To The Graduate School:

The members of the Committee approve the thesis of Suranga D. Hettiarachchi presented on September 5, 2007.

William M. Spears, Chairman

Diana F. Spears,

Thomas A. Bailey Jr.

Richard C. Anderson-Sprecher

David R. Thayer

APPROVED:

Jeffrey Van Baalen, Head, Department of Computer Science

Don Roth, Dean, The Graduate School

Hettiarachchi, Suranga D., Distributed Evolution for Swarm Robotics, Ph.D.,
Department of Computer Science, December 2007.

Traditional approaches to designing multi-agent systems are offline, in simulation, and assume the presence of a global observer. Artificial Physics (AP) or *physicomimetics* (Spears and Gordon 1999) can be used to self-organize swarms of mobile robots into formations that move towards a goal. Using an offline approach, we extend the AP framework to moving formations through obstacle fields. We provide important metrics of performance that allow us to (a) compare the utility of different generalized force laws in the artificial physics framework, (b) examine trade-offs between different metrics, and (c) provide a detailed method of comparison for future researchers in this area.

In the online, real world, a global observer may be absent, performance feedback may be delayed or perturbed by noise, agents may only interact with their local neighbors, and only a subset of agents may experience any form of performance feedback. Under these constraints, designing multi-agent systems is difficult. We present a novel approach called"Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios" or DAEDALUS to address these issues, by mimicking more closely the actual dynamics of populations of agents moving and interacting in a (task) environment.

This thesis merges DAEDALUS and AP by using obstacle avoidance as a case study to illustrate the feasibility of DAEDALUS when the environment changes. We present empirical and practical results that address (a) offline vs. online learning, (b) obstructed perception, (c) homogeneous vs. heterogeneous agent cooperation, and (d) implementation of obstacle avoidance with real robots.

DISTRIBUTED EVOLUTION FOR SWARM ROBOTICS

by
Suranga D. Hettiarachchi

A thesis submitted to the Department of Computer Science
and the Graduate School of the University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
COMPUTER SCIENCE

Laramie, Wyoming
December 2007

DEDICATION

This thesis is dedicated to my mother Malinie, father Premadasa, and my two sisters, Priyanka and Kanchana.

ACKNOWLEDGMENTS

This thesis is the result of my four and a half years of research for obtaining my degree in Computer Science. This is not just my own journey; I have been accompanied and supported by so many people who I admire the most. I would like to take this opportunity to extend my heartiest gratitude to all of them.

Let me thank my advisor Dr. William M. Spears for his steadfast support and encouragement. His passionate dedication to science, constant flow of ideas, and conviction to hold higher standards inspire and enrich me as a student, teacher, researcher and a scientist that I want to be. I am truly indebted to him for opening up a whole new world for me and, of course, for funding me.

I would also like to thank Dr. Diana F. Spears for first taking me in as a Ph.D. student and laying a solid foundation for my long and arduous battle for a Ph.D. I am grateful to her for introducing me to Dr. William M. Spears and continuously supporting me to become a better researcher.

Many thanks to the rest of my committee members, Dr. Thomas Bailey, whose immense knowledge in physics made me less dependent on physics books at the library, and Dr. Richard C. Anderson-Sprecher and Dr. David R. Thayer who gave me valuable suggestions for improvements.

I am grateful to Ms. San Dee Hutton, Mr. Shawn McGinnis, and Ms. Dolores McGinnis for being the most supportive friends of mine, and the University of Wyoming for supporting me with scholarships. Also, I like to acknowledge great ideas and collaborations by all of my colleagues, especially Paul, Dimitri, Wes, Derek, Tom and Caleb, at the University of Wyoming, Distributed Robotics Laboratory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table **Page**

Chapter 1

Introduction

1.1 Introduction

Accomplishing complex tasks within the field of robotics has traditionally involved expensive, remotely controlled individual robots. These robots are typically developed, trained, and tested in highly structured environments with predefined and unchanging conditions. This approach is unsatisfactory for at least two important reasons. First, as the requirements of missions and tasks increase in complexity the cost of these types of robots rapidly becomes prohibitive. Second, when these robots are exposed to novel environments with conditions for which they have not trained, they tend to meet with catastrophic failures.

One of the earliest ancestors of this type of robot was developed at the Artificial Intelligence Center at Stanford Research Institute in the late 1960s. This robot, dubbed Shakey for its wobbly gait (see Figure 1.1), was tested in specially prepared rooms filled with large painted obstacles. Using a black-and-white on-board camera as a primary sensor and an off-board computer which analyzed the visual input from the camera and translated it into first order predicate calculus, Shakey navigated from room to room attempting to achieve goals it received by teletype. The success of this robot was due in great part to the carefully engineered environment and experiments (Nilsson 1984).

Figure 1.1: Shakey the Robot.

Since that time, there have been tremendous improvements in robotic technology leading us to modern examples like the remotely controlled and astronomically expensive Predator Unmanned Aerial Vehicle (UAV) seen in Figure 1.2. Developed by General Atomics Aeronautical Systems, the Predator is a specific purpose robot used mainly for surveillance and reconnaissance missions. The Predator is capable of real-time distribution of its surveillance imagery drawn from synthetic aperture radar and electro-optical and infrared cameras. This data can be sent to front line command certers or to any worldwide location via satellite communication links. These capabilities have lead to extensive use of the Predator by the United States Air Force (USAF) owing to recent world events (see Figure 1.2). The success of the Predator depends on the efficient control of its functions by human operators in ground con-

trol stations. Hence, the robots mission is subject to human error and other types of failures at the control stations. The incredible expense of this device makes the possibility of such errors a devastating problem.

Figure 1.2: Predator B-2 Unmanned Aerial Vehicle.

In these traditional approaches, the robot is entirely dependent upon the human operator. The performance feedback from human to robot is vital and any delay or perturbation of this feedback loop could jeopardize the mission. Without involvement of the human observer, the robots are incapable of performing a task; consequently, removing the human global observer could be fatal to the success of the mission. The lack of autonomy in these robots is a great cause for worry.

Existing difficulties with traditional approaches to solving problems using robots encourage us to focus on designing rapidly deployable, cost-effective, scalable, adaptive, and robust swarms of robots. We prefer autonomous, distributed, mobile, sensing robots in our swarms. Our objective is to provide a scientific, yet fully practical, approach to the design and analysis of swarm robotic systems.

