A SYSTEMATIC STUDY OF PROBABILISTIC AGGREGATION STRATEGIES IN SWARM ROBOTIC SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR SOYSAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Erol Şahin
Supervisor

Examining Committee Members

Prof. Dr. Faruk Polat (METU, CENG) _____

Assist. Prof. Dr. Erol Şahin (METU, CENG) _____

Assoc. Prof. Dr. Göktürk Üçoluk (METU, CENG) _____

Assist. Prof. Dr. A. Buğra Koku (METU, ME) _____

Assist. Prof. Dr. Veysel Gazi (TOBB ETU, EE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Onur Soysal

Signature :

ABSTRACT

A SYSTEMATIC STUDY OF PROBABILISTIC
AGGREGATION STRATEGIES IN SWARM ROBOTIC SYSTEMS

Soysal, Onur

M.S., Department of Computer Engineering
Supervisor : Assist. Prof. Dr. Erol Şahin

September 2005, 52 pages

In this study, a systematic analysis of probabilistic aggregation strategies in swarm robotic systems is presented. A generic aggregation behavior is proposed as a combination of four basic behaviors: *obstacle avoidance*, *approach*, *repel*, and *wait*. The latter three basic behaviors are combined using a three-state finite state machine with two probabilistic transitions among them. Two different metrics were used to compare performance of strategies. Through systematic experiments, how the aggregation performance, as measured by these two metrics, change 1) with transition probabilities, 2) with number of simulation steps, and 3) with arena size, is studied.

Keywords: Aggregation, Swarm Robotics, Systematic Experiments, Simulation

ÖZ

OĞUL ROBOTBİLİM SİSTEMLERİNDE, RASLANTISAL TOPLANMA STRATEJİLERİ ÜZERİNE SİSTEMATİK BİR ÇALIŞMA

Soysal, Onur

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. Erol Şahin

Eylül 2005, 52 sayfa

Bu çalışmada, oğul robotbilim sistemlerinde, raslantısal toplanma stratejilerinin sistematik bir analizi sunulmuştur. *Nesnelerden kaçma, yaklaşma, kaçma* ve *bekleme* olarak tanımlanan dört basit davranışın bileşiminden oluşan, geniş kapsamlı bir davranış önerilmiştir. Son üç basit davranış, iki rastlantısal geçişi olan bir üç durumlu bir sonlu durum makinesi aracılığıyla birleştirilmiştir. Değişik stratejileri birleştirmek için, iki farklı ölçüt kullanılmıştır. Sistematik deneyler kullanılarak, 1) geçiş olasılıklarıyla, 2) benzetimin adım sayısıyla, 3) arenanın büyüklüğüyle, toplanma davranışının performansının, bu iki ölçüte göre, nasıl değiştiğini incelenmiştir.

Anahtar Kelimeler: Toplanma, Oğul Robotbilim, Sistematik Deneyler, Benzeşim

To my parents and my younger brother

ACKNOWLEDGMENTS

This is probably the hardest part of the thesis for me, since I am alone in writing this chapter. The rest of this study would not be possible without sincere support and supervision of Assist. Prof. Dr. Erol Şahin. I really feel lucky that I was able to study with him.

I must thank Erkin Bahçeci for his patience and cooperation all through this study. I must thank, Emre Uğur for he endured my know-it-all attitude, never missed happy hours and, of course, gave me LaTeX support. I lost many Matlab CD's of Levent Bayındır. Mehmet Remzi Doğar, implemented a stable WebBibTeX and Hande Çelikkanat added hundreds of entries to this system, which was invaluable for organizing the references in this study. Özgür Gülderen and Murat Yükselen did a great job in implementing an XML parser for ODE. I would like to thank, Maya Çakmak and Ali Emre Turgut for building the prototype of the robot, that can be used for implementation of this study in real robots in future. KOVAN lab. have been a great place to work in, thanks to these people.

Onur Temizsoylu, from TÜBİTAK ULAK-BİM High Performance Computing Center, supported me on technical details while using the cluster. I would like to thank, Veysel Gazi for his guidance in mathematical approaches to aggregation problem.

Şafakcan Tunçdemir, listened every little detail about this study for two years. Ömer Emre Tokel, had been beside me in my darkest hours and gave me courage and strength to continue when I needed the most.

This work was partially funded by the SWARM-BOTS project, a Future and Emerging Technologies project (IST-FET) of the European Community, under grant IST-2000-31010, and by the "Kontrol Edilebilir Robot Oğulları" Career Project awarded to Erol Şahin (104E066) by TÜBİTAK (The Turkish Scientific and Technical Research Council).

The computational resources used in this study were provided by the TÜBİTAK ULAK-BİM High Performance Computing Center. This study would not be possible without their support.

Finally, my family has supported and encouraged me all through these three long years, without them I would definitely fail. Thank you all.

TABLE OF CONTENTS

PLAGIARISM . . . iii

ABSTRACT . . . iv

ÖZ . . . v

DEDICATON . . . vi

ACKNOWLEDGMENTS . . . vii

TABLE OF CONTENTS . . . ix

LIST OF FIGURES . . . xi

CHAPTER

1 INTRODUCTION . . . 1

1.1 Autonomous Robots . . . 2

1.2 Multi-robot Systems and Swarm Robotics . . . 4

2 AGGREGATION PROBLEM . . . 7

2.1 Related Work . . . 9

3 SIMULATION ENVIRONMENT . . . 12

3.1 Porting Swarmbot3D to ODE . . . 13

3.1.1 Physics-based Simulation . . . 14

3.1.2 KODEX . . . 15

3.2 Simulated Robot Model . . . 18

3.2.1 Actuator Model . . . 18

3.2.2 Sensor Models . . . 19

4 AGGREGATION BEHAVIOR . . . 21

4.1 A Minimal Aggregation Behavior . . . 21

5 EXPERIMENTS . . . 26

5.1 Aggregation Performance Metrics . . . 27

5.1.1 Expected Cluster Size . . . 27

5.1.2 Total Distance . . . 28

5.2 Experimental Setup . . . 29

5.3 Effect of Behavior Transition Probabilities . . . 32

5.3.1 Strategy 1 33
5.3.2 Strategy 2 37
5.3.3 Strategy 3 37
5.3.4 Strategy 4 38
5.4 Effect of Time 40
5.5 Effect of Arena Size 44
6 CONCLUSION 47
REFERENCES 49

LIST OF FIGURES

FIGURES

Figure 2.1 Aggregation behavior in nature 8
Figure 2.2 Motivation of aggregation behavior 8

Figure 3.1 A screen shot from simulator 13
Figure 3.2 Comparison of Swarmbot3D using Vortex and KODEX 14
Figure 3.3 Screen shots from Vortex and KODEX versions of Swarmbot3D . . . 14
Figure 3.4 A schematic drawing of the robot model 19
Figure 3.5 Change in sound perceived by a robot 20

Figure 4.1 Minimal Aggregation behavior 22
Figure 4.2 Perceptual aliasing in sound sensors 25

Figure 5.1 A screenshot of the standard arena 30
Figure 5.2 Flow of Experiments 31
Figure 5.3 Effect of behavior transition probabilities 33
Figure 5.4 Effect of controller parameters for ECS Metric 34
Figure 5.5 Effect of controller parameters for TD Metric 35
Figure 5.6 Histograms for ECS metric 36
Figure 5.7 Finite State Machine representations of the strategies 36
Figure 5.8 Progress of the representative strategies. 39
Figure 5.9 Effect of time on metric values 40
Figure 5.10 Details of effect of time on ECS metric 41
Figure 5.11 Details of effect of time on TD metric values 41
Figure 5.12 Effect of time on ECS with high threshold 42
Figure 5.13 Details of effect of time on ECS with high threshold 43
Figure 5.14 Total distance traveled by robots with respect to time 43
Figure 5.15 Effect of arena size on metric values 44
Figure 5.16 Effect of arena size on ECS Metric 45
Figure 5.17 Effect of arena size on TD Metric 45

CHAPTER 1

INTRODUCTION

The idea of automata, or "self-operating physical artifacts" dates back to ancient times. In Jewish mythology Golem [1], a creature "built" from human parts, appears in a human-like form performing menial tasks. However, the design and building of such automata took centuries. First examples of such automata were built for entertainment purposes. In 1738, Jacques de Vaucanson built an automaton [2] which can play different songs, coded on mechanical disks, with a flute. In Japan, Hosokawa Hanzo built a tea serving doll [3] in 18th century. When a cup is put on the tray of the doll moved forward to "serve" the tea to the guest. When the guest finished his tea, placing the cup back on the tray, the doll bowed, turned back and returned to the owner. These automata can be considered as the early precursors of today's robots.

The term "robot" was first coined by Karel Kapek, in one of his plays written in 1920, to name artificial humans built to perform menial labor. However in time, the term robot moved on to describe the whole class of mechanical devices that can perform "automated" tasks. During half of the last century, the progress in robotics has been accelerating at an increasing pace, thanks to the advances in mechanical and electrical engineering and computer science.

Today, the robots have started to get into our lives to take over tasks that are too "dirty, difficult, dangerous and dull (repetitive)" for us. Autonomous vacuum cleaners, like Roomba (iRobot Inc., USA), are being sold as household items, mobile robots like, Spirit and Opportunity [4], enable the exploration of Mars under dangerous environment conditions, robotics arms have proved themselves as reliable and cost-effective workers on manufacturing lines. Also, during the recent years, robots have also made a successful entry into the consumer market as entertainment toys. Aibo (Sony, Japan) is a good example. Companies like Honda, developed humanoid

robots for advertisement purposes, hinting the competence for developing high technology.

The robots, which exist in a wide range of domains as briefly overviewed above, can roughly be classified into two major categories: industrial and autonomous. *Industrial robots* can be described as programmable manipulators which can perform repetitive tasks that require precision and speed. Since 1950's these robots have been employed in manufacturing. Industrial robots usually do not sense their environment and make many assumptions about the state of the world. Therefore these robots require highly structured environments and their working domains are usually confined to factory floors.

In this thesis, we are interested in *autonomous robots*. These robots are aimed to operate in environments as unstructured and dynamic as in our daily lives. Unlike industrial robots, these robots cannot afford to make many assumptions about their operation environment. These robots use sensing devices to obtain information about their environment. This information is used to choose between different actions. Furthermore autonomous robots can construct models of the environment and plan using these models, enabling them to anticipate and adapt to different conditions.

1.1 Autonomous Robots

One of the first working examples of the autonomous robots are the "tortoises" of Walter Grey Walter [5], built during 1950's. These robots were controlled by an analog circuitry that included two vacuum tubes as its main processing unit. Despite their limited computational capability, the tortoises were able to move around without getting stuck and could move towards light sources.

With the developments in digital computers, the main challenge in autonomous robots has shifted from the mechatronics development of the robots towards the control of the robot. In 1970's, robots emerged as a platform for artificial intelligence studies. In line with the contemporary view of intelligence of that time, a number of control architectures, like STRIPS [6], HACKER [7] and NOAH [8] were proposed. In these architectures, sensor data is used to construct a symbolic model of the world. A plan is constructed within this world model and the plan is then executed using

the actuators of the robot. The robots controlled with these architectures such as, "Shakey" in late 1960's [9], "HILARE" in 1977 [10] and "Stanford Cart" in 1977 [11], had difficulties in dealing with the uncertainty and noise in the environment and were observed to be slow in responding to changes in the environment [12].

In response to the problems experienced with the early architectures, reactive control architectures were proposed. These systems promoted *"the tight coupling between perception and action, typically in the context of motor behaviors, to produce timely robotic response in dynamic and unstructured worlds"* [12]. Rodney Brooks, one of the first proponents of such architectures emphasized three fundamental principles for autonomous robots: *situatedness*, *embodiment* and *emergence*. Situatedness principle underlined the reality of robot, rather than being an abstract entity working with arbitrary symbols. Embodiment on the other hand expressed the physical presence of robot, with its physical limitations. Emergence pointed out that intelligence is not a stand-alone property but is the result of interactions between robot and its environment [13].

An indicative example of reactive architectures is the *subsumption architecture* proposed by Brooks [14]. In this architecture, the main processing unit is a *behavior* which can be defined as a mapping from the sensors to the actuators of the robot. The subsumption architectures dictates that behaviors are organized in a layered way and that higher-level behaviors have the ability to "subsume" the lower level behaviors.

In its purest form, the reactive architectures had denounced any kind of world modeling. However as the limitations of these architectures arose in time, hybrid architectures which combine the world modeling and planning properties of earlier architectures with the responsiveness of reactive architectures are developed.

In all these different architectures, the challenge is how to design the controller of a robot such that it can perform a desired task in an unstructured and dynamic environment. This problem is difficult since, the relationship between a desired behavior and the controller of the robot can be very complex. This is due to the fact that the observed behavior is a result of the interaction between robot and environment.

Recently, as the robots became more and more affordable, the control of multi-robot systems has become a hot research topic [12, 15].

1.2 Multi-robot Systems and Swarm Robotics

Multiple robots can provide, improved performance, distributed sensing, fault tolerance and solutions to problems that are impossible to achieve with a single robot. However, the utilization of multiple robots require coordination among the robots making the development of a control system even more challenging than that of a single robot.

Studies on multi-robot systems included playing team games such as in Robocup [16], as well as studying group behaviors similar to those observed in nature such as foraging [17], forming formations, flocking [18, 19, 20] and cooperative object transport [21].

The development of control systems in the early multi-robot studies were done using extensions of existing architectures. In [20], Matarić extended the subsumption architecture, with internal states and a vector summation to incorporate social interactions. This arhitecture is applied on flocking, foraging and docking. The architecture also allowed learning to be conducted.

Similarly "ALLIANCE" architecture proposed by Parker [18], used multiple controllers with subsumption architecture. The controllers in ALLIANCE architecture competed for control of the robot. Communication between robots, guide the choice of controllers for acting on the real robot. This architecture also considers the case of robots with heterogeneous capabilities. Through communication, robots could specialize on the parts of the problem they are more capable.

Dudek *et. al*, proposes a taxonomy for describing multi robot studies [22] and in [23] Cao *et. al* give a good review of these early approaches, for cooperative multi robot studies. Main drawback of the early research on multi-robot problems is the requirement of global and precise communication usually associated with these architectures. This type of communication leads to error prone systems with poor scalability.

Swarm robotics emerged as a sub-field of multi-robot studies with the recent advances towards the mass manufacturing of small robots. Inspired from social insects, swarm robotic systems consist of large numbers of simple robots that cooperate to solve problems. Scalability and simplicity are central issues in the design of behaviors for robotic swarms and limits the complexity of the robots and their communi-

cation abilities.

Self organization is one of the main mechanisms employed by swarm robotic systems. In [24], Camazine *et. al*, define self organization as follows: *Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among lower level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern*. Camazine *et. al* also , underline the importance of feedback in self organizing systems. It is noted that, in nature, self organizing systems rely on *positive feedback*. Positive feedback is the tendency of the environment and agents to magnify the fluctuations in the system. It is observed in, growth of population, construction of nesting sites, motion of herds and schools and in aggregation. Antagonist of positive feedback is the *negative feedback* mechanism, which acts against the changes in the system. Negative feedback stabilizes the system and reduces the effect of fluctuations. In nature, negative feedback can be a result of limitations in the environment or a part of the behavior of the organisms. Depending on both positive and negative feedback mechanisms, self organizing systems can respond to changes in the environment collectively and return to stable configurations. Although observing natural systems for existence of such mechanisms is useful, it is complicated to build control algorithms that provide these mechanisms for specific tasks.

