might detour from its assigned task to aid another robot that has found an interesting target. Plan revision will be a necessary component in the implementation of such extensions. Plan revision necessitated by changes in the underlying graph representation will also be a necessary component in future work addressing changes to the known environment as unexpected features are detected during an inspection tour. The ability to replan while taking into account changes to the team and to the task itself with only a minor delay in task execution moves the implementation a significant step closer to online planning.

Chapter 5

Conclusion

5.1 Summary and Outlook

The multi-robot systems explored in this thesis range from fully distributed, swarm-inspired systems to centrally supervised, deliberative systems, adding a diverse set of "building blocks" to the field's ever-increasing collective understanding of multi-robot system design.

5.1.1 Swarm Robotic System Design

At the distributed and reactive end of the multi-robot system spectrum, Chapter 2 describes mathematical modeling methodologies developed to predict and optimize a robotic swarm's performance for several tasks. SI principles do not provide us with a way to quantitatively predict the swarm performance according to a particular metric or analyze further possible optimization margins and intrinsic limitations of this approach from an engineering point of view. In other words, if we want to achieve coordinated, self-organized group behavior based on local interactions, we need to have appropriate tools for understanding how to design and control individual units so that the swarm can achieve target behaviors and levels of performance. Models allow the engineer to capture the dynamics of these nonlinear, asynchronous, potentially large-scale systems at more abstract levels, sometimes achieving even

mathematical tractability. More generally, modeling is a means for saving time, enabling generalization to different robotic platforms, and estimating optimal system parameters, including control parameters and number of agents in a team. This work, previously published in a collaborator's thesis [35] and in two journal papers [36, 37], was one of the first attempts to develop an ad hoc modeling methodology for swarm robotic systems.

The strength of this research lies in the microscopic and macroscopic models' quantitatively correct predictions in the absence of free parameters. The predictions are validated with embodied simulations and experiments with real robots. The mapping between different implementation levels of the same experiment is precisely described, allowing the methodology's application to other experiments. Furthermore, the application of the probabilistic modeling methodology is illustrated for several examples of incremental complexity, providing additional building blocks that may be applied to the construction of models for other experiments. The strengths and limitations of the modeling methodology are extensively explored, particularly in comparison to other popular simulation tools such as sensor-based, embodied simulators.

Since the publication of this work, collaborators have made several notable improvements to the modeling methodology and applied it to several more swarm robotic experiments, including a swarm approach to the boundary inspection problem considered in Chapter 3.

5.1.2 Planning Algorithms for Multi-robot Inspection

At the centralized, deliberative end of the multi-robot system spectrum, Chapter 3 formalizes the multi-robot boundary coverage problem, in which a group of k robots is required to completely inspect the boundary of all two-dimensional objects in a specified environment. The boundary coverage problem is relatively new; I am the first to formalize the problem

and offer a complete algorithm for its solution. The method abstracts the problem into a graph-based framework. Within this framework, the task can be posed as a "postman"-type graph problem whose solution describes a complete inspection plan. Initially, a centralized supervisory agent divides the task among the robots, each of which computes its own graph solution-based inspection plan using a constructive heuristic algorithm. Improvements to the initial formulation of the graph representation decrease the required running time of the planning algorithm by two orders of magnitude, allowing the possibility of using this planning algorithm as an online route planning tool in addition to its current application as a pre-deployment route planning method.

My continuing research focuses on the extension of this boundary inspection approach to the case of long-term deployment for surveillance applications that require repetitive coverage. Motivating applications include security surveillance within structured environments and the execution of periodic tasks, such as mail delivery or trash collection.

Using the building blocks from the boundary inspection approach, I am exploring corridor inspection tasks posed within a graph framework compatible with the boundary coverage graph framework, allowing for the straightforward reuse of the planning tools implemented for boundary coverage. The main focus of this research to develop decentralized methods for adaptation of routes in response to changes in team size, task assignment, or the environment over the course of the extended surveillance period.

In considering repetitive coverage, the opportunity for robots to improve and adapt their routes arises. As repetitive coverage is desirable rather than an inefficient redundancy, the quality of the initial plan may be less critical and will likely be subject to different metrics than were considered for a single instance of boundary coverage.