1.2 Swarm Robotics

Swarm robotics refers to an approach that makes use of a large number of physical robots to solve complex problems. One of the important aspects of this approach is that a solution to a problem emerges through the local interactions of the robots in the swarm and the interaction of these robots with an environment. A large collection of inexpensive and highly capable robots improves the availability of resources for solving complex problems. The swarm robotics approach is an improvement over traditional robotic approaches, providing us with effectiveness and robustness. Robot swarms are highly effective, because they can perform tasks that one expensive robot cannot. They are robust in the sense that if some robots fail, the swarm can still achieve the task. Robot swarms provide the following advantages:

computational efficiency: the availability of multiple processors in a swarm reduces the computational overhead of one large processor in an expensive single robot. We can exploit the concurrency of computation with multiple processors in a swarm.

reliability: swarms are more reliable due to their multiplicity. Recovery from component failures is extremely important for mission critical tasks. A swarm of robots allows us to accomplish the mission even when components of some robots fail.

extensibility: since a swarm consists of multiple robots, we can alter the number of robots in the swarm and their capabilities. This extensibility allows us to possibly accomplish several different tasks in a single mission that a single robot could not provide. It also permits us to increase the size of the swarm giving the ability to explore a larger area.

responsiveness: individual robots in a swarm can act as separate hardware or software modules; this modularity allows us to isolate and handle anomalies locally. The responsiveness of individual robots minimizes the effect of anomalies propagating into other areas of the swarm.

maintainability: maintaining a swarm is also easy due to system modularity. Individual robots can be removed from a swarm or reintroduced to a swarm without jeopardizing the effectiveness of the task being accomplished.

Figure 1.3 shows a sequence of snapshots of four Maxelbot robots developed by the University of Wyoming Distributed Robotics Laboratory (UW-DRL), moving in an outdoor terrain while maintaining a diamond formation. These four robots maintain their formation as a swarm using a novel physics-based algorithm called AP-lite which we will discuss in Chapter 8.

1.3 Our Objectives

In our research, we focus on several general objectives that may allow us to address some of the swarm robotic issues and concerns. There are four general objectives:

synthesize distributed multi-robot systems that contain tens to thousands of robots interacting locally.

achieve the desired macroscopic/global behavior through microscopic/local interactions.

be as understandable and predictable as possible.

achieve desired behavior with minimum sensor information.

Figure 1.3: Four Maxelbots maintain a diamond formation in outdoor terrain.

Our specific objective within the context of this thesis is to address the related issues and concerns of a swarm of robots that reaches a goal while avoiding obstacles and maintaining a cohesive formation, even if the environment changes. Our focus is to provide an analysis of our specific objective that extensively contributes to the understanding of general swarm robotics issues.

1.4 Swarm Tasks

Swarm robotic approaches have become popular for solving complex tasks due to the robustness provided by multiple robots. Swarm robotic researchers have been trying to solve complex problems related to many different tasks. Though some of these tasks may appear to be trivial to a human, automation of the solution process using robots is a difficult and complex problem. We address several tasks that are important to swarm robotics research.

1.4.1 Search and Rescue

Imagine an earthquake in a large apartment complex? There is massive destruction of property; many people are trapped in the rubble without any help. The first responders lack resources to provide for the needy and begin a search and rescue mission. The authorities decide to release a swarm of robots with different capabilities to provide much needed help. Some of the robots search for survivors, while some others clear rubble. Another group of robots starts moving injured and sick humans to a central location while a third group of robots begins treating the injured. This sounds similar to a page from science fiction but the recent advancements in swarm robotics research appear to be solving search and rescue missions using swarms of robots.

The work by the Center for Robot Assisted Search and Rescue(CRASAR) at the University of South Florida has contributed to the advancement of search and rescuerobots (Murphy and Burke 2005). Several prototypes of robots developed at CRASAR are already being used by re departments and bomb squads.

Wolf and Choset at the CarnegieMellon University Robotics Institute have developed robot prototypes that are capable of maneuvering through narrow crawl spaces (Wolf, Brown, Casciola, Costa, Schwerin, Shamas, and Choset 2003). These \Serpentine" manipulators are rich with sensors and are motivated by the behavior of snakes and worms. It uses a vision system to search con ned environments. One of the major disadvantages of the \Serpentine" is its lack of complete autonomy.

1.4.2 Surv eillance

Observation of humans or objects from a distance using electronic equipment has become a popular method for gathering information. One of the approaches to accomplishing this surveillance task is to use multiple robots as a swarm. Swarm robotic approaches are recognized as promising surveillance techniques. Some examples of uses of robotic swarms in surveillance include identifying enemy targets, detecting unfriendly hazardous events, and controlling air tra c. Some of the signi cant work in surveillance is done at the UW-DRL by Kerr et al. and Spears et al. Kerr et al. introduced the kinetic theory approach to accomplishing the surveillance task (Kerr and Spears 2005) while Spears et al. analyzed rule-based approaches to the coverage/surveillance tasks using multiple UAVs (Spears, Zarzhitsky, Hettiarachchi, and Kerr 2005).

1.4.3 Chemical Plume Tracing (CPT)

Another task that researchers have been studying is the application of a swarm of robots toward chemical or biological plume source tracing. The task is to rapidly localize the emitter using multiple robots. Work at UW-DRL has produced promising results in this area (Zarzhitsky, Spears, Thayer, and Spears 2004). This work uses computational fluid dynamic techniques to localize the emitter. Prior approaches use either chemotaxis in which robots follow the local gradient of the plume concentration or anemotaxis in which robots are guided upstream by the velocity of the plume. The fluxotaxis approach introduced by Zarzhitsky has produced superior results than the previous two approaches.

1.4.4 Terrain Mapping

Building a map of an unknown terrain is another task that can be accomplished with a swarm of robots. A swarm of robots with a sensor suite can be deployed to create a map of an unknown area. The major advantage of this approach is that robots can simultaneously develop maps of non-overlapping regions making this task less time consuming than trying to accomplish the same with a single robot. Huber et al. at the Carnegie Mellon University introduced a new approach to 3-D terrain mapping using terrestrial range sensors (Huber and Herbert 1999). Combining this sensor technology with other robotic platforms allows us to achieve a swarm of terrain mapping robots.

1.4.5 Land Mine Detection

The objective is to use a collection of cooperative robots that are dispersed in a terrain to locate land mines. Mine detection is a very sensitive and possibly expensive task.

Traditional practice is to use human subjects (with or without heavy machinery) to detect and destroy mines in a mine field. This approach is extremely risky and time consuming. It is practical to use inexpensive autonomous robots to accomplish this task. Cooperative robot swarms are even more effective, since they can explore the mine fields in for less time.

An advanced robot system for mine detection developed by Fuji Heavy industries Co. Ltd is a step in the right direction (Aoyama, Ishikawa, Seki, Okamura, Ishimura, and Satsumi 2007). Their robot consists of four crawlers and two work arms and is capable of operating in rough terrain. Several disadvantages of this robot are limited deployability, massive size, lack of autonomy, and difficulty of transporting to remote locations. These disadvantages restrict this robot from being deployed in a swarm.

Cepolina and Zoppi argue the importance of developing cost effective mine localizing robots and introduce four concept crawlers for this task (Cepolina and Zoppi 2003). They investigate two methods: (1) transporting a sensor suite to a mine field by carrying it on a suitable platform similar to Aoyama's work, (2) bringing air samples from a mine field to a remote safe location. This second method is called Remote Explosive Scent Tracing (REST).

1.4.6 Task Environments

There are various types of task environments for testing different tasks. Our focus is to use widely accepted task environments that are less complex to simulate. The majority of the environments that have been studied are highly structured and specifically suited to test a single task. It is difficult to identify and design a generic task environment to suit all of the tasks that we discussed above.