One of the earliest studies on generation of self organizing behaviors is conducted by Reynolds [25] in 1987. In this study, he demonstrated use of simple rules to generate, flocking behavior in a group of artificial agents. This study also points out that self organization behaviors can be achieved by simple rules. Following this study, two main approaches were used in the literature to achieve self organization. First approach is using optimization methods like evolutionary methods to generate control systems. Problems including aggregation [26], cooperative transport [27], functional assembling [28] and coordinated motion [29] is studied with this method. Stefano Nolfi and Dario Floreano provide many examples of such applications [30].

Second approach to create self organizing behaviors is to use carefully designed algorithms to control agents' behavior. Simple and probabilistic algorithms which are suitable to modeling [31] are employed in this approach. Statistical models of individual behaviors can be used to construct a macroscopic model of the swarm. This model then can be used to predict emergent behavior of the swarm. Macro-

scopic models allow deeper understanding of swarms, faster simulations and design of better individual behaviors. Problems including clustering [17], task allocation and pattern formation [32] is studied with this method. Bahçeci *et. al* [33] gives a review of recent studies on pattern formation.

Patterns formed by individuals in the swarm can provide, increased efficiency and cooperation [20, 32]. Pattern formation usually requires robots to be in close proximity. This can be achieved by *aggregation* behavior. Even though there are studies that show successful aggregation on swarm robotic systems, there is a lack of systematic experiments to explore this behavior.

This thesis presents systematic experiments conducted for understanding aggregation behavior in a swarm of robots. A minimal behavior that uses probabilistic transitions, to arbitrate reactive basic behaviors is implemented. Systematic experiments are conducted on physically embodied simulation of robot swarm. A preliminary version of this thesis is presented in IEEE Swarm Intelligence Symposium, in Pasedena, California, USA [34]. Chapter 2 gives a definition of aggregation behavior and surveys the state of the art for this problem. The simulation environment is described in Chapter 3, followed by the behavior of robots in Chapter 4. Chapter 5 describes the experiments conducted using the simulator environment and finally Chapter 6 includes conclusions and possible future work.

CHAPTER 2

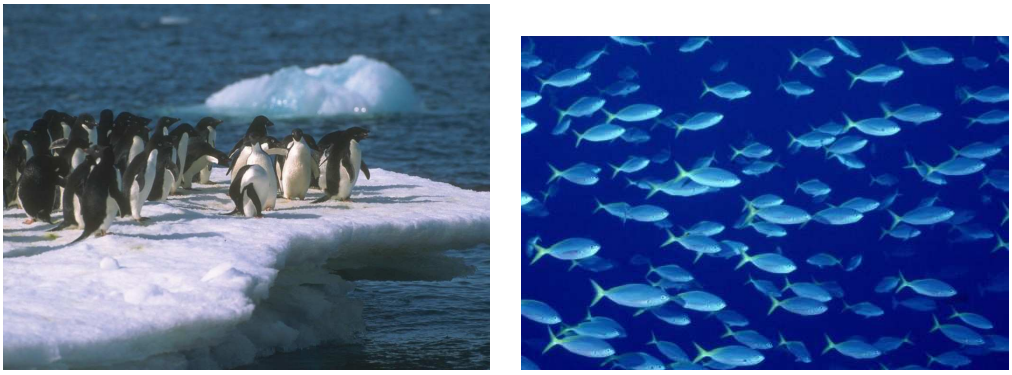
AGGREGATION PROBLEM

Aggregation is one of the fundamental behaviors of swarms in nature and is observed in organisms ranging from bacteria to social insects and mammals [24]. The aggregation behavior in penguins and in fish schools is shown in Figure 2.1. Aggregation helps organisms to avoid predators, resist hostile environmental conditions and find mates. Some of the aggregation behaviors are known to be facilitated by environmental clues; flies use light and temperature, and sow bugs use humidity for aggregation. However, other aggregations are self-organized. Aggregation of cockroaches, young penguins and fish schools don't use such clues but are rather result of emergent cooperative decision.

This study focuses on the self-organized aggregation behaviors for swarm robotic systems [29, 35]. This behavior is required to form a robot cluster, where robots in the cluster is in close proximity, from any distribution of robots. This motivation is shown in Figure 2.2. Aggregation task requires distributed decision to form a globally observable pattern. Even with explicit communication, robots need to know their exact positions and orientations to exactly determine the global view.

Aggregation behavior is essential for swarm robotic systems. Swarm robotics use simple and incapable robots and these robots must cooperate to accomplish tasks. Usually cooperation requires being in some proximity with other robots. Additionally, aggregation is a requisite for swarm robotic behaviors, such as self-assembly and pattern formation.

Aggregation of a robotic swarm is especially challenging since in such systems, individuals have to rely on a rather myopic and crude perception of their world. The perception of robots in these systems is myopic due to the limited range of perception. In addition to this, sensor data is usually noisy and ambiguous, such that only



(a) **(b)**

Figure 2.1: Aggregation behavior in nature. **(a)** Aggregation in penguins **(b)** Aggregation in fish schools. Aggregation provides protection from environmental conditions and from predators for these species.

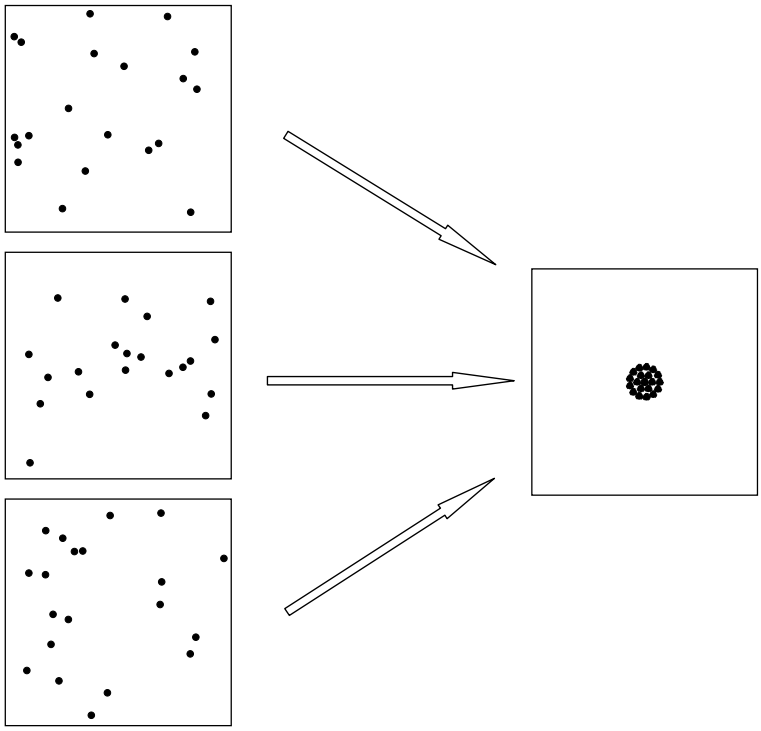


Figure 2.2: The goal of aggregation behavior is to aggregate a number of robots that are initially dispersed.

rough details of the environment can be extracted. Swarm robotic systems, as a principle, should not use global communication and robots must use simple processing to solve problems [36]. As a result, common control algorithms in swarm robotic systems are reactive. Although advances in technology allow cheaper and more effective sensing, communication and computing, simple robots promise more scalable and fault tolerant systems as observed in the nature. Simplistic approach uses less assumptions hence provide a larger field of application. Reliance on global communication and precise sensing creates a fragile system even with improved technology.

In this study, aggregation of a swarm of robots in a bounded arena is investigated. The robots in this study have limited perception range, through ambiguous and noisy sensors.

2.1 Related Work

Aggregation studies in the literature come from different disciplines: biology, robotics and control theory. In this section, these studies will be summarized.

In [37], Deneubourg *et al.* investigate aggregation behavior in weaver ants and cockroaches. Weaver ants aggregate in chains to abridge gaps and cockroaches aggregate together in hiding sites. In these species, individuals rest in aggregations for varying time spans. The study investigates the effect of individual resting time to aggregation. Results indicate that the amount of time individuals spend in aggregations are modulated by environmental conditions and presence of other individuals. Individuals tend to spend more time in large aggregations, providing positive feedback for growth of aggregations. Individuals also spend more time on favorable sites, causing larger aggregates to form on such sites. Using this simple scheme individuals are able to make collective decisions.

Jeanson *et al.*, in [38], present a model of aggregation behavior of cockroach larvae in homogeneous conditions. The aggregation behavior observed in this species, include wall following, and two different resting behaviors. Individual behavior is modeled through systematic experiments. It is observed that the behavior of individual cockroaches depend on the number of cockroaches in close vicinity. A model of this behavior is used in robotic experiments to obtain similar behavior in a group of Alice robots (K-Team, Switzerland).

One of the early robotics applications of aggregation behavior is done by Melhuish *et al.* [39]. In this study, robots are required to form clusters of predetermined size around infrared beacons. Similar to the sounds produced by birds and frogs the method proposed uses chorus consisting of individuals where individuals try to produce sound simulatenously. However, individuals have small variations in elicitation of sound. Using these variations in sound elicitation, individuals can approximate the size of the clusters. This study has also been tested on systems without infrared beacons that trigger aggregation. Results indicate it is only possible to obtain this kind of self organized aggregation behavior, in virtually noiseless environment.

In [26, 29] genetic algorithms are used to evolve neural network controllers for a swarm of robots in aggregation task. Performance of evolved controllers are investigated using rigid body simulation of robots. These studies produced, different aggregation strategies including static clustering where robots stop in clusters, and dynamic clustering where robots continue moving in robot clusters. In spite of the fact that aggregation behavior can be synthesized using this methodology, due to the complexity of analyzing resultant controllers, it is difficult to arrive at general conclusions about aggregation.

In control theory aggregation is often referred as gathering, agreement or rendezvous. These studies usually consider abstract models of robots with varying detail in modeling. Most of the studies [40, 41] in this approach ignore the dynamics of the robots, representing robots as points without orientation. Even though some studies use kinematic models of robots, use of rigid body physics is not common practice.

Another important assumption in these studies is the limit on perception range. Some studies in aggregation consider the infinite visibility case where robots are able to perceive all robots. Under the unlimited visibility assumption, strong results on aggregation such as explicit bounds on the swarm size and bounds on the time of convergence can be obtained [42, 43, 44]. This approach is quite powerful for theoretical analysis however, unlimited perception range is an unrealistic assumption for real robotic swarms.

Real biological swarms have limited range of perception. It is, thus, more realistic to use models with robots that can only perceive a limited range. Studies indicate, when perception range is limited, there are some required conditions for aggrega-

tion. One of these conditions is defined using a visibility graph, which is constructed by robots as nodes and by edges between robots that are able to perceive each other. In order for a deterministic control algorithm to allow aggregation, this graph needs to have at least one node which is accessible from all other nodes in the graph [45]. Flocchini *et. al* noted that, even convergence may not be enough since convergence may take infinitely long time [40]. They proposed a control algorithm that requires only limited visibility with distinguishable robots. Their algorithm guarantees aggregation in finite time and only requires a common orientation decided by robots.

Another study by Gordon *et. al* investigate gathering of agents with asynchronous distributed control [41]. In this study, theoretical convergence is supported with simple kinematic simulation of robots modeled as point bodies. Asynchronous nature of the model introduces a randomness in the behavior of agent clusters. Agent clusters move randomly due to unordered motion of agents. These random motions allow, to a degree, the system to aggregate when necessity conditions are not satisfied.

Importance of strong theoretical support is emphasized in *Swarm engineering approach* by Sanza T. Kazadi [46]. This approach defines the global goals as mathematical constraints. Behaviors are then synthesized to satisfy these constraints. The behavior of system can be investigated using the goal constraints. Lee *et. al* apply this concept to robot aggregation in their recent work [47]. Their implementation of aggregation behavior allows robots to approximately estimate the size of the robot aggregation. Using this information and the analysis of puck clustering problem where similar properties exists, implemented aggregation behavior produces clusters with increasing size.

In all of these studies, aggregation behaviors were analyzed under a rather narrow range of parameter choices. In this thesis, we propose a minimal aggregation behavior obtained through a combination of some simple behaviors. Simple behaviors were combined with subsumption architecture and a finite state machine. By systematically varying the parameters of the minimal aggregation behavior and the environment, we analyzed the aggregation performance of different aggregation strategies using two different metrics. Four representative strategies were chosen using the results of experiments on effect of parameters. These four strategies were then analyzed with respect to simulation time and size of the arena used in experiments.

CHAPTER 3

SIMULATION ENVIRONMENT

In this study a port of the Swarmbot3D [48] simulator is used to conduct our experiments. Swarmbot3D is a physics-based simulator developed within the Swarm-bots project [49]. The simulator provides models of *s-bot*s, mobile robots which have the ability to connect to or disconnect from each other. Swarmbot3D provides models of s-bots with increasing detail. The robots have both long and short range signaling and sensing abilities. The sensing and actuation models used are either taken from sensor data obtained from the actual s-bot or derived from realistic sensor models. The simulator used Vortex(Critical Mass Labs, USA), a commercial physics-based simulation library, to realistically simulate the physical interactions between the robots and the environment. Swarmbot3D simulator is verified against real robots and many studies on swarm robotics is conducted on this simulator, including aggregation studies. Figure 3.1 shows a screen capture from the simulator used in experiments.

There were two main drawbacks of the Swarmbot3D simulator to be used in our study. First, the simulator was slow since it used a physics-based simulation library making it difficult to perform systematic simulations. Second, the simulator used the Vortex, which required license per machine.

In order to be able to conduct systematic experiments in a reasonable time, we ported the Swarmbot3D simulator to the *ODE* (Open Dynamic Engine) physics simulation library, an open source physics engine. This allowed us to run the ported simulator on a computer cluster in parallel, speeding up our experiments.

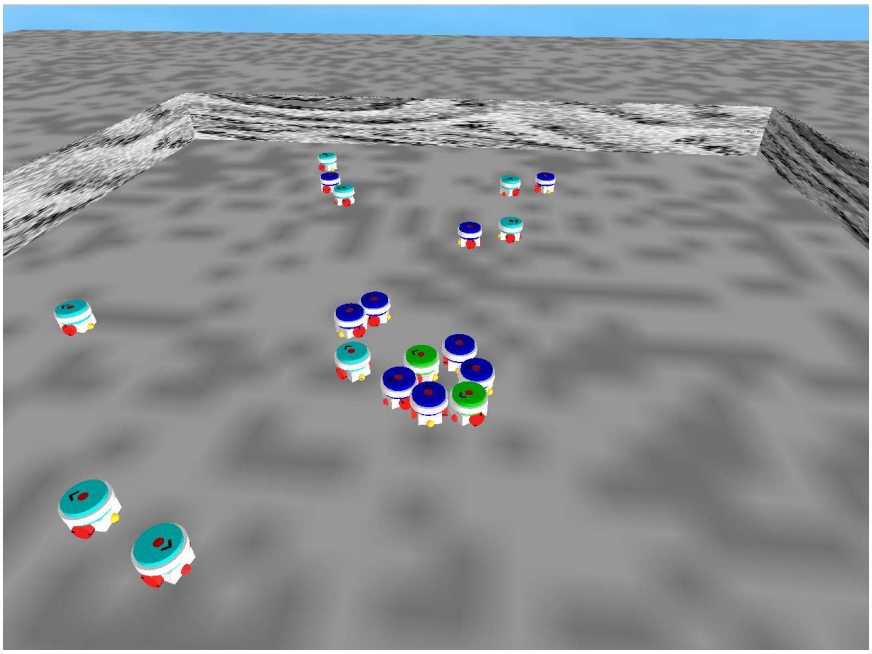


Figure 3.1: A screen shot from simulator.