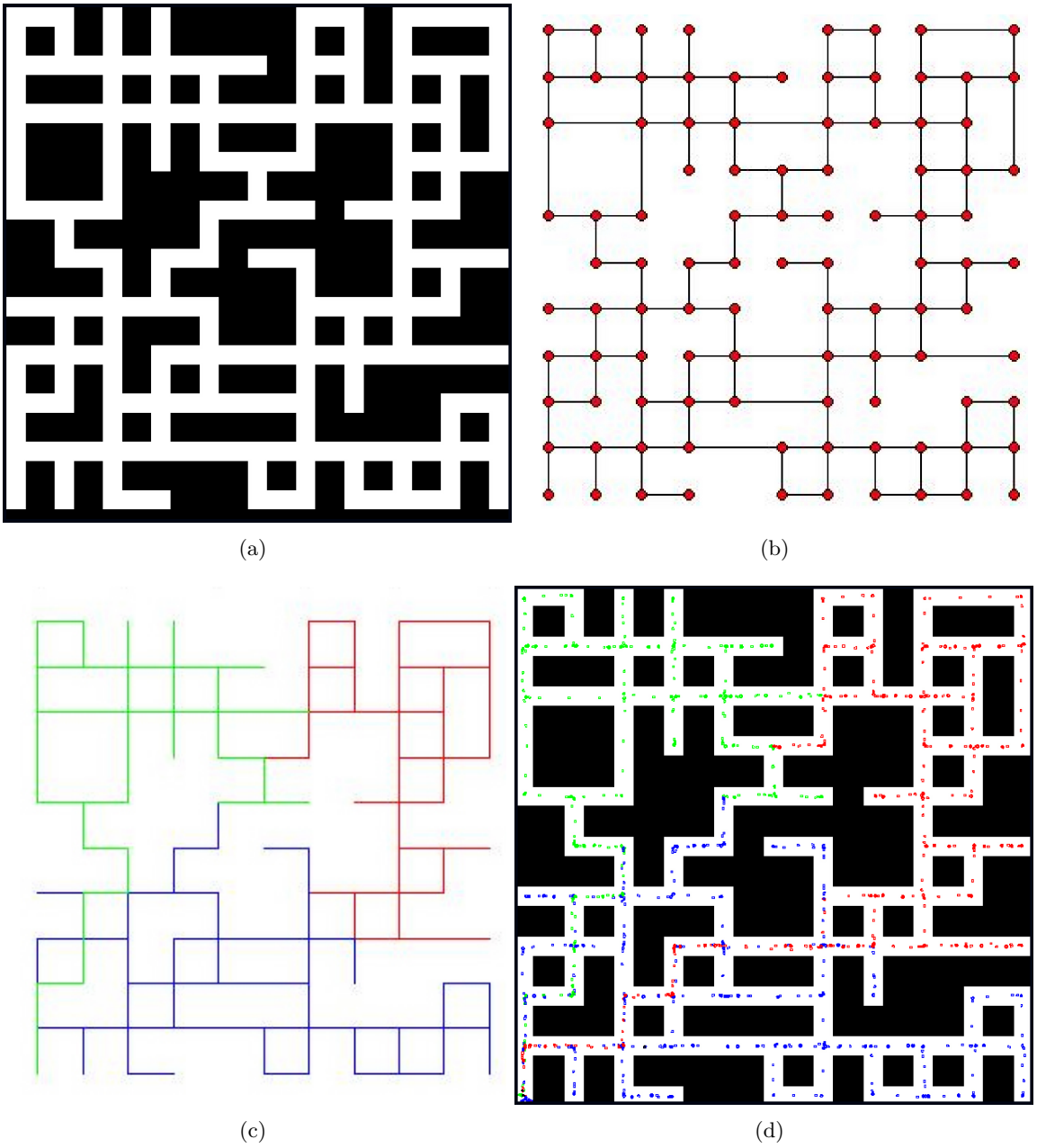
A useful metric for tracking the team performance in such a task is to consider how

often specific features are visited (or edges are traversed). For such a metric, a single-robot solution executed by a team of robots deployed at even time intervals (i.e., the task is temporally partitioned) could achieve performance similar to that achieved by a spatially-partitioned multi-robot solution, so a metric demonstrating the benefit of the multi-robot solution is also necessary. Robot capacity is such a metric; if the task at hand is to collect trash, a robot patrolling a shorter multi-robot route more frequently will require less storage capacity than a robot patrolling a long single-robot route. A spatial partitioning approach will also reduce interference between robots at common points on the graph.