Researchers have commonly used various types of simulated environments to test their robot swarms. However, the use of simulations to model robots is often criticized

due to their inability to model complex real-world scenarios. We believe that, even with the limitations of simulated environments, these models are still of great value in exploring robot behaviors. Simulations can certainly be used as basic investigative tools in the study of a complex swarms of robots. Thus, we use simulations as a first step towards the implemertation of a working system.

Figure 1.4 shows a simulation of a simple structured environment where a swarm of robots cooperate to accomplish a task. Figure 1.5 shows a highly structured lab environment where a swarm of four robots cooperate to accomplish a task (Courtesy of the Center of Engineering Science and Advance Research, Oak Ridge National Laboratory)

1.5 Obstacle Avoidance

For all the tasks in the previous sections, avoiding obstacles in the task environment is extremely important. The phrase obstacle avoidance refers to combining software and hardware methodologies that intelligently drive robots from one point to another without colliding with other objects in the environment. Our focus is to design, develop, and implement a swarm of robots that is capable of autonomous navigation towards a goal in an obstacle-laden environment while maintaining a hexagonal lattice formation. If the robots have limited sensor information and/or the environment changes while the robots are in the field, successfully accomplishing the obstacle avoidance task becomes much more difficult. We accomplish formation control, obstacle avoidance, and reaching a goal using techniques inspired by physical systems. We are motivated by physics because aggregate behaviors seen in classical physics are potentially reproducible with collections of mobile robots. We use our understanding of classical physics to derive the collective behaviors for robots. We do not restrict

Figure 1.4: Group of robots in a simulated environment.

Figure 1.5: Group of robots in a highly structured lab environment - Oak Ridge National Laboratory.

ourselves to copying physics precisely so modifications are possible. Our obstacle avoidance approach is rapidly deployable, distributed, scalable, adaptive and cost effective.

1.6 Contributions

The work in this thesis makes many contributions to several areas of swarm robotics research.

Improved performance in obstacle avoidance:

- applied new force law for robot control, to improve performance.
- provided novel objective performance metrics for obstacle avoiding swarms.
- improved scalability of the swarm in obstacle avoidance.
- improved performance of obstacle avoidance with obstructed perception.

Invented a real-time learning algorithm (DAEDALUS):

- demonstrated that a swarm can improve performance by mutating and exchanging force laws.
- demonstrated feasibility of DAEDALUS with obstacle avoidance, in environments three times denser than the norm.
- explored the trade-offs of mutation on homogeneous and heterogeneous swarm learning.

implemented hardware:

- presented a novel robot control algorithm that merges physicomimetics with obstacle avoidance.

1.7 Preview

In this thesis, we focus on several issues relevant to a swarm of robots moving in formation while avoiding obstacles and reaching a goal. Though we are not restricted to a single task environment, we use a simulation world modeled with generally accepted standards and present an empirical analysis of our results. We implement a novel physics-based control algorithm for robot control and obstacle avoidance and present our results with physical robots in an outdoor setting.

In this chapter, we have presented several tasks where obstacle avoidance and swarm formations are important. In Chapter 2, we explore current trends and alternative approaches to solving swarm formation control and obstacle avoidance. We devote Chapter 3 to presenting two physics-based force law algorithms. Chapter 4 explains our parameter optimization method for force laws, and our simulation tool. Chapter 5 provides an empirical analysis of "online learning" using several performance measurement metrics. In Chapters 6 and 7, we present our novel adaptive learning algorithm called "DAEDALUS" and explore the effect of mutation in the "online learning" of distributed swarm robotics using heterogeneous robots. Chapter 8 presents our robot control algorithm (AP-lite), the obstacle avoidance module (OAM), and the results showing the accuracy of AP-lite and OAM using physical robots.

Chapter 2

Related Work

2.1 Introduction

Swarm robotic approaches have become popular for solving complex tasks due to robust capabilities provided by multiple robots. We refer to designing, implementing and testing a swarm of cooperating agents commensurate to some criterion designed to produce a global outcome of a complex task as "swarm engineering" (Kazadi 2005). Swarm engineering approaches have been studied and presented in numerous research articles. This chapter presents some of the approaches adopted by researchers for swarm formation control and obstacle avoidance.

First, we present different approaches to swarm formation control such as behavior-based, rule-based, potential fields (PF), control-theoretic, and physicomimetics. These approaches have been applied to different swarm tasks. Next, we present more relevant work in the obstacle avoidance area. Finally, we explain different sensor network approaches for swarm engineering.

2.2 Behavior-based and Rule-based Swarm Robotics

Behavior-based swarm intelligence techniques are ethologically motivated and have had excellent success with foraging, task allocation, and division of labor problems.

Beni et al. introduced the concept of swarm intelligence while studying cellular robotic systems in 1989 (Beni and Wang 1989). They investigated the properties of simulated self-organizing agents in cellular robotic systems. Bonabeau et al. extended this work providing a rigorous look at the mechanisms underlying collective behavior in social insects (Bonabeau, Dorigo, and Theraulaz 1999). This work, titled "Swarm Intelligence: From Natural to Artificial Systems" includes attempts to design distributed cooperative problem-solving algorithms inspired by the collective behavior of insect colonies. They present experimental results of several studies related to foraging, division of labor, clustering and sorting, nest building, and cooperative transportation tasks. Hayes et at. described a biologically motivated distributed algorithm called "Spiral Surge" by which a group of collaborating agents can solve the full odor localization task more efficiently than a single agent (Hayes, Martinoli, and Goodman 2001). They demonstrated that a group of real robots under fully distributed control can successfully traverse a real odor plume which is a sub-task of odor localization.

Both behavior-based and rule-based systems have proved quite successful in exhibiting a variety of behaviors in a heuristic manner. Fredslund and Matarić studied the problem of achieving global behavior in a group of distributed robots using only local sensing and minimal communication, in the context of formations (Fredslund and Matarić 2002). The key concept of their algorithm is that each robot follows a designated "friend" robot at the appropriate angle and distance using a proximity sensor that can provide the angle and distance information of the friend. By panning the sensor appropriately, the algorithm simply keeps the friend centered in the sensor's view. They presented their results using four and eight robots in different formations. Balch and Arkin accomplished robot formations using the following two step process: "detect-formation-position" which is a perceptual process that determines

the robot's position in the formation based on the current environment data, and \maintain-formation" which generates motor commands to direct the robot towards the correct location (Balch and Arkin 1998).

2.3 Potential Fields

One of the earliest physics-based techniques is the potential fields approach (e.g., (Khatib 1986)). Most of the PF literature deals with a small number of robots (typically just one) that navigate through a field of obstacles to get to a target location. The environment, rather than the robots, exert forces. Obstacles exert repulsive forces while goals exert attractive forces (Kim and Khosla 1991; Koren and Borenstein 1991).

Recently, Howard et al. and Vail et al. extended PF to include inter-agent repulsive forces{ for the purpose of achieving coverage (Howard, Matari c, and Sukhatme 2002; Vail and Veloso 2003). Although this work was developed independently of physicomimetics framework, it affirms the feasibility of a physics force-based approach. Another physics-based method is the \Engineered Collective" work by Duncan at the University of New Mexico and Robinett at the Sandia National Laboratory. Their technique has been applied to search-and-rescue and other related tasks (Schoenwald, Feddema, and Oppel 2001).