3.1 Porting Swarmbot3D to ODE

The port of the Swarmbot3D is done through the implementation of a wrapper, called *KODEX* (Kovan ODE eXtensions), to allow this simulator to run using the ODE library.

The work done within KODEX consists of three major parts: First, the Vortex functions that are being used by the Swarmbot3D simulator are implemented using ODE counterparts, where possible. Within the context of this work, only the functions that are used for the implementation of the Swarmbot3D simulator are implemented. Second, ODE is extended to include model loading abilities, which are essential for simulations developed in Vortex but not supported by ODE. This allows one to use the robot, and environment models developed for the Swarmbot3D to be loaded to ODE with minimal changes. Third, a new set of utility functions, such as new and better graphics rendering capabilities are implemented for ODE. Figure 3.2 shows the conceptual representation of KODEX versus Vortex.

Figure 3.3 shows two snapshots of the Swarmbot3D simulator. The first snapshot, shown in Figure 3.3 **(a)**, is captured from the original Swarmbot3D simulator that used the Vortex Library. The second snapshot, shown in Figure 3.3 **(b)**, is captured

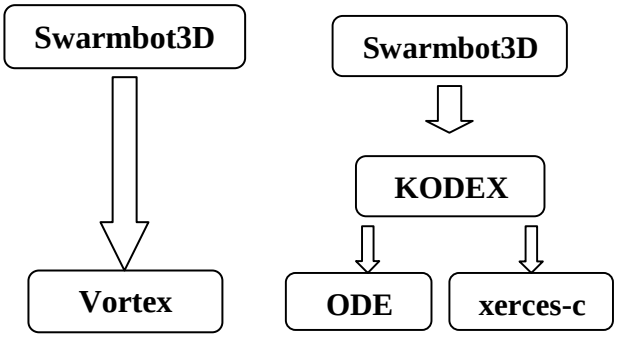


Figure 3.2: Comparison of Swarmbot3D using Vortex and KODEX. **(a)** Swarmbot3D using Vortex. **(b)** Swarmbot3D using KODEX.

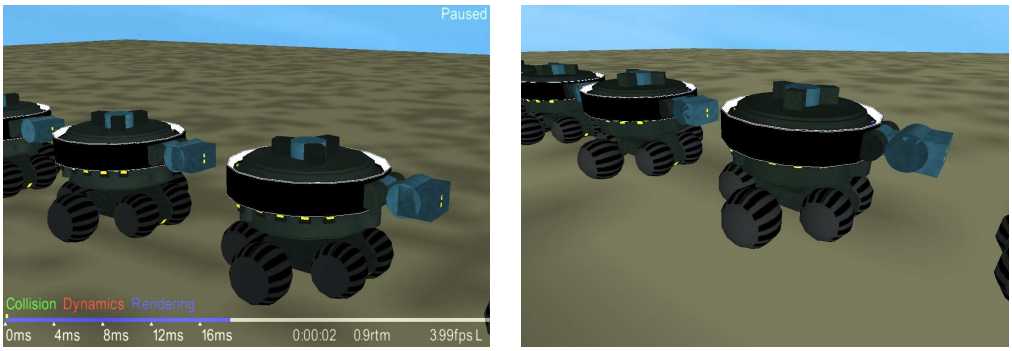


Figure 3.3: **(a)** A scene from Swarmbot3D simulator using Vortex. **(b)** A similar scene with KODEX.

from the ported Swarmbot3D simulator which used ODE as wrapped by KODEX.

In the next subsection, we briefly describe the physics-based simulation.

3.1.1 Physics-based Simulation

Simulation of robots with physical reality requires realistic models of forces, frictions and velocities. Traditional kinematics based models ignore force dynamics of the systems they model. These models are plausible when interactions between bodies are limited.

Physics-based simulation is achieved using, *Rigid body dynamics*, which allows modeling environment and robots as a set of *rigid bodies* connected with *joints*. Rigid bodies are solid objects that don't deform, which can be modeled as a set of particles constrained to move together. Joints are used to connect bodies with different constraints in their motions, like wheels of a car and arms of a clock.

Rigid body dynamics also take into account forces acting on objects, which allows accelerations to be realistically modeled. This way springs, pendulums and many similar devices can be created. More importantly collusions between bodies can be simulated by temporary joints, in this way realistic forces can be exerted on colliding bodies.

In this approach, each body has constant and varying properties. Constant properties include mass, moment of inertia, friction coefficient and varying properties are position, velocity, rotation and angular velocity. Joints, collusions and forces are represented in differential equations defining change in these varying properties. Although these underlying principles for rigid body dynamics is quite simple, for multiple bodies analytical solution is not possible for general case. So numerical approximations are used. Fast, stable and accurate implementation of this numerical approximation is a very challenging task.

Physics System Developer Kits provide numerical integrators to approximate rigid body dynamics. There are various commercial, (VORTEX [50], Novodex [51] and Havok [52]) and open source (ODE [53] and Dyna3d [54]) physics SDK's are available in the market.

Rigid body dynamics is useful in faithful simulation of vehicles, mechanical devices and robots. Computer games, realistic animations as in movies and simulators used in research use rigid body dynamics.

Player-Stage simulator from USC [55], uses a module named Gazebo [56] which models physically embodied robots with rigid body dynamics. Webots [57, 58], which was originally a kinematics based simulator has recently integrated rigid body dynamics using ODE.

In the following section, technical details of KODEX library is given.

3.1.2 KODEX

KODEX is developed in C++ for Linux systems as well as Windows systems using Cygwin.

KODEX provides a very similar interface to Vortex. This allows easy migration from Vortex to KODEX. Many of the applications of Vortex library can be run in KODEX with minimal changes. But it should be noted that KODEX provides only some of the support functionality present in Vortex and is not intended to fully emu-

late the Vortex library.

KODEX improves the functionality of ODE with model loading capabilities. KODEX can load Vortex models and improves ODE with some utility functions and better rendering capabilities.

As a result of this, it is much easier to create physical simulations, than to use plain ODE. With ODE alone, environment and objects must be constructed by writing code. KODEX simplifies this process using XML files, which are parsed to construct the world. This reduces the task of loading a scene to a single line of code. This allows creation of much more complex environments and models easily.

XML files can also be easily extended to add additional user parameters. XML file syntax is also compatible with Vortex further simplifying the migration process.

By allowing textured high performance rendering KODEX improves the visual output of ODE. KODEX also includes better camera and material handling than plain ODE. Rendering can be useful for monitoring the simulations in robotics and camera control is quite important for observing the simulation environment.

KODEX uses two external libraries: `xerces-c` is used for parsing XML files and ODE is used for physics calculations. Rendering is done using a heavily modified version of the reference renderer distributed with ODE. This renderer uses OpenGL and thus it is very portable.

For the rigid body simulation, Vortex and ODE provide similar features, with ODE being slightly more limited. ODE provide bodies which model the mass, inertia and velocity of objects in the environment, geoms which model the colliding parts of bodies and joints which are used to connect rigid bodies together. Although these parts are roughly equivalent to Vortex counterparts, their representations are different.

For instance, position and rotation of bodies are represented in different structures in ODE but they are kept in the same matrix in Vortex. Vortex provides direct access to position and rotation of bodies thus these matrices must be kept up-to-date for compatibility. In KODEX, a map between ODE and Vortex-like versions of these matrices is kept. At each simulation step, all Vortex-like matrices are updated with their associated ODE data.

Another challenging problem encountered during implementation of KODEX is the lack of cylinder implementation in standard ODE implementation. There are

user contributed modules that provide cylinders and in KODEX these modules are included with minor modifications.

Rendering is not an essential part of ODE. Only a reference renderer is provided with the library which is both incapable and inefficient. KODEX uses a heavily modified version of this renderer, with texture and lightning support, improved camera handling and support for additional cameras.

Additionally, KODEX is being used in other projects. In MACS European Union project, KODEX is used to simulate, Kurt-2 robot [59] developed by Fraunhofer AiS. This simulation requires, robot to manipulate objects and complex sensors like camera and laser range turret which are provided by the KODEX library.

The features of KODEX can be summarized as follows:

- *Open source*: This is the major power of KODEX. It can be modified for further needs or customized for specific applications. Since no license is required, KODEX is also very suitable for systematic studies using multiple computers.

- *Platform independent*: KODEX currently works on Linux and Windows platforms with Cygwin.

- *Loading world from XML files*: The parser included in KODEX allows loading of world setup from an easily understandable file. This file format also reduces requirement for rebuilding code for different setups, making it very suitable for a series of experiments.

- *Extensible XML structure*: XML files can contain user defined tags for application specific data. These data can be obtained by the application easily.

- *Easy migration from Vortex code*: Interfaces provided by KODEX are very similar to Vortex interfaces, allowing very easy migration of programs written for Vortex engine to KODEX.

- *Collisions between boxes, cylinders, spheres, triangle meshes and composite objects*: KODEX uses ODE and some user contributed code to provide these collisions. With basic geometries, composite objects and triangle meshes many real world problems can be modeled.

- *Textures and materials in rendering*: Textures and materials allow more visual information to be displayed to user. Examples include but are not limited to emulating LEDs by changing textures on robots.

- *Better camera handling*: Many dynamic simulations require constant debugging and a good view around the simulation environment is a useful tool to understand problems. KODEX uses a flexible motion model for camera to allow better management of the scene.

- *Support for additional cameras*: KODEX provide support for additional cameras that can be used for modeling camera sensors.

3.2 Simulated Robot Model

Simplest model of Swarmbot3D, *fast model* as described in [48], is intended for evolutionary runs and has many successful results [26, 60, 61, 27, 28, 29]. In this study we also used the fast model of simulator. S-bot robot which moves using differential drive, includes proximity and sound sensors. Figure 3.4 shows a schematic drawing of the robot. The large circle (radius: 2.9 cm.) represents the body of the robot oriented upwards and the black rectangles denote the wheels. Gray triangles represent the directional sound sensors. Although in [29] three microphones were shown in the model, these microphones were later extrapolated into four microphones. The dotted lines represent the location and the approximate range of the infrared sensors. The small gray circle at the center of the body indicates the omni-directional sound emitter. Infrared proximity sensors are modeled using sampling data obtained from the real robot with the addition of white noise as described in [48].

3.2.1 Actuator Model

The robot used in the experiments, has two separately controlled wheels mounted on a circular chassis. Motors driving wheels are modeled as motorized hinge joints without rotational limits. These joints allow simple but useful and fairly realistic modeling of motors.

The robots also have omni-directional sound emitters which are constantly kept active during this study.

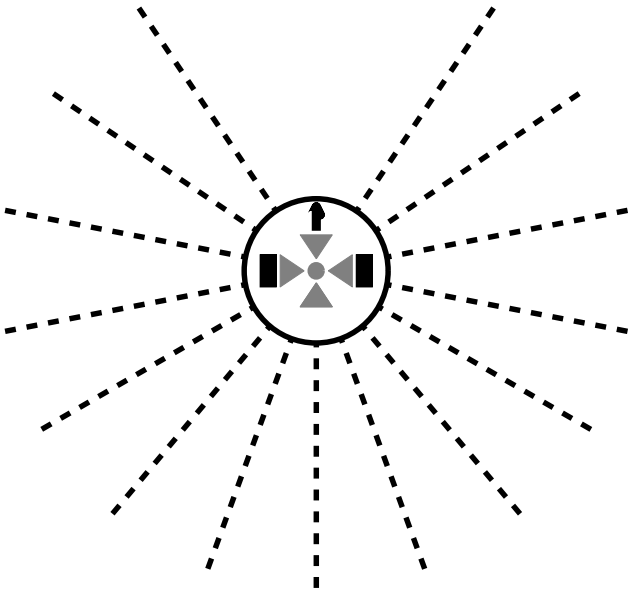


Figure 3.4: A schematic drawing of the robot model. See text for detailed explanation of this figure.

3.2.2 Sensor Models

The robot contains 4 directional sound sensors and 15 IR sensors as shown in Figure 3.4. The IR sensors are modeled using a sampling model described in [35]. The sensors use the maximum of the sensor values computed from robots and white noise with magnitude of 5% of the range of original signal is added to this sensor's readings.

An additional feature extractor algorithm that can detect robots in close proximity is modeled. This can be implemented by using modulation in IR sensing, using a different set of IR sensors.

Sound sensors are not capable of distinguishing different sound sources. In case of multiple sound sources, the result is a sum of the effects of these sources. The echoes and interference are not modeled. Range of sound sensor is 100 cm. Figure 3.5 shows the perceived sound value for a robot. In the x-axis the distance between robot cluster and the perceiving robot changes and y-axis shows the perceived magnitude. Series depict different cluster sizes. In this figure the robots are assumed to be tightly packed and center of mass of cluster is placed directly in front of the sound sensor.

Each simulation step corresponds to 0.1s of real time. Swarmbot3D working with KODEX can perform around 50 frames per second for 20 robots, which corresponds

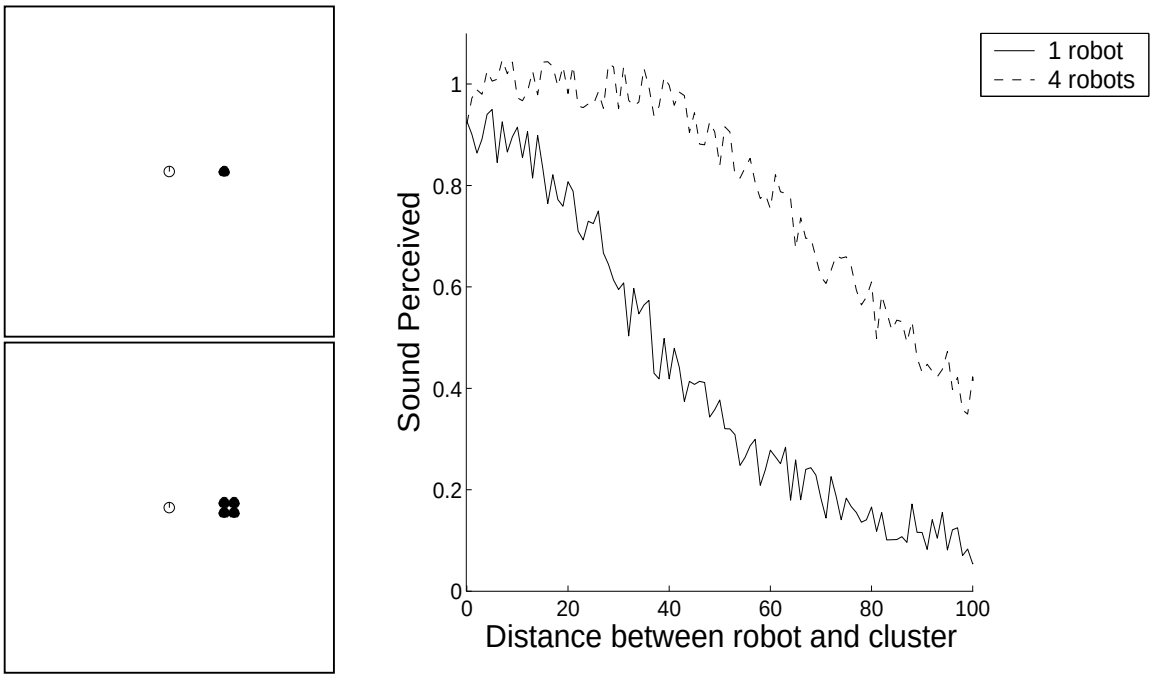


Figure 3.5: Change of sound perceived by a robot with distance and number of robots in the robot cluster

to 5 times speed up with respect to real robots.