To facilitate this continuing work, the boundary inspection planning methods have been implemented in Player/Stage [83] and extended to a corridor coverage task, a more intuitive choice than boundary coverage for a repetitive coverage task. For a large value of r_{max} relative to the width of a corridor, a corridor environment's graph representation would resemble the graph representation for a boundary coverage task inside a densely packed inspection region. Corridor-based test environments lend themselves easily to a graph representation, with edges representing the inspection of hallways and vertices representing hallway intersections. Test environments with wide variations in size and complexity can also be quickly generated for the purposes of systematic testing of the inspection methods. Figure 5.1 illustrates the implementation of corridor coverage in Player/Stage, posed within the graph framework developed for boundary coverage.

5.2 Towards Intelligent, Autonomous Multi-agent Systems

My long-term research interests lie in the design and control of real-world multi-agent systems, endowing interacting mobile and embedded systems with intelligence and autonomy while safely keeping humans in control. For a variety of applications, the capability of simul-



(a) (b)

(c) (d)

Figure 5.1: Corridor coverage in Player/Stage, posed within the graph framework developed for boundary coverage. (a) The corridor test environment. (b) The graph representation. (c) Planned inspection routes for $k = 3$. (d) Screenshot from Player/Stage in which colored "trail markers" indicate where each robot traveled in the course of inspection.

taneous sensing and action in multiple locations that is inherent to multi-robot approaches offers potential advantages over single robot systems in robustness, efficiency, and application feasibility. With the proliferation of wireless networks and the increasing prevalence of intelligent consumer devices, I see a wide array of future opportunities for real-world multi-agent systems to assist humans in everyday tasks in addition to traditional "dull, dirty, or dangerous" robotics applications. Work remains to be done in classic coverage tasks, such as collaborative mapping, surveillance, and demining. There are also opportunities to extend coverage methods to allow for automated inspection of pipelines, cables, and various structures. Intelligent transportation systems could benefit from multi-agent approaches, such as congestion control managed collaboratively by passenger vehicles and embedded sensor networks along roadways. Personal networks of intelligent devices will have a variety of consumer applications. The cooperative deployment of sensor-network-based communication and localization infrastructure with a team of robots to carry out exploration and observation tasks in hazardous environments, such as the deep sea or surfaces of other planets, could advance the frontiers of various fields in biology, geology, and planetary science.

For every multi-agent application, we must find the correct balance of robustness, efficiency, simplicity, and sophistication. As we add to the existing catalogue of building blocks for a variety of robot tasks, we may use them to design and implement incrementally more complex multi-robot systems while we gain a better understanding of methods for their formal classification and analysis. In my research, I will continue to look beyond implementation and validation of a particular system and ask how its design can contribute to the development of a more general design methodology.

Bibliography

[1] C. V. Jones, D. A. Shell, M. J. Matarić, and B. P. Gerkey, "Principled approaches to the design of multi-robot systems," in *Workshop on Networked Robotics, International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 71–80.

[2] B. Gerkey and M. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[3] R. Beckers, O. E. Holland, and J.-L. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," in *Fourth Workshop on Artificial Life*, B. R. and M. P., Eds. Boston, MA, USA: The MIT Press, 1994, pp. 181–189.

[4] A. Martinoli, A. J. Ijspeert, and F. Mondada, "Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots," *Robotics and Autonomous Systems*, vol. 29, pp. 51–63, 1999.

[5] A. Martinoli, A. J. Ijspeert, and L. M. Gambardella, "A probabilistic model for understanding and comparing collective aggregation mechanisms," in *Fifth European Conf. on Artificial Life*, ser. Lectures Notes in Computer Science, D. Floreano, F. Mondada, and J.-D. Nicoud, Eds., Lausanne, Switzerland, 1999, pp. 575–584.

[6] O. Holland and C. Melhuish, "Stigmergy, self-organisation, and sorting in collective robotics," *Artificial Life*, vol. 5, pp. 173–202, 1999.

[7] M. J. B. Krieger and J.-B. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 65–84, 2000.