The social potential fields (SPF) framework is highly related to physicomimetics framework (Reif and Wang 1998). Reif and Wang rely on a force-law simulation that is similar to the physicomimetics approach, allowing different forces between different robots. Their emphasis is on synthesizing desired formations by designing graphs that have a unique potential energy (PE) embedding. Bruemmer et al. also utilize the SPF approach as a means to coordinate group behavior and promote the emer-

gence of swarm intelligence as seen in a colony of ants or swarm of bees (Bruemmer, Dudenhoeffer, McKay, and Anderson 2002).

2.4 Control Theoretic Swarm Robotics

Control-theoretic approaches have also been applied effectively (e.g., (Fax and Murray 2002)). Our approach does not make the assumption of having leaders and followers, as in (Desai, Ostrowski, and Kumar 1998; Desai, Ostrowski, and Kumar 2001).

Fax and Murray considered the problem of cooperation among a collection of vehicles performing a shared task using inter-vehicle communication to coordinate their actions. They applied tools from graph theory to relate the topology of the communication network to formation stability, and provided a mathematical analysis to determine the effect of the graph on formation stability.

Desai et al. proposed a method that uses only local sensor-based information for leader-follower formation control. Their method uses nonlinear control theory to achieve formations. They conducted experiments with several follower robots navigating around an obstacle with one or two leaders. They presented their results with one or more leaders and with one or two follower(s) moving around a single obstacle.

2.5 Physicomimetics Approaches

"Physicomimetics" or artificial physics (AP) is another swarm robotic approach motivated by classical physics. This approach provides excellent techniques for distributed control of large collections of mobile physical agents as well as theoretical foundations for analyzing swarm behaviors. The physicomimetics framework provides an effective basis for self-organization, fault-tolerance and self-repair (Spears, Gordon-Spears, Hamann, and Heil 2004). Our work is an extension of the physicomimetics frame-

work provided by Spears et al. A thorough discussion of this framework is provided in Chapter 3.

Wiegand et al. generalized physicomimetics to extend the power of multi-agent systems by specializing particles and their interactions, and they showed the effectiveness of this generalized representation by evolving a solution to a challenging multi-agent resource protection problem (Wiegand, Potter, Sofge, and Spears 2006).

Kerr et al. presented two different algorithms based on the physicomimetics framework, achieving maximal coverage throughout an unexplored corridor (Kerr and Spears 2005). The first algorithm is an extension of the physicomimetics, which employs virtual repulsive forces for multi-agent coordination. The second algorithm is based on the Kinetic theory of gases which models inter-particle and particle-wall collisions.

Using the physicomimetics approach Zarzhitsky et al. explored the CPT task with obstacle avoidance (Zarzhitsky, Spears, and Spears 2005). In the algorithms presented, robots share flow-field variables with their neighbors and use the values to calculate the next way-points based on the robots' own local coordinate axes. These values decide the robot's next move which translates into a virtual force. They implemented obstacle avoidance using the CPT algorithm in a hierarchical architecture, in which robots usually navigate around obstacles before collision avoidance becomes necessary.

2.6 Obstacle Avoidance

In the specific context of obstacle avoidance, the most relevant papers are (Balch and Arkin 1998; Balch and Hybinette 2000; Fredslund and Matarić 2002). Balch and Arkin examine the behavior of four robots moving in formation through an obstacle

eld with 2% coverage. Balch et al. extend this to an obstacle eld of 5% coverage, and also investigate the behavior of 32 robots moving around one medium size obstacle (Balch and Hybinette 2000). Fredslund and Matari c examine a maximum of eight robots moving around two wall obstacles (Fredslund and Matari c 2002). To the best of our knowledge, we are the rst to systematically examine larger numbers of robots and obstacles using swarm robotics approaches.

Schultz presented a projective planning algorithm for real-time autonomous underwater navigation through obstacles (Schultz 1991). He use \SAMUEL" , a learning system based on genetic algorithms to learn high-performance reactive strategies for navigation and collision avoidance. He presented results in simulation with a single autonomous underwater vehicle and showed that SAMUEL can achieve a success rate of 96% on randomly generated mine elds over a human-designed strategy that has an average success rate of only 8%.

Simmons presents a new method for local obstacle avoidance by indoor mobile robots that formulates the problem as one of constrained optimization in velocity space (Simmons 1996). The robot chooses velocity commands that satisfy all the constraints such as the physical limitations of the robot and the environmental limitations, and maximize an objective function that trades o speed, safety and goal-directedness. He demonstrates the utilit y of this algorithm using a single physical robot in a controlled laboratory environment.

Another relevant work in obstacle avoidance is dynamic window approach (DWA). This approach is mostly suited for robots moving at a high speed and is derived directly from the robot's motion dynamics (Fox, Burgard, and Thrun 1995). In the DWA, the search for commands that control the robot is carried out in space of velocities. The robot only considers the velocities that are safe with respect to the obstacles. They present their results with tests done using an autonomous robot

called \RHINO" which uses proximity sensors to compute the dynamic window.

Borenstein et al. present an approach that permits the detection of unknown obstacles simultaneously with the steering of the mobile robot to avoid collisions and advance toward the target (Borenstein and Koren 1989). They use a \virtual force eld" method that uses potential elds for navigation and certainty grids for obstacle representation. They showed that this method is especially suitable for noisy and inaccurate sensor inputs. They also addressed the \lo cal minimum trap" problem where robots get stuck in a U-shaped obstacle. We address this issue in Chapters 6 and 7.

O'Hara uses an embedded network distributed throughout the environment to approximate the path-planning space and uses the network to compute a navigational path using a framework called \GNA Ts" when the environment changes (O'Hara, Bigio, Dodson, Irani, Walker, and Balch 2005). The dynamism of the environment is modeled with an opening and closing door in the experimental setup. However, the embedded network is immobile, whereas our network is completely mobile.

2.7 Summary

We have discussed several di erent approaches to both robot swarm formation control and robot obstacle avoidance. Each of these approaches has its advantages and limitations.

Behavior-based and rule-based techniques do not make use of potential elds or forces. Instead, they deal directly with velocity vectors and heuristics for changing those vectors (although the term \p otential eld" is often used in the behavior-based literature, it refers to a eld that di ers from the strict Newtonian physics de nition). Also, they either assume the existence of a global controller or they are based on

heuristics. Approaches that assume the presence of a global controller fail to adapt well to unknown environments.

The physicomimetics approaches focus on potential energy and force balance equations. Physicomimetics is capable of creating fluid-like formations as well as solid formations. These different formations can be created by simply modifying force law parameters. In addition, physicomimetics is capable of providing behavioral assurance to agents that adapt in dynamic environments. Physicomimetics is fully distributed and does not assume the presence of a global controller, and is computationally efficient due to the lack of computation of potential fields (Spears, Gordon-Spears, Hamann, and Heil 2004). Our work is a novel extension to physicomimetics that provides superior results in the area of heterogeneous and homogeneous swarm control using the obstacle avoidance task.