It should be noted, however, that this simulator was neither verified against the original Swarmbot3D simulator, nor against the physical robots. Therefore, we make no claims about the portability of the controllers onto the physical robots. Yet, for the purpose of this study, it is our belief that the sensor and signaling models which were taken from the Swarmbot3D simulator are sufficient since this study aims to analyze aggregation behavior in swarm robotic systems in general.

CHAPTER 4

AGGREGATION BEHAVIOR

The aggregation behavior observed in social insects is a result of simple individual behaviors. Computational capabilities of individuals are so low that many of the responses of these organisms are reactive [24].

During aggregation, insects display a combination of reactive behaviors. Three main behaviors can be labeled among these simple behaviors: moving randomly, approaching to an aggregate and waiting in the aggregate [37]. Individuals can help formation of aggregates by approaching to aggregate and waiting in the aggregate. Similarly by moving randomly, aggregates can be dispersed. Due to the limited perception range of the individuals, aggregation observed in nature, requires formation and dispersion of many aggregates. These formation are the intermediate steps of the final aggregate.

Individual decisions for switching behaviors are non deterministic. Although the reason behind this randomness is not clear, external factors are known to be effecting the probabilities for switching behaviors [37, 38].

In this study, we present a minimal aggregation behavior that uses reactive sub behaviors that are switched using probabilistic rules. Next section gives details of the proposed behavior.

4.1 A Minimal Aggregation Behavior

Minimal aggregation behavior used in this study is implemented as a combination of three basic behaviors and obstacle avoidance. These behaviors are arranged in two layers according to the subsumption architecture as shown in Figure 4.1.

In the higher level, three behaviors exist: *approach*, *repel* and *wait*. The *approach* behavior uses sound sensors to estimate the relative direction of the loudest sound

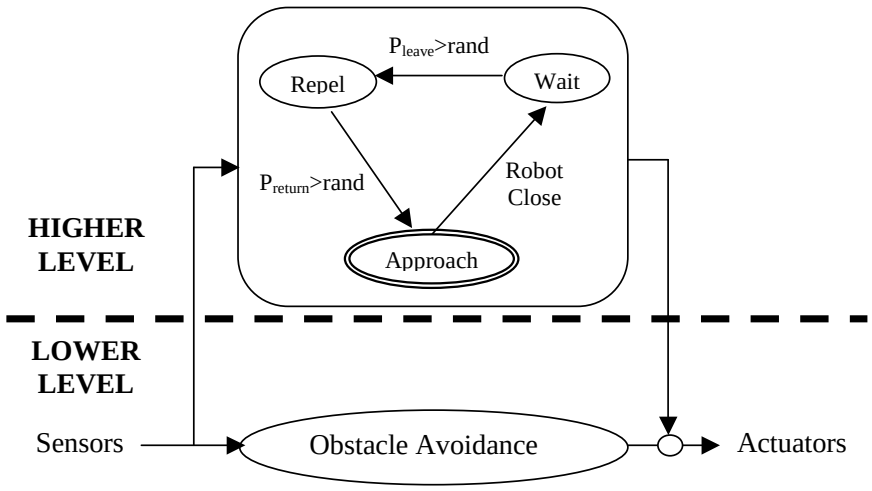


Figure 4.1: Minimal aggregation behavior. Ovals display the behaviors and arrows represent the behavior transitions.

(a rough indication of the closest robot cluster) and drives the robot toward this approximated direction.

The *repel* behavior is the opposite of the *approach behavior*. It drives the robot in the opposite direction of the loudest sound.

During the *wait* behavior, the robot stays in its place.

In the lower level, an *obstacle avoidance* behavior is implemented, which becomes active when the values of infrared sensors become larger than a fixed threshold. This threshold is chosen as 10% of the range of IR sensor values, derived from the investigation of sample data used of IR modeling. Without this behavior robots get stuck on the walls and other aggregates.

The higher-level behaviors are arbitrated using a finite state machine (FSM) with probabilistic transitions as shown in Figure 5.7(a) to implement a class of aggregation behaviors. At each state, the corresponding behavior becomes active. The robot initially starts in the *approach* state. In this state, the robot approaches the largest robot cluster in its view and switches to the *wait* state when it satisfies the "robot close" condition. The "robot close" condition is signaled when a robot can be perceived using infrared sensors. During the *wait* state, the robot picks a random number uniformly within the range $[0 - 1]$ at each time step. If this number is larger that P_{leave}, then the robot switches to the *repel* state. Otherwise, the robot remains in the *wait* state. Similarly, when the robot is in the *repel* state, with probability P_{return} the robot

switches back to the *approach* state.

At the beginning of simulation, all robots are in *approach* state, algorithm shown in Algorithm 1 is run for each robot. *SoundVector* is calculated using sum of the vectors created from sound sensors. Four such vectors are created each with direction same as the relative direction of one of the sound sensors and magnitude equal to the sensor value read from that sensor. Similarly for each proximity sensor, a vector is generated and the *obstacleVector* is the sum of these vectors.

Here *soundThreshold* is introduced to allow robots to deal with noise inherent in sensors and to reduce jaggy behavior. This also allows robots to explore arena when they can not perceive any other robot. *avoidThreshold* on the other hand controls when the obstacle avoidance will suppress other behaviors. Here also note that, in *wait* state, obstacle avoidance is not activated, since without this modification, robots can not wait close to other robots.

Robots use raw sensor values with their relative headings to approximate the relative direction of robot cluster. For this purpose, at each time step, a vector is constructed for each sound sensor, with relative direction same as the heading of the sound sensor and magnitude equal to the value of sensor at that time step. These four vectors are added to approximate direction of the cluster.

The robots try to minimize the angle between desired direction, which is described by either *SoundVector* or *obstacleVector*, and their heading. In order to reduce dead locks and allow smoother movement, robots turn in place until the angle is less than $\frac{\pi}{3}$. When the angle is less than $\frac{\pi}{3}$, robots both move forward and turn toward the desired vector.

It is our belief that the FSM, shown in Figure 4.1, represents a simple model of reactive, self-organizing, aggregation behaviors that are of interest to swarm robotic systems. The resulting behavior can be summarized as: *Approach the sound source. Wait within that cluster for a random time. Run away from sound sources. After random amount of time approach back to the loudest sound source.*

Note that the robot can not distinguish sound sources, causing robot to move to the middle of sound sources when two sound sources with equal intensity are at the same distance. Similarly, from the view point of the robot, a single robot that is close by will sound as loud as a large robot cluster that is further. This phenomenon is illustrated in Figure 4.2, where all setups shown are perceived same by the robot in

Algorithm 1 Aggregation Behavior

currentState ← *approach*
for Each control step **do**
  Read sensor data
  Calculate *soundVector*
  Calculate *obstacleVector*
  if *currentState* = *approach* **then**
    if *mag(obstacleVector)>avoidThreshold* **then**
      move away from *obstacleVector*
    else if *mag(soundVector)>noiseThreshold* **then**
      move toward *soundVector*
    else
      move forward
    if *robotClose* **then**
      currentState ← *wait*
  else if *currentState=repel* **then**
    if *mag(obstacleVector)>avoidThreshold* **then**
      move away from *obstacleVector*
    else if *mag(sound vector)>noiseThreshold* **then**
      move away from *soundVector*
    else
      move forward
    if $rand < P_{return}$ **then**
      currentState ← *approach*
  else if *currentState* = *wait* **then**
    stop motors
    if $rand < P_{leave}$ **then**
      currentState ← *repel*
  Apply actuator outputs

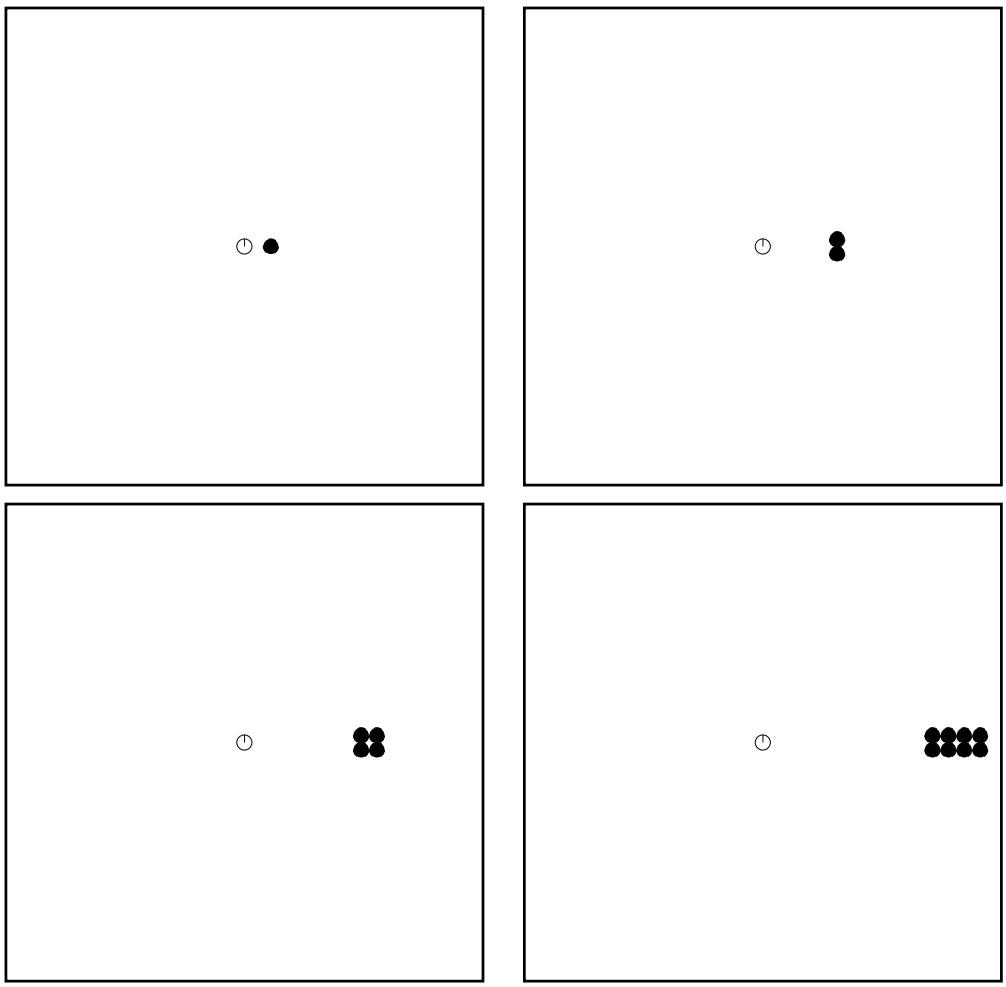


Figure 4.2: Perceptual aliasing in sound sensors. The robot at the center perceived approximately same sound intensities. Therefore robot has no perceptual ability to distinguish between these four different cases.

the middle.

The minimal aggregation behavior described in this chapter is used in the experiments presented in the next chapter.

CHAPTER 5

EXPERIMENTS

In this chapter the experiments done in this study will be presented. These experiments aim to understand the effect of individual behavior on behavior of the swarm, manifestation of aggregation behavior with time and the effect of the size of environment on aggregation.

The minimal aggregation behavior we used in this study has two probabilistic transitions, which govern the overall behavior of the robot. The first set of experiments aims to understand the effect of these probabilities on aggregation. Behavior transition probabilities, as will be revealed in following sections, lead to different and interesting strategies. A set of possible transition probability pairs were chosen for these experiments and runs using these pairs are conducted on the simulator.

Following the investigation of transition probabilities, progress of aggregation behavior is examined using four representative strategies, obtained from first set of experiments. This is followed by experiments to understand the effect of the arena size on aggregation performance, again on the four representative strategies. Increasing the arena size allows investigation of aggregation when visibility constraints mentioned in literature [45, 41, 40] are relaxed.

In order to analyze aggregation behavior a *metric* of the aggregation performance is required. It is difficult to define aggregation performance since aggregation is usually a task dependent and a subjective term. There are different performance metrics proposed in the literature for this purpose and in this study, two performance metrics with different characteristics are proposed. Section 5.1 gives details about the metrics used. This section is followed by, Section 5.2, which gives details of the experimental setup. Section 5.3 presents experiments made on behavior transition probabilities. Finally, Section 5.4 and Section 5.5 report experiments on time dynamics and arena

size.

5.1 Aggregation Performance Metrics

Aggregation performance metrics are required to compare the performance of different aggregation behaviors. The difficulty of constructing aggregation metrics arises from the subjective and task depended nature of aggregation. Good aggregation for one task may not be as good for another task.

In literature various metrics were used for evaluating aggregation. Jeason *et. al* used ratio of largest cluster formed to the number of robots in [38]. Trianni *et. al* on the other hand used the sum of distances of robots to the center of mass of the robots, with some exceptions and normalization[26]. In [47], the aggregation behavior involves decreasing number of active robots. The number of active robots decrease with increasing cluster size This property allows them to measure performance of aggregation using the reduction in number of active robots.

Considering the variety in requirements we introduce two different aggregation performance metrics. These metrics will be explained in detail in the following subsections.

5.1.1 Expected Cluster Size

Expected Cluster Size (ECS) metric aims to estimate the size of the cluster any robot belongs to after the aggregation algorithm is run on the swarm. Calculation of this metric involves counting the number of clusters, so the clusters must be identified mathematically.

In the implementation of ECS metric, a threshold $T_{RobotClose}$ is used to determine robots neighbor robots. The robots which are closer than the $T_{RobotClose}$ are named neighbors, which can be labeled to be in the same cluster immediately. Transitivity of this neighborhood relation is used to determine clusters.

Let $dist(R_i, R_j)$ denote distance of i^{th} to j^{th} robot. Then neighborhood relationship is defined as follows:

$$Neigh(R_i, R_j) = \begin{cases} 1 & ;\text{if } dist(R_i, R_j) < T_{RobotClose} \\ 0 & ;\text{o.w.} \end{cases} \tag{5.1}$$

Using this neighborhood information, robots in the same cluster are determined using a connected component determination algorithm. This algorithm constructs the transitive closure of the neighborhood matrix. Using the transitive closure of neighborhood matrix, each connected component is labeled as a separate cluster.

Cluster size for a robot, $Size(R_i)$, is the number of robots in the cluster that robot belongs to. This metric calculates the average of cluster sizes for each robot in the swarm, where n is the number of robots in the swarm.

$$ECS = \frac{1}{n}\sum_{i=1}^{n} Size(R_i) \tag{5.2}$$

This metric ignores spatial distribution of clusters, but gives a measure for size of cluster each robot belongs to. This approach is useful for applications where robots must maintain local links with other robots in a cluster. Average size of robot clusters is also important in applications where specific number of robots are required such as in many self-assembly and pattern formation tasks. In [38], Jeanson *et al.* used a similar metric for measuring the aggregation performance. Their work only uses the size of largest cluster, ignoring the distribution of the robots outside the largest cluster. In contrast ECS metric, by taking average of cluster sizes, takes this point into account.

In this study, $T_{RobotClose}$ is set to $10\ cm$ which is approximately the distance from which robots can be reliably detected with the infrared proximity sensors.

5.1.2 Total Distance

The second metric, called *Total Distance* (TD), measures the total of distances between each robot pair. This metric gives more information about the spatial distribution of the swarm and clusters. This metric uses negative of distance to emphasize high metric value for better clustering. This metric is defined as follows:

$$TD = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} dist(R_i, R_j) \tag{5.3}$$

This metric is useful when density of robots in an area is significant. However, when local communications are required this metric doesn't always correspond to locally connected groups of robots. This metric is essentially equivalent to the metric

used by Trianni *et al.*[26], but further penalizes outlier robots and doesn't include normalization.