[8] M. J. Matarić, "Interaction and intelligent behavior," Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA," Doctoral Dissertation, 1994.

[9] A. J. Ijspeert, A. Martinoli, A. Billard, and L. Gambardella, "Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment," *Autonomous Robots*, vol. 11, no. 2, pp. 149–171, 2001.

[10] A. Martinoli and F. Mondada, "Collective and cooperative group behaviours: Biologically inspired experiments in robotics," in *Fourth International Symp. on Experimental Robotics ISER-95*, ser. Lecture Notes in Control and Information Sciences, O. Khatib and J. K. Salisbury, Eds., Stanford, CA, USA, 1995, pp. 3–10.

[11] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowoski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems)*, vol. 18, no. 5, pp. 813–825, 2002.

[12] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems)*, vol. 18, no. 5, pp. 758–768, 2002.

[13] C. R. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 85–101, 2000.

[14] L. E. Parker, "Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance," *Autonomous Robots*, vol. 8, pp. 239–267, 2000.

[15] J. Fredslund and M. J. Matarić, "General algorithm for robot formations using local sensing and minimal communication," *IEEE Trans. on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems)*, vol. 18, no. 5, pp. 837–846, 2002.

[16] I. D. Kelly and D. A. Keating, "Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots," in *Third Conf. on Mechatronics and Machine Vision in Practice*, vol. 1, Gaimardes, Portugal, 1996, pp. 1–4.

[17] A. T. Hayes, A. Martinoli, R. M. Goodman, H. T. Nagle, J. W. Gardner, and K. Persaud, "Distributed odor source localization," *IEEE Sensors (Special Issue on Artificial Olfaction)*, vol. 2, no. 3, pp. 260–271, 2002.

[18] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Swarm robotic odor localization: Off-line optimization and validation with real robots," *Robotica (Special Issue on Bio-Inspired Robotics)*, vol. 21, pp. 427–441, 2003.

[19] A. Billard, A. J. Ijspeert, and A. Martinoli, "A multi-robot system for adaptive exploration of a fast changing environment: Probabilistic modelling and experimental study," *Connection Science (Special Issue on Adaptive Robots)*, vol. 11, no. 3/4, pp. 359–379, 1999.

[20] B. Yamauchi, "Decentralized coordination for multi-robot exploration," *Robotics and Autonomous Systems*, vol. 29, no. 1, pp. 111–118, 1999.

[21] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999.

[22] T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel, "Cs freiburg: Coordinating robots for successful soccer playing," *IEEE Trans. on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems)*, vol. 18, no. 5, pp. 685–699, 2002.

[23] G. Beni and J. Wang, "Swarm intelligence," in *Seventh Annual Meeting of the Robotics Society of Japan*, Tokyo, Japan, 1989, pp. 425–428.

[24] A. Martinoli, G. Theraulaz, and J.-L. Deneubourg, "Quand les robots imitent la nature," *La Recherche*, vol. 358, pp. 56–62, 2002.

[25] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, ser. SFI Studies in the Science of Complexity. New York, NY: Oxford University Press, 1999.

[26] J. K. Parrish and W. M. Hamner, *Animal Groups in Three Dimensions*. Cambridge University Press, 1997.

[27] S. Kazadi, A. Abdul-Khaliq, and G. R. M., "On the convergence of puck clustering systems," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 93–117, 2002.

[28] K. Lerman and A. Galstyan, "Mathematical model of foraging in a group of robots: Effect of interference," *Autonomous Robots*, vol. 13, no. 127-141, 2002.

[29] K. Sugawara and M. Sano, "Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system," *Physica D*, vol. 100, pp. 343–354, 1997.

[30] K. Sugawara, M. Sano, I. Yoshihara, and K. Abe, "Cooperative behavior of interacting robots," *Artificial Life and Robotics*, vol. 2, pp. 62–67, 1998.

[31] W. Agassounon, A. Martinoli, and R. M. Goodman, "A scalable, distributed algorithm for allocating workers in embedded systems," in *IEEE Conf. on Systems, Man and Cybernetics SMC-01*, Tuscon, AZ, USA, 2001, pp. 3367–3373.

[32] K. Lerman, A. Galstyan, A. Martinoli, and A. J. Ijspeert, "A macroscopic analytical model of collaboration in distributed robotic systems," *Artificial Life*, vol. 7, no. 4, pp. 375–393, 2001.