Chapter 3

Physicomimetics

3.1 Introduction

This chapter provides an overview of the framework for distributed control of robots in a swarm, called "physicomimetics" or "artificial physics" (Spears and Gordon 1999). Spears and Gordon use the term "artificial" (or virtual) because although this framework is motivated by natural physical forces, it is not restricted to them. Although the forces exerted upon a robot by other robots and that environment are virtual, robots act as if they are real. Thus the robot's sensors must sense far enough to allow it to compute the force to which it is reacting. The robot's effectors must allow it to respond to these perceived forces. Spears's explain their motivation for this framework in (Spears, Gordon-Spears, Hamann, and Heil 2004):

> At first blush, creating hexagons appears to be somewhat complicated, requiring sensors that can calculate distance, the number of neighbors, their angles, etc. However, only distance and bearing information is required. To understand this, recall an old high-school geometry lesson in which six circles of radius R can be drawn on the perimeter of a central circle of radius R. Figure 3.1 illustrates this construction. If the particles (shown as small circular spots) are deposited at the intersections of the

circles they form a hexagon with a particle in the middle.

Figure 3.1: How circles can create hexagons.

There are two potential advantages to this approach. First, in the real physical world, collections of small entities yield surprisingly complex behavior from very simple interactions between the entities. Thus, there is an accepted precedence that complex control can emerge through simple local interactions. Second, since the approach is largely independent of the size and number of robots, the results scale well to larger robots and larger sets of robots.

3.2 Physicomimetics Approach

In the physicomimetics framework, virtual physics forces drive a swarm robotics system to a desired configuration or state. The desired configuration is one that minimizes overall system potential energy and the system acts as a molecular dynamics ($\vec{F} = m\vec{a}$) simulation.

Each robot has position $\vec{p}$ and velocity $\vec{v}$. We use a discrete-time approximation to the continuous behavior of the robots, with time-step t. At each time step, the

position of each robot undergoes a perturbation Δp. The perturbation depends on the current velocity, i.e., $\Delta p = v \Delta t$. The velocity of each robot at each time step also changes by Δv. The change in velocity is controlled by the force on the robot, i.e., $\Delta v = F \Delta t / m$, where m is the mass of that robot and $\vec{F}$ is the force on that robot. F and v denote the magnitude of vectors $\vec{F}$ and $\vec{v}$. A frictional force is included for self-stabilization and modeled by decreasing the robot's velocity by a constant multiplicative factor. Figure 3.2 shows the perturbation of the robots R and R_4 due to forces exerted upon them by other robots and the environment.

Figure 3.2: Robots R and R_4 undergo a perturbation to their positions due to forces from other robots and the environment. Robot R_4 does not sense forces from robots R_1 through R_3 due to sensor proximity.

Our objective is to have the physicomimetics framework map easily to physical hardware, and Spears's physicomimetics framework reflects this design philosophy. Having a mass m associated with each robot allows our simulated robots to have momentum. Robots need not have the same mass. The frictional force allows us to model actual friction, whether it is unavoidable or deliberate, in the real robotic system. With full friction, the robots come to a complete stop between sensor readings

and with no friction the robots continue to move as they sense. The time step t reflects the amount of time the robots need to perform their sensor readings. If t is small, the robots get readings very often whereas if the time step is large, readings are obtained infrequently. We have also included a parameter F_{max}, which provides a necessary restriction on the acceleration a robot can achieve. Also, a parameter V_{max} restricts the maximum velocity of the robots (and can always be scaled appropriately with t to ensure smooth path trajectories).

3.3 Newtonian Force Law

The Newtonian Force Law (Newtonian) has been used in prior work (Spears, Spears, Heil, Kerr, and Hettiarachchi 2004) and is a generalization of the "Newtonian" gravitational force law which includes both attraction and repulsion. The force law is:

$$F_{i,j} = \frac{m_i m_j G}{r^p} \tag{3.1}$$

$F \leq F_{max}$ is the magnitude of the force between two robots i and j, and r is the distance between the two robots. The masses of the robots are denoted as m_i and m_j, and are assumed to be set to 1.0 in this thesis. The variable G affects the strength of the force. The variable p is a user-defined power that controls the reduction in strength with distance. The force is repulsive if $r < R$, attractive if $r > R$, and is zero beyond a certain range (e.g., $1.5R$), to enforce the local nature of the force law. R is the desired separation between a robot and neighboring robots. In order to achieve optimal behavior, the values of G, p, and F_{max} must be determined as well as the amount of friction. The Newtonian force law generally creates rigid formations that act as solids, even in the presence of sensor and locomotion uncertainty.

3.4 Lennard-Jones (LJ) Force Law

In this thesis we also investigate the utility of a second force law, which is a generalization of the Lennard-Jones (LJ) force law. The LJ potential function was first proposed by John Lennard-Jones in 1929. This potential function models two distinct forces between neutral molecules and atoms. The forces are based on the distances between the molecules; at long ranges the attractive force makes the molecules move closer and at short ranges the repulsive force makes the molecules move apart, causing the molecules to maintain a natural balance. The LJ potential function can be given by the expression:

$$LJ\,P_r = 4\left[\left(\frac{}{r}\right)^{12} - \left(\frac{}{r}\right)^{6}\right] \tag{3.2}$$

As shown in Figure 3.3, whenever = 1 and = r, the interaction energy between two molecules is at zero, which is the molecule's equilibrium. When the separation distance r > 1, interaction energy quickly decreases to -1 and then increases and eventually reaches zero due to longer range, causing non-interaction between molecules. When r < 1, the interaction energy between two molecules is very high, reaching 1 . Due to the behavior shown by the LJ potential function, this becomes an ideal function to model interactions between robots and their environments.

To model interactions of robots in a swarm, we need to transform the LJ potential function to a force function. Since the force between two molecules is the negated derivative of the potential,

$$F = -\frac{d(LJ\,P_r)}{dr}; \tag{3.3}$$

the force between robots i, j could be derived as:

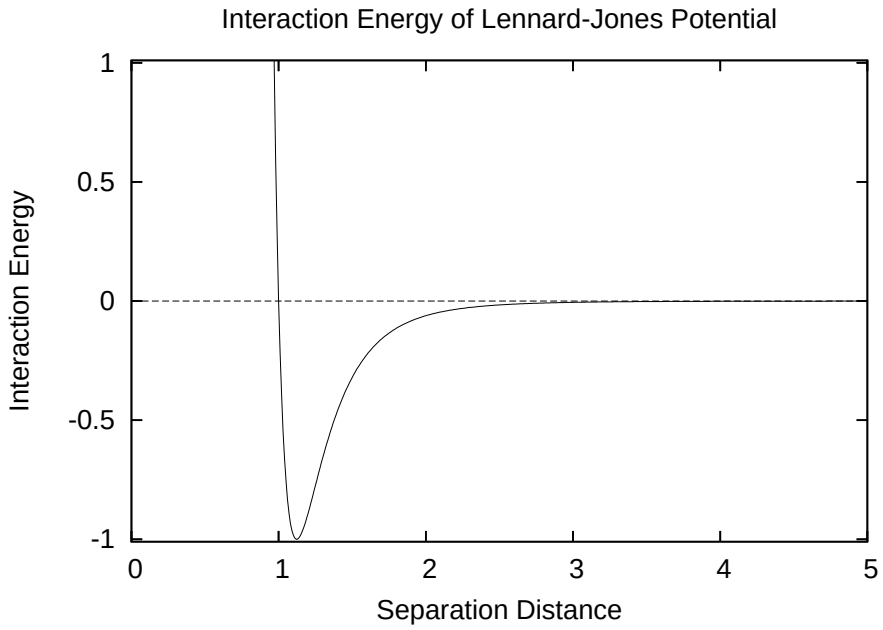

Figure 3.3: Interaction potential of LJ with = 1 and = 1.