5.2 Experimental Setup

All experiments use a bounded square arena similar to the one shown in Figure 5.1. Bounded arena assumption is introduced to simplify the problem to a manageable size. Without borders, either robots must have memory or the visibility graph of robots must be connected. Therefore we chose to keep borders, as in many studies in literature[39, 38, 26]. Although the use of square arenas is not common, we didn't have sampling data for IR sensors against concave surfaces, hence we kept the arena shape square. The default size of this square arena is chosen to be large enough to allow visibility problems to arise, and small enough to be solvable by simple algorithms in some cases. Size of the standard arena is $200\ cm\ \times\ 200\ cm$, this arena is shown in Figure 5.1.

Experiments are conducted in the simulator environment using 20 robots. Here the number of robots is chosen to allow more than one stable group of robots. With more than seven robots, the geometry of the robots, allows fairly stable clusters. By using 20 robots more than one such large stable cluster can be formed.

Fifty runs are made for each data point in the figures shown with different random initial placements. All experiments use the same fifty random initial placements. Robots are able to move at a maximum speed of $0.3cm$ per simulation step. At this simulation speed robots can cover, in the standard run time of 160000 simulation steps, a maximum distance of $48000cm$, 240 times the length of the standard arena. This procedure is summarized in Figure 5.2.

Performance metrics are evaluated for the positions of robots at the end of each run. A total of 2200 runs, for 160000 steps and 200 runs for 320000 steps are conducted, for the results presented. Although KODEX provides a descent speed up, this amount of simulation is still costly. A single run of the simulation for the 160000 steps requires approximately an hour to complete on a 2.4Ghz Pentium4 computer. 2200 such runs are reported in this document, which would require more than 90 days on a single computer. Without the license restrictions of Vortex, KODEX can be run in parallel. We used the 128 node Beowulf cluster in TÜBİTAK ULAKBİM

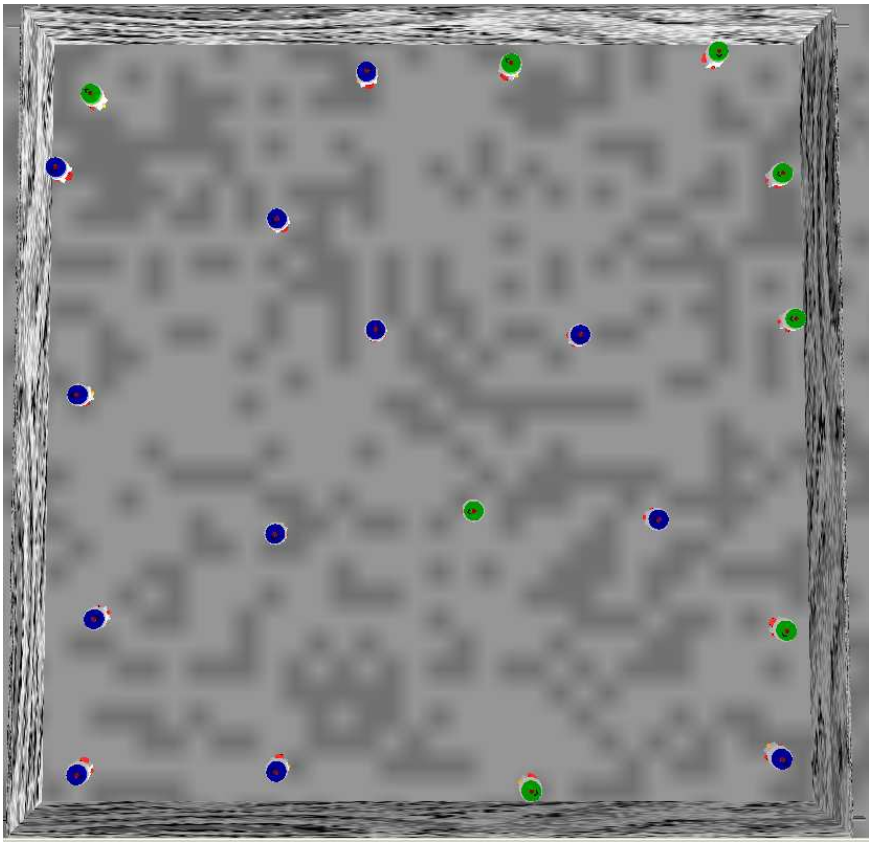


Figure 5.1: A screenshot of the standard arena used in experiments.

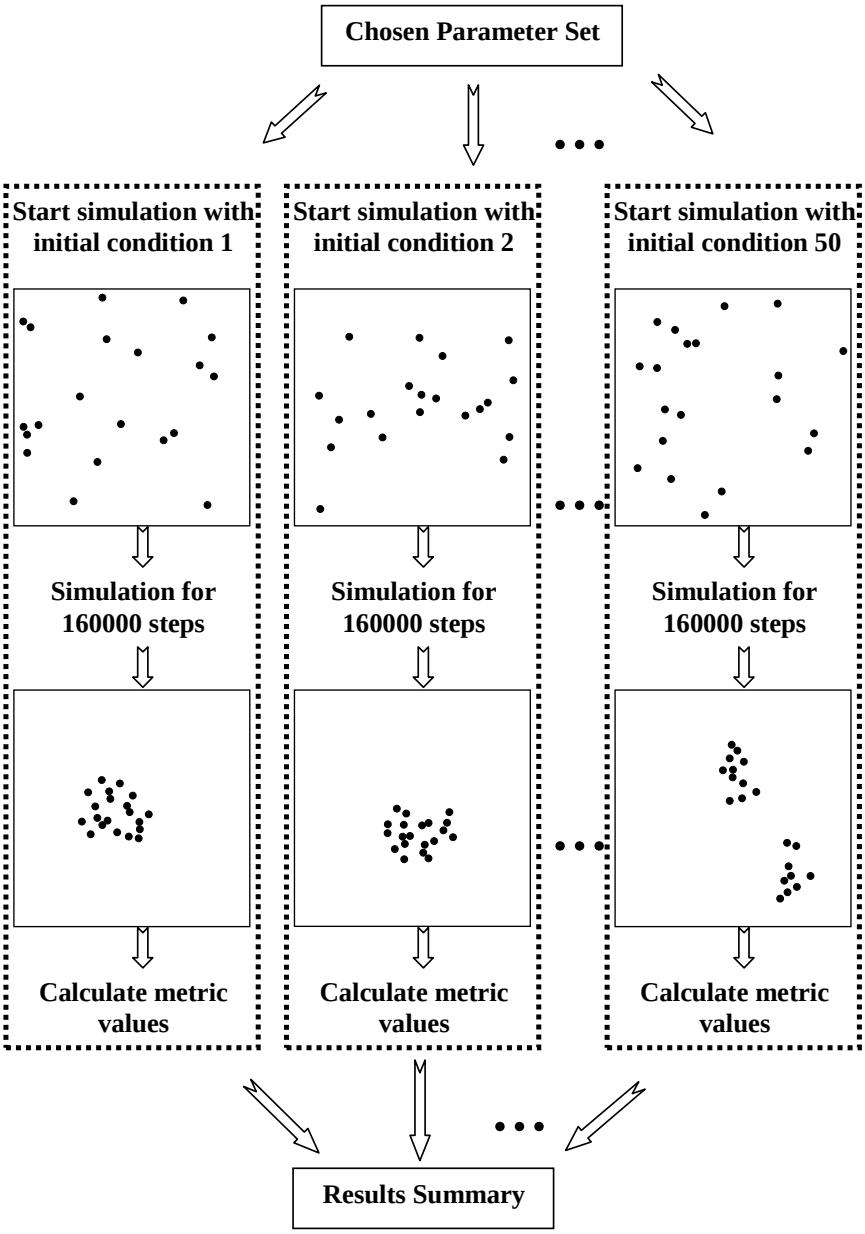


Figure 5.2: This figure shows the flow of experiments. Each dotted box represents an independent simulation.

High Performance Computing Center for this purpose, reducing the computation time to under a single day, provided all machines were dedicated to this task, which is almost never the case.

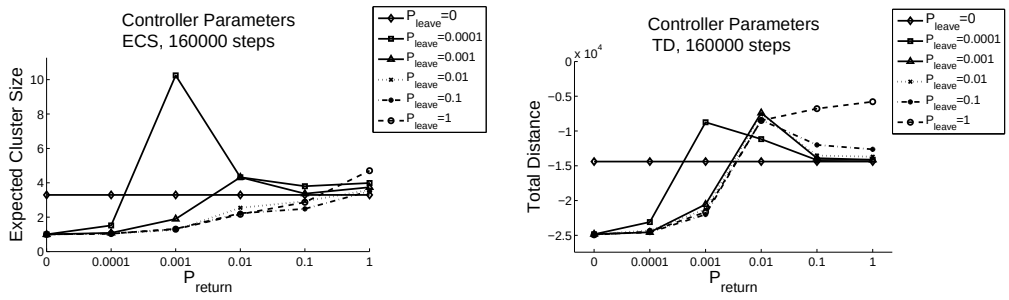
Through the work, we collaborated with TÜBİTAK ULAKBİM High Performance Computing Center, through beta testing. Initial experiments were conducted using *Parallel Virtual Machine* (PVM)[62] library utilities implemented. With the installation of *Sun Grid Engine* (SGE)[63], all simulation runs are ported to be executed from SGE array jobs. SGE array jobs are intended for repetitive tasks similar to the task described in this study. Scripts implemented for SGE array jobs are also provided to other users of ULAKBİM High Performance Computing Center.

In the following sections, the experiments conducted using this framework is described.

5.3 Effect of Behavior Transition Probabilities

This part of the experiments aims to understand the effect of behavior transition probabilities P_{leave} and P_{return} to aggregation performance. These probabilities change the amount of time robots spend in behavior states. The experiments reported in this section aim to understand these effects. Figure 5.3 summarizes the effect of different P_{leave} and P_{return} probabilities in the standard arena. In this figure median performance for 50 runs of each pair of behavior transition probabilities is plotted. P_{return} values change on $x - axis$ and different P_{leave} values are on different series. In Figure 5.4 and Figure 5.5 the aggregation performance is plotted in detail for ECS and TD metrics respectively. Each sub-figure in these plots represent a different P_{leave} value and again P_{return} changes on $x - axis$. In these figures, boxes represent the interquartile interval and whiskers represent the minimum and maximum values found for ECS and TD metrics.

One of the factors for high variance of the results can be explained by the fact that the aggregation algorithm is being observed at different stages. Although 160000 is sufficient for some of the initial conditions to form large clusters, it is not sufficient for all initial conditions. When performance grows larger, the range of different stages of aggregation also grows larger. Figure 5.6 shows histogram of ECS metric values for aggregation behaviors. In this figure, each subplot is the histogram of ECS



(a) **(b)**

Figure 5.3: Effect of behavior transition probabilities. (a) On ECS metric values (b) On TD metric values.

performance values for the 50 runs conducted for corresponding behavior transition probabilities. In this figure P_{return} changes on horizontally and P_{leave} changes vertically. In each histogram, ECS value increases from left to right and bars indicate the number of occurrences of the corresponding ECS value.

In order to remedy this problem, the number of runs is increased and results has shown no significant improvement in variance. Median values in this case seems quite similar. This result points out that the number of runs is sufficient and the variance is inherent to the system.

Another important factor in the high variance, is the instantenous calculation of performance. Robots in this study are usually in motion, which introduces additional variance to the aggregation performance of group.

By choosing different values for controller parameters, we constructed 4 strategies with different characteristics. Behavior of these four strategies can be seen from screenshots of the arena for varying simulation steps in Figure 5.8. Details of these strategies are explained in the following subsections.

5.3.1 Strategy 1

When $P_{leave} \neq 0, P_{return} \neq 0$, the behavior has two competing dynamics determined by the transition probabilities. Increasing P_{leave} increases the time spent by robots in repel state, thus reduces size of the robot clusters. But this also allows more robots to search for clusters, in turn increasing the chance of forming larger clusters.

Increasing P_{return} on the other hand reduces the time robots spend in repel state. By this way, distance traveled by robot while changing clusters is also reduced. When

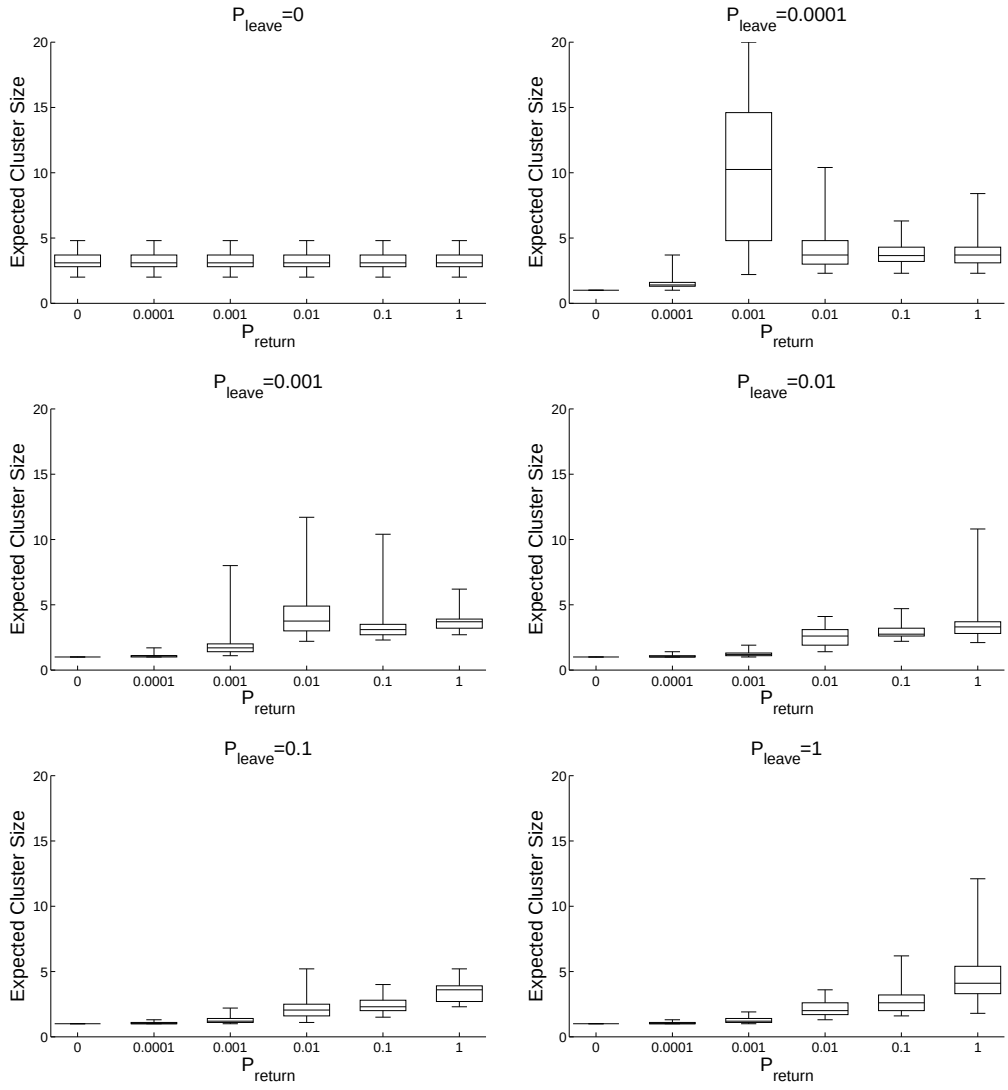


Figure 5.4: Change of ECS metric values for varying P_{leave} and P_{return} for standard arena for 50 runs. Each figure shows a different P_{leave} value. P_{return} changes along x axis. Boxes in the middle are interquartile interval. The line in middle of boxes are the median, and whiskers are the min and max values for ECS metric.