[33] A. Martinoli and K. Easton, "Modeling swarm robotic systems," in *Eighth International Symp. on Experimental Robotics ISER-02*, ser. Springer Tracts in Advanced Robotics (2003), B. Siciliano and P. Dario, Eds., Sant'Angelo d'Ischia, Italy, 2002, pp. 297–306.

[34] ——, "Optimization of swarm robotic systems via macroscopic models," in *Second International Workshop on Multi-Robot Systems*, A. C. Schultz, L. E. Parker, and F. E. Schneider, Eds., Washington, DC, USA, 2003, pp. 181–192.

[35] W. Agassounon, "Modeling artificial, mobile swarm systems," Doctoral Dissertation, California Institute of Technology, 2003.

[36] A. Martinoli, K. Easton, and W. Agassounon, "Modeling of swarm robotic systems: A case study in collaborative distributed manipulation," *International Journal of Robotics Research (Special Issue on Experimental Robotics)*, vol. 23, no. 4, pp. 415–436, 2004.

[37] W. Agassounon, A. Martinoli, and K. Easton, "Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes," *Autonomous Robots (Special Issue on Swarm Robotics)*, vol. 17, no. 2-3, pp. 163–191, 2004.

[38] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *IEEE International Conference on Robotics and Automation ICRA-05*, 2005, pp. 735–741.

[39] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 7–40, 2001.

[40] Z. J. Butler, "Distributed coverage of rectilinear environments," Doctoral Dissertation, Carnegie Mellon University, 2000.

[41] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[42] N. Correll and A. Martinoli, "Modeling and optimization of a swarm-intelligent inspection system," in *Seventh International Symp. on Distributed Autonomous Robotic Systems DARS-04*. Springer Verlag, 2004, pp. 369–378.

[43] ——, "Collective inspection of regular structures using a swarm of miniature robots," in *Ninth International Symp. on Experimental Robotics ISER-04*. Springer Tracts in Advanced Robotics, 2004, pp. 375–385.

[44] ——, "Modeling and analysis of beaconless and beacon-based policies for a swarm-intelligent inspection system," in *IEEE International Conference on Robotics and Automation ICRA-05*, Barcelona, Spain, 2005, pp. 2488–2493.

[45] ——, "Towards optimal control of self-organized robotic inspection systems," in *Eighth International IFAC Symp. on Robot Control SYROCO-06*, 2006.

[46] Y. Zhang, E. K. Antonsson, and A. Martinoli, "Evolving neural controllers for collective robotic inspection," *Proc. of the 9th Online World Conf. on Soft Computing in Industrial Applications*, pp. 721–733, 2004.

[47] J. A. Fax, "Optimal and cooperative control of vehicle formations," Doctoral Dissertation, California Institute of Technology, 2002.

[48] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 6, pp. 905–908, 2001, 1042-296X.

[49] R. Olfati-Saber and R. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *IEEE Conference on Decision and Control CDC-02*, 2002.

[50] E. Klavins, "Graph grammars for self assembling robotic systems," in *IEEE International Conference on Robotics and Automation ICRA-04*, New Orleans, LA, USA, 2004.

[51] ——, "Communication complexity of multi-robot systems," in *Workshop on the Algorithmic Foundations of Robotics WAFR-02*, New York, NY, USA, 2002.

[52] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2 - 4, pp. 181–196, 2004.

[53] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme, "A unifying framework for tunable topology control in sensor networks," University of Southern California, Center for Robotics and Embedded Systems Technical Report, Tech. Rep. CRES-05-004, 2005.

[54] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *National Conference on Artificial Intelligence AAAI-96*, Portland, OR, USA, 1996.

[55] A. Howard, "Multi-robot mapping using manifold representations," in *IEEE International Conference on Robotics and Automation ICRA-04*, New Orleans, LA, USA, 2004.

[56] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1 - 4, pp. 77–98, 2001.

[57] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000, 0018-9219.

[58] P. Tabuada, G. J. Pappas, and P. Lima, *Composing Abstractions of Hybrid Systems*, ser. Lecture Notes in Computer Science, 2002, 2289.