$$F_{i;j} = \frac{4}{r}\left[\frac{12}{r^{13}} + \frac{6}{r^{7}}\right] \tag{3.4}$$

and = R is the desired distance between two robots. We derive the force function for interaction between two robots as:

$$F_{i;j} = 24\left[\frac{2d^{12}}{r^{13}} - \frac{c^{6}}{r^{7}}\right] \tag{3.5}$$

Again, F F_{max} is the magnitude of the force between two robots, and r is the distance between the two robots. The variable a ects the strength of the force, while c and d control the relative balance between the attractive and repulsive components. In order to achieve optimal behavior, the values of , c, d, and F_{max} must be determined as well as the amount of friction. Our motivation for trying the LJ force law is that (depending on the parameter settings) it can easily model crystalline solid

formations, liquids, and gases. The pseudocode of the physicomimetics algorithm that uses the LJ force law for robot-robot interactions can be seen in Figure 3.4. By changing the parameter settings of the force law, we can model liquid or solid behavior of the swarm.

3.5 Summary

We presented two approaches to the physicomimetics framework which are capable of distributed control of robots in a swarm. In the physicomimetics framework, virtual physics forces drive a swarm robotics system to a desired configuration or state. Through the sensors and effectors robots sense and react to virtual forces from other robots and environment as if these forces are real. These approaches provide us with two advantages: complex behavior of the system can be achieved from simple interactions between robots, and the behavior scales to large numbers of robots.

```
float   distance,   turn,   vx,   vy,   delta_vx,   delta_vy
float   netforce,   c,   d,
float   deltax,   deltay,   current_x,   current_y,   next_x,   next_y
float   delta_t  = 1.0
float   VMAX= 20.0
float   FR = 0.9
float   R = 50.0
float   Fx = 0
float   Fy = 0
float   TIME_STEP= mass = 1.0

void  compute_ newLocation()
     v = FR *  v
     for  all  robots  except  this  robot  r_i
          determine  distance  r  to  robot  r_j
          determine  the  spherical  coordinate      to  r_j
          if  ( r  <= 1.5  *  R)
               netforce   = interactionForce   (R,  r)
          endif
          Fx = Fx + (netforce   * cos( ))
          Fy = Fy + (netforce   * sin( ))
     endfor
     delta_vx = TIME_STEP*  Fx /  mass
     delta_vy = TIME_STEP*  Fy /  mass
     vx = vx + delta_vx
     vy = vy + delta_vy
     v = sqrt(vx   * vx + vy  *  vy)
     if  (v  > VMAX)
          vx = (vx  *  VMAX)/v
          vy = (vy  *  VMAX)/v
     endif
     deltax  = vx * TIME_STEP
     deltay  = vy * TIME_STEP
     next_x = current_x + deltax
     next_y = current_y + deltay
end compute_ newLocation

float   interactionForce   (desi redDi st ance,  distance)
     float   attractive,     repulsive,     force
     attractive    = 2 *  d *  pow(desiredDistan ce,1 2)  /  pow(distance,13)
     repulsive   = c *  pow(desiredDistan ce,6)  /  pow(distance,7)
     force  = 24 *    * (attractive     -  repulsive)
     if  (force   > FMAX)//  FMAXis  a  learned  parameter
          force   = FMAX
     if  (force   < -FMAX)
          force   = -FMAX
     return   force
end interactionForce
```

Figure 3.4: Pseudocode of the physicomimeticsalgorithm that usesthe LJ force law for robot-robot interactions.

Chapter 4

Evolutionary Learning

4.1 Introduction

Motivated by Darwinian evolution and natural selection, Evolutionary Learning (EL), has been rapidly developing and gaining popularity as a powerful general learning approach. The machine learning community recognizes EL as a population-based learning technique that can be used to find exact or approximate solutions to optimization and search problems. Symbolic systems such as rule-based systems as well as sub-symbolic systems such as artificial neural networks have used evolutionary learning as an optimization tool. EL systems allow selection of an optimum or most satisfactory individual in the last generation as the final learned system (Yao, Liu, and Darwen 1996).

Our reasons for choosing EL depend on the following three motives: first, we have the necessary background for implementing evolutionary learning systems; second, we are aware that evolutionary learning outperforms Reinforcement Learning (RL) in non-Markovian tasks (de Croon, van Dartel, and Posma 2005). Croon empirically demonstrates that RL and EL methods result in different levels of performance when applied to a non-Markovian task like the active categorical perception task (ACP). In the ACP task, the agent has to categorize falling objects by catching or avoiding them. During the ACP task, the proportion of the ambiguous sensor states can be varied.

Croon et al. demonstrate that EL outperforms RL at all levels of sensor ambiguity and the relative performance of EL increases with the proportion of ambiguous sensor states. Croon argues that the reason for this performance difference is that in RL the learned policy consists of those state-action pairs that individually have the highest estimated values, while the performance of a policy for a non-Markovian task highly depends on the combination of state-action pairs selected. The third reason for using this population-based stochastic algorithm is that it quickly generates individuals that have robust performance.

Given generalized force laws, such as the Newtonian force law or the Lennard-Jones (LJ) force law that we discussed in Chapter 3, it is necessary to optimize the parameters to achieve the best performance. We accomplish this task using an Evolutionary Algorithm (EA). We develop an EL system for a swarm of robots learning the obstacle avoidance task (Spears, Jong, Back, Fogel, and de Garis 1993). Our intention is to have the robots learn the optimum parameter settings of the force laws (rules); thus they successfully maintain formation while avoiding obstacles, and reaching a goal in a complex obstacle-laden environment.

4.2 Evolutionary Algorithms

We use an Evolutionary Algorithm (EA) to optimize force law parameters necessary for a swarm of robots to learn to navigate within a complex environment with obstacles and a goal. EAs are a part of the EL approach and as such are motivated by biological evolution. An organism's attributes such as its anatomy and behavior are decided by its genotype or biological data structures. In a computer program these data structures can be easily manipulated and can also be interpreted as information that encodes different characteristics (phenotype) of the organism. An EA can use

```
Randomly generate an initial population of N individuals
Evaluate the initial population
While termination criterion not met
    Select parents
    Apply reproduction operators and produce children
    Evaluate children
    Select individuals for next generation
endwhile
```

Figure 4.1: Typical EA pseudocode.

genetic operators such as reproduction, natural selection, mutation, recombination, and competition which are inspired by Darwinian evolution. Thus, using an EA to optimize parameter settings for a robot or any mechanical device makes perfect sense.