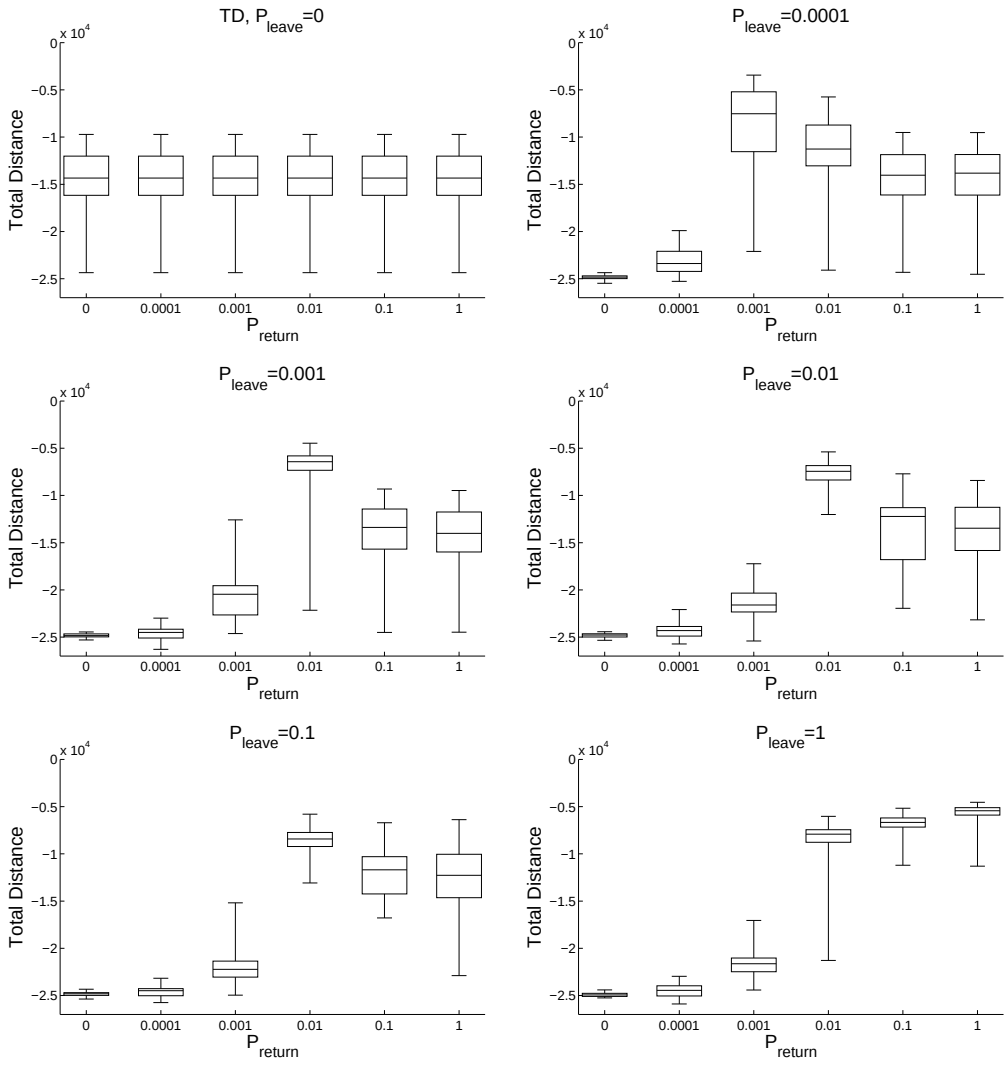


Figure 5.5: Change of TD metric values for varying P_{leave} and P_{return} for standard arena for 50 runs. Each figure shows a different P_{leave} value. P_{return} changes along x axis. Boxes in the middle are interquartile interval. The line in middle of boxes are the median, and whiskers are the min and max values for TD metric.

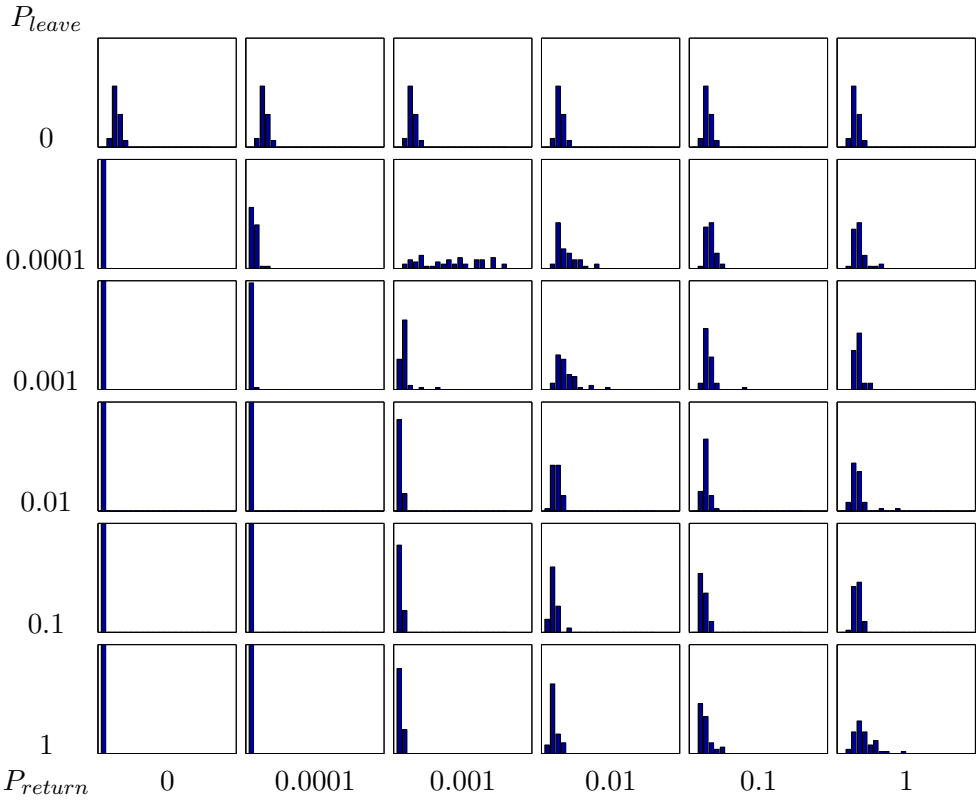


Figure 5.6: Histograms representing the distribution of ECS metric for different behavior transition probabilities.

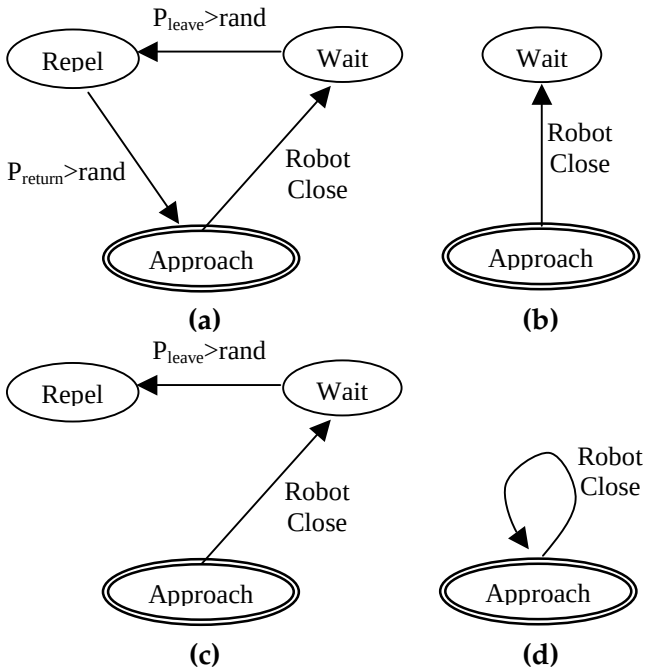


Figure 5.7: Finite State Machine representations of the representative strategies. (a) strategy 1, (b) strategy 2, (c) strategy 3, (d) strategy 4. See text for detailed description.

robots can not move far away from robot clusters, more stable clusters are obtained.

To represent this strategy, P_{leave} is chosen to be 0.0001 and P_{return} is chosen to be 0.001. Aggregation behavior using these probabilities is the highest performing value discovered in this study for the ECS metric and performs reasonably well with the TD metric. Figure 5.7(a) shows the FSM corresponding to this controller.

5.3.2 Strategy 2

When $P_{leave} = 0$, the FSM of the resulting behavior is reduced to Figure 5.7(b). In this case robots remain in *wait* state. This behavior can be summarized as: *Move toward sound sources and when close to a robot, stay there forever.*

In this strategy, when robot gets near another robot, the robot changes into *wait* state. Since the probability of switching to *repel* state is zero, the robot stays within the *wait* state forever. This strategy is deterministic, in the sense that robots never use probabilistic transitions of the behavior. This behavior leads to small "frozen" aggregations. Where robot clusters converge to locally formed clusters and stay in these clusters. It is also interesting to note that, the result of this strategy is only dependent on the initial distribution of the robots, since no random process is involved.

Due to the fact that robots being stuck in *wait* state, the expected cluster size for this strategy is independent of P_{return} and is considered as our baseline performance.

5.3.3 Strategy 3

When $P_{leave} \neq 0, P_{return} = 0$, the FSM of the resulting behavior is reduced to Figure 5.7(c). In this strategy, similar to Strategy 2, robots are stuck in a state: *repel*. Since this *repel* state causes robots to move away from sound sources, this behavior is really poor in aggregation performance. The behavior can be summarized as: *Move toward sound sources initially but then run away from sound sources forever.*

In this strategy, eventually, all robots are in the *repel* state, which means they are actively avoiding sound sources. This destroys all aggregations and leads to the worst aggregation performance.

An interesting point to note about this strategy is that, this behavior can accomplish another task that can be of interest to swarm robotic systems, *segregation*. In *segregation* task, robots are required to be as far as possible from each other, while

maintaining visibility. Robots using this strategy can cover an area efficiently, similar to gas molecules, in a closed box.

5.3.4 Strategy 4

When $P_{leave} = P_{return} = 1$, the FSM of the resulting behavior is reduced to Figure 5.7(d). In this behavior, robots doesn't utilize the full potential of aggregation behavior. This time robots are stuck in *approach* behavior.

The strategy can thus be summarized as: *Move toward sound sources but don't stop; avoid robots like walls.* In this strategy robot never remains in states other than *approach*, since the probabilities of transition are one. This is equivalent to a fully reactive control with only *approach* and *obstacle avoidance*.

This behavior creates dynamic robot aggregations, exploiting the *obstacle avoidance* behavior. Robots approaching each other asynchronously and randomly cause a random drift in the robot clusters. When robots get too close, they try to avoid each other using *obstacle avoidance* behavior, moving in unpredictable directions. Robots are in constant motion in this strategy, as a result, the clusters they form are also in motion.

Although there is only a minor difference between this behavior and the behavior with $P_{leave} = 0.1$ and $P_{return} = 1$; the emergent behavior is quite different. In the latter case, robots use *wait* behavior and can't benefit from drift effect caused by obstacle avoidance in strategy 4.

In this section we introduced the results of experiments conducted on thirty six different behavior transition probabilities. Performance of chosen behavior transition probabilities were investigated using systematic simulation runs. Among these cases, four distinct strategies were qualitatively described. We should underline that these four different strategies are obtained by changing only the transition probabilities of robots, allowing static aggregation, dynamic aggregation and even segregation. The aggregation behaviors obtained perform differently according to the ECS and TD metric. When wait behavior is employed, robots move less amount of distance, and these kind of strategies perform better according to the ECS metric. When wait behavior is not used effectively, robots move continuously and these kind of strategies perform better in TD metric.

In the next sections we further investigate these four strategies.

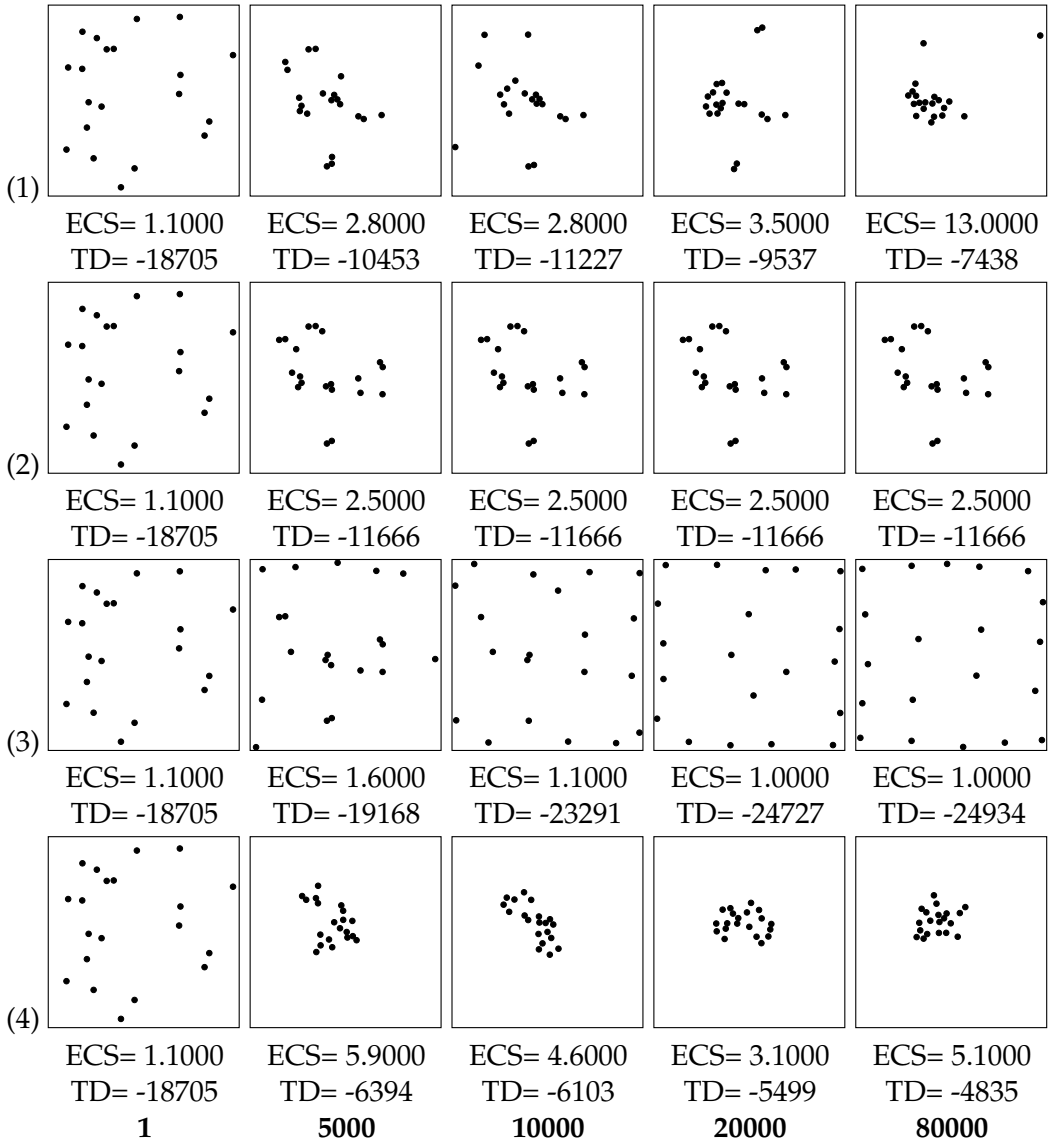
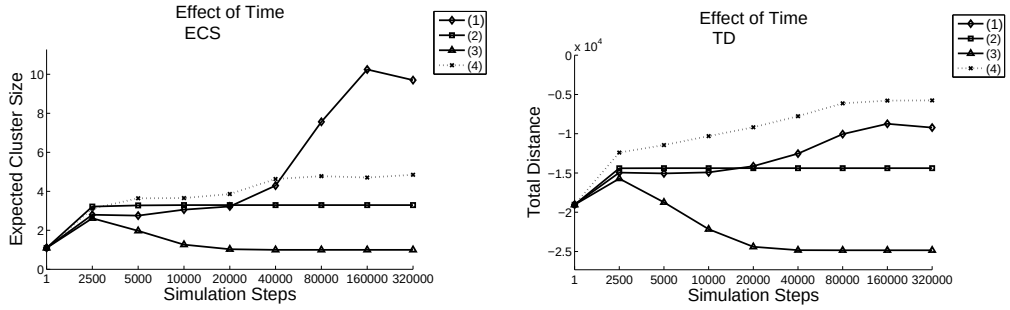


Figure 5.8: Positions of the robots in the arena for different time steps for four strategies.