[59] R. Chauvin and P. Janin, "Facteurs de direction et d'excitation au cours de l'accomplissement d'une tche chez formica polyctena," *Insectes Sociaux*, vol. 22, pp. 199–206, 1975.

[60] F. Mondada, E. Franzi, and P. Ienne, "Mobile robot miniaturization: A tool for investigation in control algorithms," in *Third International Symp. on Experimental Robotics*, ser. Lecture Notes in Control and Information Sciences, T. Yoshikawa and F. Miyazaki, Eds. Kyoto, Japan: Springer Verlag, 1993, pp. 501–513.

[61] O. Michel, "Webots: Symbiosis between virtual and real mobile robots," in *First International Conf. on Virtual Worlds*, J.-C. Heuding, Ed. Paris, France: Springer Verlag, 1998, pp. 254–263.

[62] A. Martinoli, "Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies," EPFL," Doctoral Dissertation, 1999.

[63] P.-P. Grassé, "La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la thorie de la stigmergie: essai dinterprtation du comportement des termites constructeurs," *Insectes Sociaux*, vol. 6, no. 41-80, 1959.

[64] N. Correll and A. Martinoli, "System identification of self-organizing robotic swarms," in *Eighth International Symp. on Distributed Autonomous Robotic Systems DARS-06*, 2006.

[65] L. Chrétien, "Organisation spatiale du matériel provenant de lexcavation du nid chez messor marbarus et des cadavres chez lasius niger (hymenopterea: Formicidea)," Doc-

toral Dissertation, Université Libre de Bruxelles, Belgium, 1996.

[66] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Comparing distributed exploration strategies with simulated and real autonomous robots," in *Fifth International Symp. on Distributed Autonomous Robotic Systems DARS-00*, L. E. Parker, G. Bekey, and J. Bahren, Eds., Knoxville, TN, USA, 2000, pp. 261–270.

[67] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems," in *Swarm Robotics Workshop: State-of-the-art Survey*, E. Sahin and W. Spears, Eds., vol. 3342. Springer-Verlag, 2005, pp. 143–152.

[68] K. Easton and A. Martinoli, "Efficiency and optimization of explicit and implicit communication schemes in collaborative robotics experiments," in *IEEE International Conf. on Intelligent Robots and Systems IROS-02*, Lausanne, Switzerland, 2002, pp. 2795–2800.

[69] A. T. Hayes, "Self-organized robotic system design and autonomous odor localization," Doctoral Dissertation, California Institute of Technology, 2002.

[70] L. Li, A. Martinoli, and Y. Abu-Mostafa, "Learning and measuring specialization in collaborative swarm systems," *Adaptive Behavior (Special Issue on Mathematics and Algorithms of Social Interactions)*, vol. 12, no. 3-4, pp. 199–212, 2004.

[71] K. Easton and J. Burdick, "A coverage algorithm for multi-robot boundary inspection," in *IEEE International Conference on Robotics and Automation ICRA-05*, Barcelona, Spain, 2005.

[72] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[73] M.-K. Kwan, "Graphic programming using odd or even points," *Chinese Math*, no. 1, pp. 273–277, 1962.

[74] J. Edmonds, "The chinese postman problem," *Operations Research*, vol. 13, no. Suppl. 1, pp. B73–B77, 1965.

[75] J. Edmonds and E. L. Johnson, "Matching, euler tours, and the chinese postman," *Mathematical Programming*, vol. 5, pp. 88–124, 1973.

[76] C. S. Orloff, "A fundamental problem in vehicle routing," *Networks*, vol. 4, pp. 35–64, 1974.

[77] D. Ahr and G. Reinelt, "New heuristics and lower bounds for the min-max k-chinese postman problem," in *Algorithms-Esa 2002, Proceedings*, ser. Lecture Notes in Computer Science, 2002, vol. 2461, pp. 64–74.

[78] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," *SIAM Journal on Computing*, vol. 7, no. 2, pp. 178–193, 1978.

[79] J. K. Lenstra and A. H. G. R. Kan, "On general routing problems," *Networks*, vol. 6, pp. 273–280, 1976.

[80] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.

[81] A. Gibbons, *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[82] J. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem," *Proc. American Math. Society 7*, pp. 48–50, 1956.

[83] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *International Conference on Advanced Robotics ICAR-03*, Coimbra, Portugal, 2003, pp. 317–323.