EAs are stochastic optimization algorithms that evolve an optimum solution from a randomly initialized set of candidate solutions, modifying individuals in the solution space to perform better in the environment. An EA uses a tness evaluation function to determine the quality of an individual. Genetic operators such as recombination and mutation are used to create o spring from existing individuals. The relative quality of an individual in the population determines the individual's survival and its capability to reproduce. The initial population or the rst generation of candidate solutions are usually randomly generated, and the subsequent generations are decided by the tness of the individuals based on the tness proportional selection scheme. Fitness proportional selection assures that individuals with higher tness get to reproduce new individuals for the next generation, and that the individuals perform poorly in the environment will eventually perish. Maintaining the diversity of the population of solution space is extremely important; the diversity of the population can be maintained using genetic operators like recombination and mutation. Figure 4.1 shows the typical execution steps of an EA.

The commonly used EA in Figure 4.1 maintains a constant population size, but the size of the child population can vary based on the implementation type of the EA. Though all types of EAs guarantee a solution in the solution space, when the termination criterion is met, they do not guarantee the optimality of a solution. This may actually cause the learning algorithm to get stuck in a local optimum. Carefully choosing more suitable strategies specific to the problem domain being studied could prevent the EA from getting stuck in a local optimum.

4.2.1 Representation of an Individual

Choosing an efficient representation (the genotype) for an individual is one of the important EA design decisions. There are many ways to represent an individual, and the choice of representation is strongly dependent on the problem that we need to solve. When choosing a representation we should bear in mind that this decision determines how the fitness of the individual is evaluated and what type of genetic operators are used to create a new population for the next generation from the current population. There are three commonly used representations: bit-strings, integers and real values.

When optimizing parameters with real values, we can naturally encode the real numbers directly in the individual representation (see Table 4.1). Each of the X_i is referred to as an "allele" of the genotype. Since we evolve parameters for the force laws, we use real values restricted to an upper and a lower bound in our individual representation. We represent our force law parameters in a vector data structure. the interpretation of this structure changes based on the type of force law that is being used.

Individual	X_1	X_2	...	X_n

Table 4.1: An individual is represented as n real numbers.

4.2.2 Fitness Evaluation

Another major step in the EA execution is evaluating the fitness of an individual in a population. The fitness function quantifies the relative strength of an individual compared to other individuals in the population. This allows the EA to rank individuals and determine the individuals that are allowed to reproduce (selection probability). In one possible scheme, individuals that have higher fitness than the average fitness of the population are allowed to reproduce and contribute their genetic makeup to future generations.

The fitness evaluation also depends on the representation of the individuals. When the individual is represented as a real-valued structure, we map this representation to a fitness function that evaluates the individual's fitness to a real value as shown below.

$$F : R^n \rightarrow R \tag{4.1}$$

4.2.3 Genetic Operators

EAs use genetic operators to create offspring from an individual or several individuals in the current population. The two main genetic operators are recombination and mutation. Based on the type of EL approach, the use of these operators may vary. Genetic operators help to maintain the genetic variation of a population, thus avoiding premature convergence of that population.

There are numerous mutation techniques in the literature and they depend on

Parent	0	1	1	0	1	1	1	0	1
Offspring	0	1	0	0	1	1	1	0	1

Table 4.2: Before and after bit-flip mutation of a bit-string representation.

Parent	3	1	4	7	0	2	5	8	6
Offspring	3	1	5	7	0	2	4	8	6

Table 4.3: Before and after two-swap mutation of an integer representation.

the type of individual representation. Bit-string representations use a bit-flip method (see Table 4.2). In this method, the EA randomly picks an allele (a bit) and logically negates the value of that allele. A standard mutation rate is $P_m = 1/L$, where L is the length of the genotype.

The two-swap operator is used for permutation integer representations. The EA randomly picks two alleles and swaps their positions within the individual. This successfully restricts the duplication of any of the chosen alleles (see Table 4.3).

Gaussian mutation is typically used in real-value representations. This method adds a randomly generated real value from a Gaussian distribution with mean 0.0 and standard deviation σ to an allele chosen with probability $1/L$ (see Table 4.4).

Since we represent force law parameters with real values, we use Gaussian mutation as one of the genetic operators in our EA. An execution trace of our Gaussian mutation operator is shown in Figure 4.2.

The other widely used genetic operator is the recombination operator, also referred to as "crossover". Recombination creates one or more offspring by merging alleles

Parent	1.3	4.1	2.4	4.7	0.2	2.2	1.5	4.1	2.6
GaussianStep	1.3	4.1	2.4+0.7	4.7	0.2	2.2	1.5+0.3	4.1	2.6
Offspring	1.3	4.1	3.1	4.7	0.2	2.2	1.8	4.1	2.6

Table 4.4: Before and after Gaussian mutation of a real-value representation.

```
for all individuals in the population
    make a copy of the i^th individual
    for all alleles in the copied individual
        if ( U(0;1) < P_m )
            add N(0; ) to the j^th allele
        endif
        do boundary checking on the mutated values
    endfor
    replace i^th individual with the copy
endfor
```

Figure 4.2: Pseudocode of Gaussian mutation.

from two or several parents. The most popular recombination methods are one-point, multi-point and uniform recombination. In one-point crossover, two parents are split at a single point. Then the two allele segments are swapped to create two offspring. In multi-point crossover, the parent is split at multiple points and segments are swapped between pairs of cut points. In uniform crossover, individual alleles are swapped between two parents with a fixed probability. Our EA uses one-point crossover to produce offspring.

4.2.4 Selection

Selection determines the survival of individuals from generation to generation. Our EA uses fitness proportional selection. In fitness proportional selection, individuals are given a probability of being selected that is directly proportional to their fitness. The execution steps of the fitness proportional selection algorithm are shown in Figure 4.3. We decided to choose fitness proportional selection over other standard EA selection strategies. Our prior observations have shown that fitness proportional selection produces very desirable behavior for the obstacle avoidance task. This may

```
for all individuals in the population
    compute cumulative expected number of offspring
endfor
r = U(0; 1)
population counter = 0
while population counter is less than population size
    if ( r < cumulative expected number of offspring of individual i)
        (offspring counter of individual i) ++
        r += 1
        (population counter)++
    else
        next individual (i++)
    endif
endwhile
create offspring
```

Figure 4.3: Pseudocode of stochastic universal sampling for tness proportional selection.

be due to the fact that selection pressure depends on our evaluation function and the population dynamics as shown by Sarma (Sarma and de Jong 1998). Sarma showed that selection pressure and performance depend on the type of evaluation function and the population dynamics.

The (,) selection scheme, which is also commonly used by EA community produces a local optimum when used in our obstacle avoidance task. In (,) selection, the EA generates o spring from a population of parents (>), and from the o spring it keeps the best individuals to for the next generation.

4.3 EA for Obstacle Av oidance

Obstacle avoidance is a common task that robots need to accomplish in numerous problem domains such as search and rescue, surveillance, chemical plume tracing, terrain mapping and mine detection. Learning the optimal parameter settings needed

to accomplish the task is extremely important.

We developed a simulation tool consisting of an EA and a performance measurement metric for a swarm of robots to learn the obstacle avoidance task. We explain our performance measuremeth metric in Chapter 5.