(a) **(b)**

Figure 5.9: The effect of time on metric values for the four strategies. (a) shows median of metric values for ECS metric, (b) shows median of metric values for TD metric.

5.4 Effect of Time

Robots must move to form clusters in aggregation behavior. This motion requires time. Moreover, the probabilistic controller used in this study includes *wait* behavior, which increases time required for aggregation. This part of experiments aims to observe phases of aggregation with respect to time.

Figure 5.9 gives the summary of change in performance of strategies with respect to number of simulation steps conducted for both, ECS and TD metrics. In this figures, number of simulation steps conducted change in $x - axis$ and performance metric values change on $y - axis$. This figure shows the median values of performance. The results are given in more detail in Figures 5.10 and 5.11. Where change in performance of the strategies are separated in different plots. In these figures, each box represent the interquartile interval and whiskers show the minimum and maximum value.

It can be seen that the ECS performance of strategy 1 is even lower than strategy 2 and 4 during the first 20000 steps. But after 40000 steps, performance of strategy 1 increases rapidly. Another interesting point is the initial increase of performance for strategy 3. All robots start with approach behavior thus they form initial clusters corresponding to increasing performance until 2500 simulation steps. After a random amount of time controlled with P_{leave} robots start moving away from sound sources. In case of strategy 3, robots never change into the *approach* behavior. Which causes the performance of this strategy to fall afterward.

According to the results of the experiments with the TD metric, 4th strategy is su-

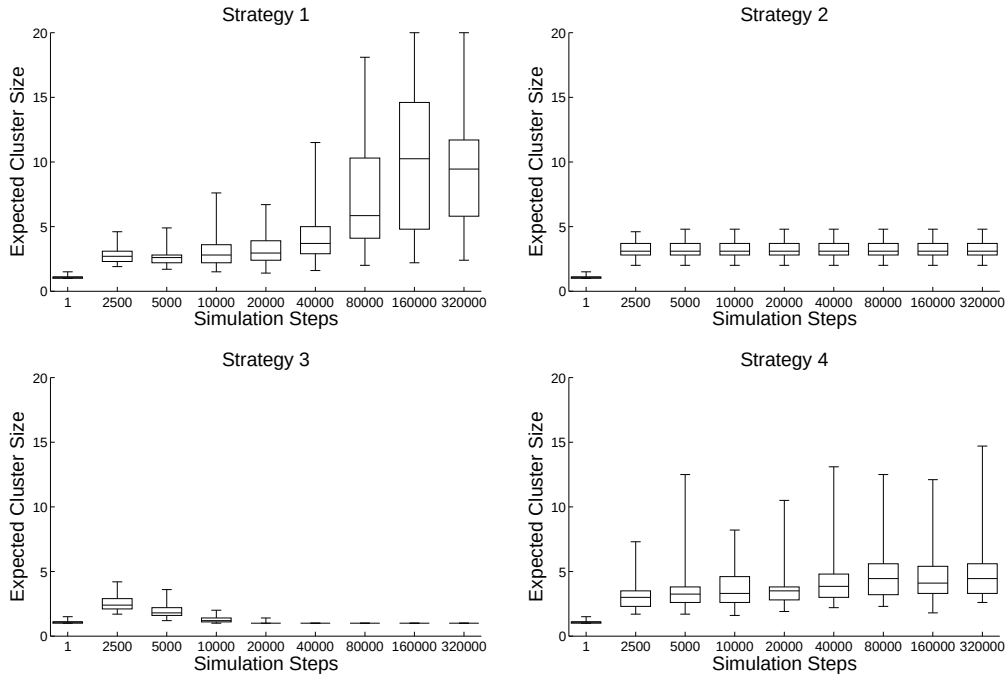


Figure 5.10: Effect of time on ECS metric values for the four strategies. Boxes in the middle are interquartile interval. The line in middle of boxes are the median, and whiskers are the minimum and maximum values for ECS metric.

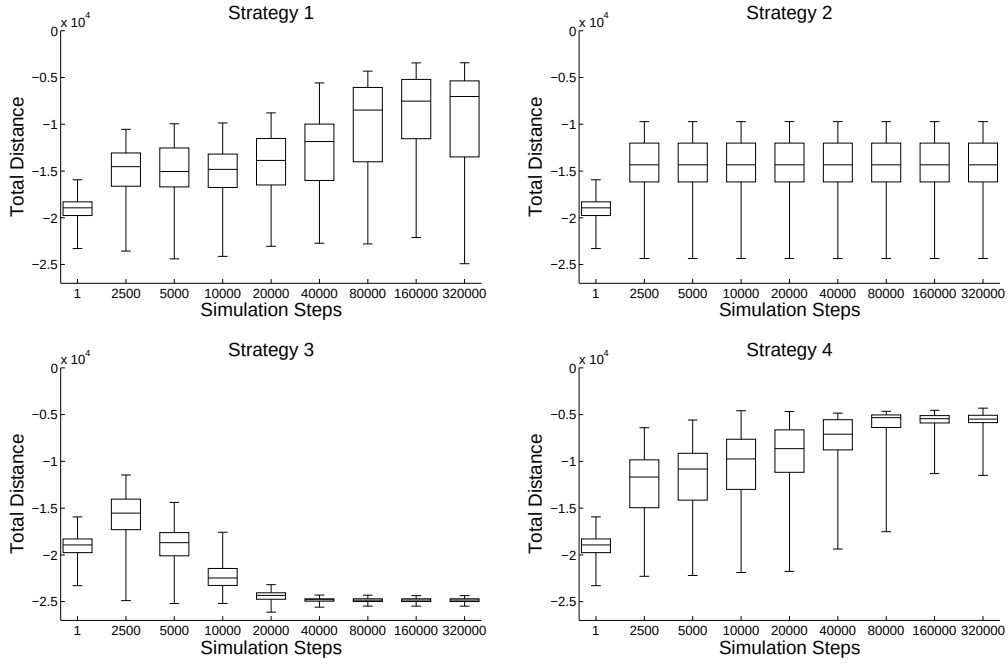


Figure 5.11: Effect of time on TD metric values for the four strategies. Boxes in the middle are interquartile interval. The line in middle of boxes are the median, and whiskers are the min and max values for TD metric.

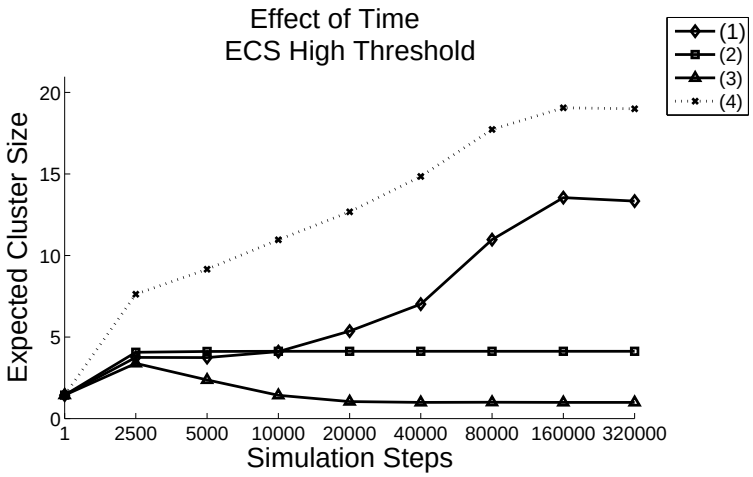


Figure 5.12: Effect of time on ECS for the four strategies with $T_{RobotClose}$ is equal to $13\ cm$.

perior. This effect can be explained by the definition of aggregation for the ECS metric. The threshold used for deciding on neighboring robots was $10\ cm$. This threshold is found to be high enough to detect clusters in strategy 1, but is not sufficient to label clusters in strategy 4 where clusters are more sparse internally. Figure 5.12 shows median cluster sizes for a larger threshold (maximum distance possible for IR detection range, $13\ cm$) for the four strategies. ECS metric values are calculated in the same way except the threshold is chosen to be higher. Similar to the other results, reported in this study, detailed results for the four strategies are separately plotted on Figure 5.13.

Although the performance of strategy 4 is high, it is not very feasible for robotic systems. Apart from the risk of having large number of robots moving in close proximity, large energy consumption due to motion is problematic. In this strategy, robots never cease to move, therefore they use more energy. Figure 5.14 plots total distance traveled by all robots in the swarm for different strategies. In strategy 2, robots move only a little before coming to full stop and in strategy 3, robots move the largest distances among all strategies. There is also a significant difference between distances traveled by robots using strategy 1 and strategy 4.

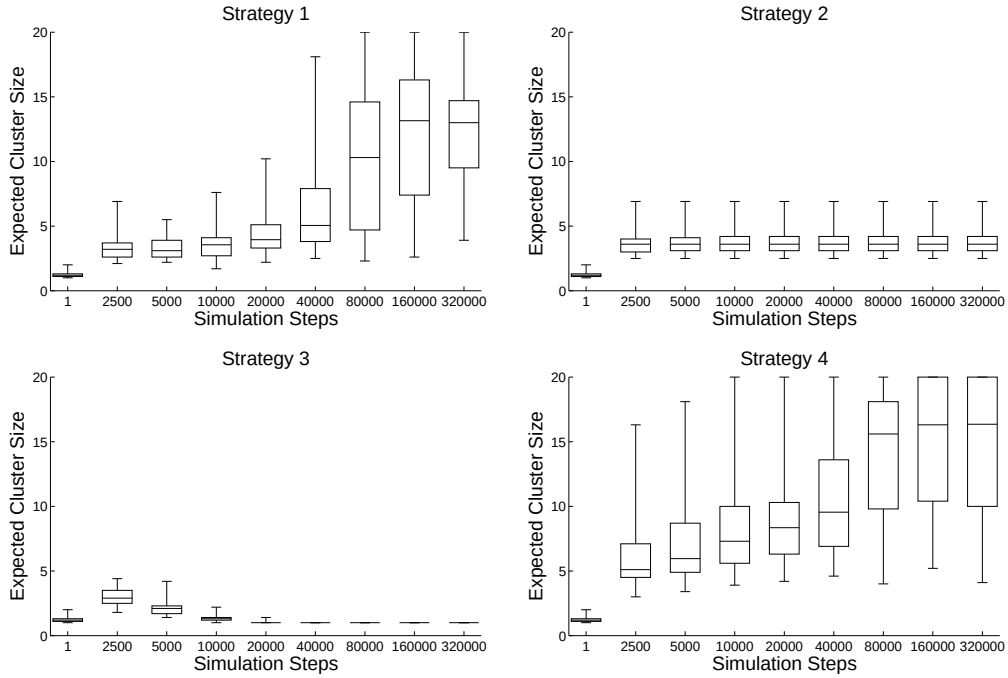


Figure 5.13: Effect of time on ECS for the four strategies with $T_{RobotClose}$ is equal to $13\ cm$. Boxes in the middle are interquartile interval. The line in middle of boxes are the median, and whiskers are the min and max values for ECS metric.

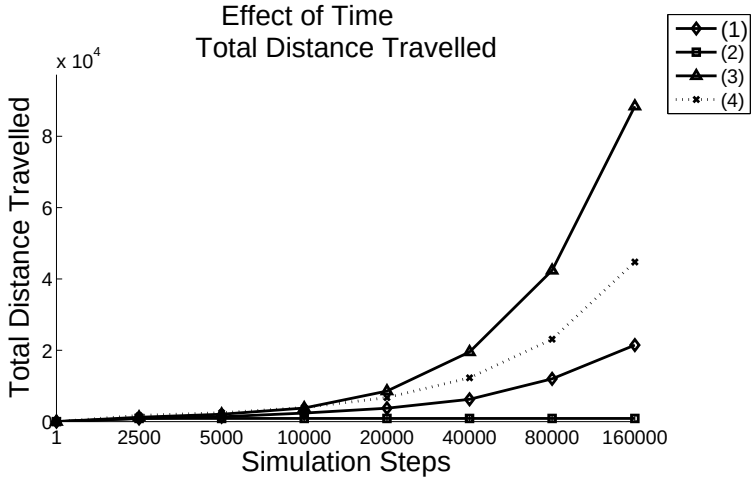
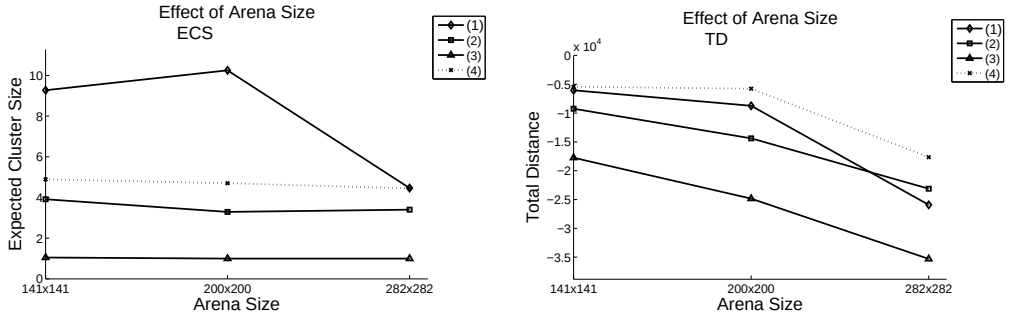


Figure 5.14: Total distance traveled by robots with respect to time for the four strategies.

5.5 Effect of Arena Size

Arena size in this study, controls the visibility of robots to each other. In standard arena size of 200 cm $\times$ 200 cm, a robot in the middle can perceive most of the arena. By changing the size of arena, relative size of the perceived area is changed. Arena size also effects the distance robots must traverse while forming robot clusters.



(a) **(b)**

Figure 5.15: Effect of arena size on ECS for the four strategies. (a) shows median of metric values for ECS metric, (b) shows median of metric values for TD metric

Figures 5.16 and 5.17 display the performance of the four different strategies in different arena sizes. The results show that even when the transition probabilities are chosen for the arena with size 200 cm $\times$ 200 cm, the behavior can still outperform the other strategies in the smaller arena and perform reasonably well for the larger arena.

Larger arena poses a rather interesting challenge for aggregation. With this arena size, the visibility graph of robots is usually disconnected. In this case, deterministic algorithms are bound to fail. Gordon *et. al* [41], note that additional random behavior can lead to better clustering when visibility graph is disconnected. Small improvement in performance supports this argument.

In small arena, without considering the perceptual aliasing effect, theoretical limit for performance is complete clustering. This is not observed in our experiments. It should be noted that, the sound sensor model used in this study is much simpler than the long range sensor models used in visibility graph discussions[40, 45, 41]. These discussions assume sensors to be able to distinguish sources or at least cluster directions. So although there exists a more general statement about limits of performance in large arena, these studies are not directly applicable to small arena in this

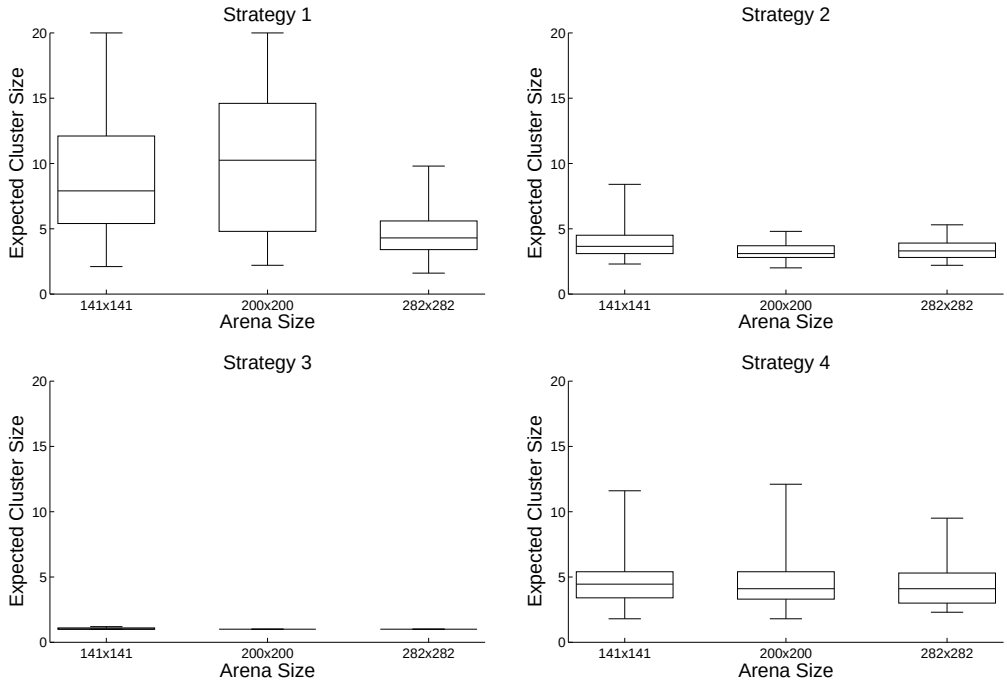


Figure 5.16: Effect of arena size on ECS metric values for the four strategies. Median of performance of 50 runs with respect to arena size for 160000 simulation steps are shown together with interquartile interval as whiskers.

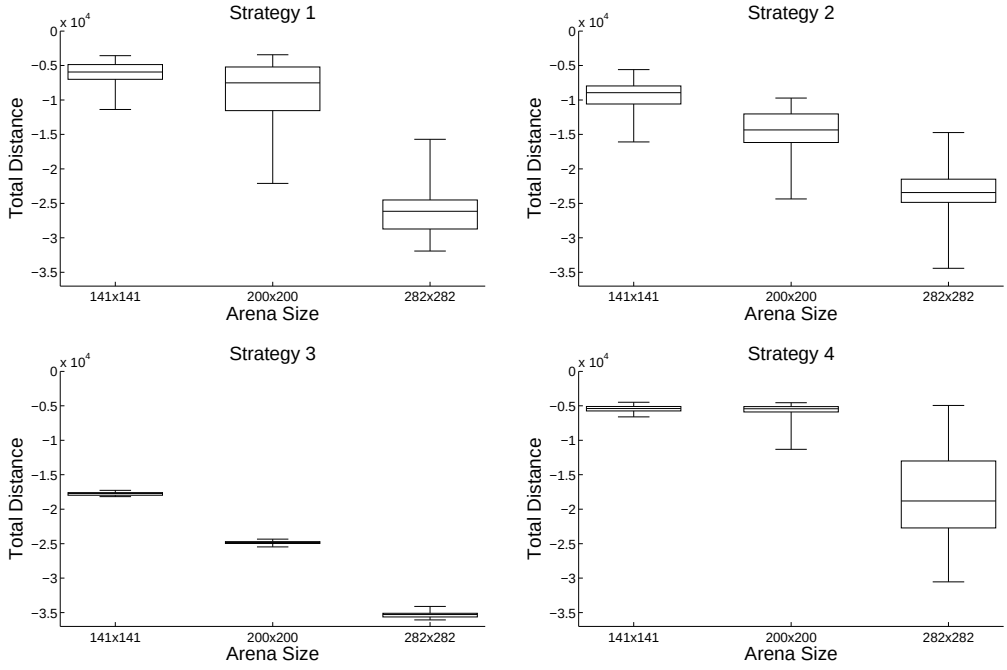


Figure 5.17: Effect of arena size on TD metric values for the four strategies. Median of performance of 50 runs with respect to arena size for 160000 simulation steps are shown together with interquartile interval as whiskers.

study.

CHAPTER 6

CONCLUSION

A minimal aggregation behavior composed of simple reactive behaviors is presented in this study. Parameters of this behavior and some parameters of the environment has been analyzed using two performance metrics. To the best of our knowledge, this kind of systematic study is the first in literature for aggregation in swarm robotic systems.

Strategies obtained through variation of parameters provide segregation like behavior (strategy 3), static clustering (strategy 2), and dynamic clustering (strategies 1 and 4). Clustering metrics favor different dynamic clustering methods, indicating importance of an appropriate metric for the required task. Furthermore, threshold choice for the ECS metric is important since it can dramatically change the obtained results.

Although both strategies 1 and 4 form and break clusters during runs, their energy characteristics are different. Strategy 1 is more similar to cockroach behavior, where clustering is modulated by resting. This strategy allows robots to stay in closer proximity. On the other hand, strategy 4 looks more similar to schooling/flocking observed in fish and flies where agents are constantly in motion. In this strategy average distance between robots in the same cluster is relatively larger due to obstacle avoidance.

We wish to note that behaviors described in this study are not dependent on the specifics of the platform, thus are quite portable. Sound sensors can be replaced with any approximation of clusters around the robot and close range detection of other robots can be implemented with different methods like leds, bump sensors, etc.

Simplicity of the controller used allows a good framework for further modeling. Macroscopic modeling of aggregation is one of the topics the we hope to explore in

future.

In order to increase the aggregation performance, changing strategies during aggregation process might be useful. Starting with fast techniques like fourth strategy and then switching to first strategy might be a viable option. The change of strategy can also be arbitrated by an adaptation mechanism, that uses information robots receive during aggregation process.

Finally an implementation of the minimal aggregation behavior proposed on real robots is planned to arrive on stronger claims about the topic.

REFERENCES

[1] http://en.wikipedia.org/wiki/Golem.

[2] http://www.automates anciens.com/english_version/frames/english_frames.htm.

[3] http://int.kateigaho.com/spr05/robots.html.

[4] "http://marsrovers.jpl.nasa.gov/mission/."

[5] W. G. Walter, *The Living Brain*. New York: W. W. Norton, 1963.

[6] R. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3/4, pp. 189–208, 1971.

[7] G. J. Sussman, *A Computer Model of Skill Acquisition*. New York, NY, USA: Elsevier Science Inc., 1975. ISBN: 044400159X.

[8] E. D. Sacerdoti, *A structure for plans and behavior.* PhD thesis, SRI International, Menlo Park, CA., 1975. AAI7605794.

[9] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," in *Autonomous Mobile Robots: Control, Planning, and Architecture (Vol. 2)*, pp. 233–244, Los Alamitos, CA: IEEE Computer Society Press, 1991.

[10] G. Giralt, R. Chantila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," *Autonomous robot vehicles*, pp. 420–443, 1990. Springer-Verlag New York, Inc.,New York, NY, USA,ISBN: 0-387-97240-4.

[11] H. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, p. 584, Aug 1977.

[12] R. C. Arkin, *Behavior-based Robotics*. Cambridge, MA, USA: MIT Press, 1998. ISBN:0262011654.

[13] R. A. Brooks, "New approaches to robotics," *Science*, vol. 253, pp. 1227–1232, September 1991.

[14] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14–23, 3 1986.

[15] R. R. Murphy, *Introduction to AI Robotics*. MIT Press, 2000. ISBN:0262133830.

[16] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, (New York, NY, USA), pp. 340–347, ACM Press, 1997.

[17] A. Martinoli, A. 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.

[18] L. E. Parker, *Heterogeneous multi-robot cooperation*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.

[19] L. E. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[20] M. J. Mataric, *Interaction and Intelligent Behavior*. PhD thesis, MIT EECS, MIT AI Lab Tech Report AITR-1495, Aug 1994.

[21] C. R. Kube and H. Zhang, "Collective robotic intelligence," in *1992 International Conference on Simulation of Adaptive Behaviour*, pp. 460–468, 1992.

[22] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, pp. 375–397, 1996.

[23] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 7–27, 1997.

[24] S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraulaz, *Self-Organization in Biological Systems*. Princeton, NJ, USA: Princeton University Press, 2001. ISBN: 0691012113.

[25] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 25–34, ACM Press, 1987. ISBN: 0-89791-227-6.

[26] V. Trianni, R. Groß, T. Labella, E. Şahin, and M. Dorigo, "Evolving aggregation behaviors in a swarm of robots," in *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)* (W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, eds.), vol. 2801 of *Lecture Notes in Artificial Intelligence*, pp. 865–874, Springer Verlag, Heidelberg, Germany, 2003.

[27] R. Groß and M. Dorigo, "Evolving a cooperative transport behavior for two simple robots," in *Artificial Evolution – 6th International Conference, Evolution Artificielle (EA 2003)* (P. Liardet, P. Collet, C. Fonlupt, E. Lutton, and M. Schoenauer, eds.), vol. 2936 of *Lecture Notes in Computer Science*, pp. 305–317, Springer Verlag, Berlin, Germany, 2004.

[28] V. Trianni, E. Tuci, and M. Dorigo, "Evolving functional self-assembling in a swarm of autonomous robots," in *From Animals to Animats VIII. Proceedings of the 8^{th} International Conference on Simulation of Adaptive Behavior* (S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, eds.), pp. 405–414, MIT Press, Cambridge, MA, 2004.

[29] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L. M. Gambardella, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots*, vol. 17, no. 2-3, pp. 223–245, 2004.

[30] S. Nolfi and D. Floreano, *Evolutionary Robotics*. The MIT Press, November 2000.

[31] 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. Şahin and W. Spears, eds.), no. 3342, (Berlin Heidelberg), pp. 143–152, Springer-Verlag, 2005.

[32] W. Spears, D. Spears, J. Hamann, and R. Heil, "Distributed, physics-based control of swarms of vehicles," *Autonomous Robots*, vol. 17, pp. 137–162, 2004.

[33] E. Bahçeci, O. Soysal, and E. Şahin, "A review: Pattern formation and adaptation in multi-robot systems," Tech. Rep. CMU-RI-TR-03-43, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Oct 2003.

[34] O. Soysal and E. Şahin, "Probabilistic aggregation strategies in swarm robotic systems," in *Proc. of the IEEE Swarm Intelligence Symposium*, (Pasadena, California), June 2005.

[35] E. Şahin and W. Spears, eds., *Swarm Robotics Workshop: State-of-the-art Survey*, vol. 3342 of *Lecture Notes in Computer Science*. Berlin Heidelberg: Springer-Verlag, 2005.

[36] M. Dorigo and E. Şahin, "Special issue: Swarm robotics," *Autonomous Robots*, vol. 17, pp. 111–113, 2004.

[37] J. L. Deneubourg, A. Lioni, and C. Detrain, "Dynamics of aggregation and emergence of cooperation," *Biological Bulletin*, vol. 202, pp. 262–267, June 2002.

[38] R. Jeanson, C. Rivault, J. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz, "Self-organised aggregation in cockroaches," *Animal Behaviour*, vol. 69, pp. 169–180, 2005.

[39] C. Melhuish, O. Holland, and S. Hoddell, "Convoying: using chorusing to form travelling groups of minimal agents," *Journal of Robotics and Autonomous Systems*, vol. 28, pp. 206–217, 1999.

[40] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Gathering of asynchronous robots with limited visibility.," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 147–168, 2005.

[41] N. Gordon, I. A. Wagner, and A. M. Bruckstein, "Gathering multiple robotic a(ge)nts with limited sensing capabilities," in *Lecture Notes in Computer Science*, vol. 3172, pp. 142–153, Jan 2004.

[42] V. Gazi and K. M. Passino, "Stability analysis of swarms," *IEEE Transactions on Automatic Control*, vol. 48, pp. 692 – 697, April 2003.

[43] V. Gazi, "Swarm aggregations using artificial potentials and sliding mode control," in *42nd IEEE Conference on Decision and Control*, vol. 2, pp. 2041– 2046, 2003.

[44] V. Gazi and K. M. Passino, "A class of attractions/repulsion functions for stable swarm aggregations," *International Journal of Control*, vol. 77, pp. 1567–1579, Dec 2004.

[45] Z. Lin, B. Francis, and M. Ma, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Transactions on Automatic Control*, vol. 50, pp. 121–127, Jan 2005.

[46] S. T. Kazadi, *Swarm Engineering*. PhD thesis, Caltech, 2000.

[47] C. Lee, M. Kim, and S. Kazadi, "Robot clustering," in *IEEE SMC [submitted]*, 2005.

[48] F. Mondada, G. C. Pettinaro, A. Guignard, I. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo, "Swarm-bot: a new distributed robotic concept," *Autonomous Robots*, vol. 17, no. 2-3, pp. 193–221, 2004.

[49] M. Dorigo, E. Tuci, R. Groß, V. Trianni, T. H. Labella, S. Nouyan, and C. Ampatzis, "The swarm-bots project," in *Swarm Robotics Workshop: State-of-the-art Survey* (E. Sahin and W. Spears, eds.), no. 3342 in Lecture Notes in Computer Science, (Berlin Heidelberg), pp. 31–44, Springer-Verlag, 2005.

[50] "http://www.cm-labs.com/."

[51] "http://www.ageia.com/novodex.html."

[52] "http://www.havok.com/."

[53] "http://ode.org/."

[54] "http://www.llnl.gov/eng/mdg/codes/dyna3d/body_dyna3d.html."

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

[56] "http://playerstage.sourceforge.net/gazebo/gazebo.html."

[57] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[58] Webots, "http://www.cyberbotics.com." Commercial Mobile Robot Simulation Software.

[59] http://www.ais.fraunhofer.de/KURT2/.

[60] V. Trianni, S. Nolfi, and M. Dorigo, "Cooperative hole avoidance in a *swarm-bot*," *Robotics and Autonomous Systems*, 2005. to appear.

[61] V. Trianni, T. H. Labella, and M. Dorigo, "Evolution of direct communication for a swarm-bot performing hole avoidance," in *Ant Colony Optimization and Swarm Intelligence – Proceedings of ANTS 2004 – Fourth International Workshop* (M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, eds.), vol. 3172 of *Lecture Notes in Computer Science*, pp. 131–142, Springer Verlag, Berlin, Germany, 2004.

[62] http://www.csm.ornl.gov/pvm/pvm_home.html.

[63] http://gridengine.sunsource.net/.