4.3.1 Simulation Architecture

Our simulation architecture, as shown in Figure 4.4, consists of four modules: an EA for evolving the population of force laws, an environment generator, a global observer which evaluates the performance of a particular force law, and a performance measuremeth module which evaluates the quality of the optimum force law. A detailed discussion of the performance measuremeth module is provided in Chapter 5.

Figure 4.4: The architecture of the simulation tool.

Our EA is a population-based stochastic optimization algorithm inspired by natural evolution. The EA randomly initializes the initial population of force laws, and

then, it mutates and recombines the candidate solutions (individuals) based on their performance in our environment. Finally, the EA creates a population of offspring from the parent population of candidate solutions. Every individual in the population is a vector of real-valued parameters, representing an instantiation of either the Newtonian or LJ force law (depending on the force law being optimized).

The environment generator creates task environments to test the force laws. The environment consists of robots, randomly positioned obstacles, and a goal. Each force law is tested on n different environment instances created by the environment generator. Each robot carries a copy of the force law and navigates towards the goal while avoiding obstacles. Robots are given a limited amount of time to accomplish the obstacle avoidance task and reach the goal while maintaining the formation. We refer to this as an evaluation run.

$$fitness_{ind} = \frac{R_1 + R_2 + \quad + R_n}{n} \tag{4.2}$$

The global observer (fitness function) evaluates the performance of the force law in an instance of the environment and assigns a fitness value, R_i. Each evaluation run must be completed within a specific time interval, and the fitness assignment occurs at the end of the time interval which is also the end of an evaluation run. The final fitness, $fitness_{ind}$, of an individual is computed once n evaluation runs are completed.

Once the termination criteria of the EA is met, the EA outputs the optimal parameter setting for the force law that is being optimized. The termination criteria of our EA is G generations. The top level execution trace of our obstacle avoidance EA is presented in Figure 4.5

```
generate n random instances of environments
generate initial EA population
for G Generations
    for N individuals in the population
        for k environment instances
            evaluate an individual
            sum the fitness value
            next environment instance
        endfor
        compute the fitness of the individual
        next individual
    endfor
    select individuals for the next generation
    apply genetic operators
    create offspring
    next Generation
endfor
```

Figure 4.5: The top level pseudocode of our obstacle avoidance EA.

4.3.2 Obstacle Avoidance Simulation Tool

Our obstacle avoidance simulation tool in Figure 4.6 is an extension to Adam Sciambi's original version of the physicomimetics simulation tool[1]. The simulation is implemented using JAVA and runs on Linux-based machines.

The simulation tool consists of a Graphical User Interface (GUI), a training module with an EA, and a performance evaluation module. Whether it is optimizing force laws using the EA or evaluating the performance of the optimal parameter settings, the user can observe the behavior of the robots in the environment using the GUI.

Using sliders in the bottom right of the GUI, the user can change:

the number of robots: a minimum of 1 robot to a maximum of 100 robots. This sets the amount of robots that are being trained or the number of robots that

[1] http://www.cs.uwyo.edu/~wspears/ap.2D/DEMO.html

Figure 4.6: Simulated tool for obstacle avoidance task

are being tested with the trained force law. All of the robots start in the bottom left corner of the simulation world, but this could vary based on the starting radius.

the number of obstacles: a minimum of 1 obstacle to a maximum of 100 obstacles. This sets the amount of obstacles that are randomly placed in the simulation world. Initially our robots are in a tight cluster, and high potential energy (Spears, Spears, and Heil 2004) in this cluster creates an explosion when the robots start moving. This explosion causes a large number of collisions (proximity collisions), if there are obstacles near by. Also, when the robots reach the goal, they rotate and adjust the formation to its minimum potential energy state again, causing proximity collisions. To avoid proximity collisions, we do not position obstacles close to the initial position of the robots and the

goal. This provides our robots sufficient space to get into formation without colliding with obstacles.

sensor range: a minimum of 0.1R to a maximum of 19.9R. This sets the range each robot perceives in its neighborhood. Smaller sensor range causes the robots to detect a limited distance while higher sensor range allows them to detect further. Our robots start learning their environment with a sensor range of 1.5R.

starting radius: a minimum of 0.1R to a maximum of 9.9R. This sets the initial positions of each robot relative to each other. A smaller starting radius positions robots closer to each other in a cluster, and a larger starting radius positions them away from one another. The starting radius has no effect when the robots start navigating towards the goal. When the robots are navigating, their positions are decided by the forces acting on them and the desired separation distance R.

friction: a minimum value of 0 to a maximum value of 1. Friction controls the damping of the simulated robot's velocity, providing system stability. This controls the robot's acceleration, preventing robots from launching out of the simulation world. Friction is modeled by decreasing the robot's velocity by a constant multiplicative factor.

time step: a minimum of 0.1 to a maximum of 5.0. The time step determines how fast or slow the next update to the environment will happen. At the minimum setting, there are frequent updates and at the maximum setting the system is updated less frequently.

Using buttons in the bottom left of the simulation (see Figure 4.6, the user can:

\Start" button: releases robots from their initial position. The robots start moving towards the goal through the obstacles.

\Stop" button: stops the robots movement and put them back at the initial start position.

\Restart" button: stops the simulation and restarts it.

\Op en Defaults" button: opens a window for the user to choose the types of robot formation: triangular (hexagon) or square, and if the formation is in two dimensions (2D) or three dimensions (3D). The button label switches to \Close Defaults" once the window is open and closes the window when pressed again.

\Quit" button: terminates the simulation run, closes all the windows and releases the memory back to the system.

Using drop down menus, the user can:

select the time interval to update the graphic canvas where the simulation world is displayed using \Cycles/F rame". The menu consists of v e options: 1, 5, 10, 100, 500, and 1000. The option \1" updates the display canvas every time step and the option \1000" updates the canvas every 1000 time steps. The system updates the robot positions every time step regardless of updates on the canvas.

change the robot formation from 2D to 3D or 3D to 2D while the robots are in the eld.

change the robot formation from triangle (hexagon) to square or square to triangle (hexagon) while the robots are in the eld.

A User can display the connectivity of the robots in formation by clicking the \Connect" box; if two robots are in their sensor range there is a line drawn between

the two robots. Error and Noise in the simulation world are two other options that a user can choose to display.

4.4 Fitness Evaluation for Obstacle Avoidance

We carefully designed a fitness function to evaluate force law individuals. Our fitness function consists of three objectives.

maintaining formation: robots form hexagonal lattice formations and maintain the formation while navigating. A robot attracts its neighbors if the neighbors are 1.5R distance away and repulses its neighbors if the neighbors are closer than 1.5R distance.

avoiding obstacles: robots are capable of sensing the repulsive forces of obstacles from a distance of $R_o + 20$ from the center of the obstacle. R_o is the radius of an obstacle, which is 10. If the distance between the center of the robot and the center of the obstacle is less than $R_o + 1$, a collision occurs. The radius of the robot is 1.

reaching a goal: robots should reach the goal while maintaining formation and avoiding collisions with the obstacles. Robots sense the global attractive force of the goal at any distance. Robots are given a limited amount of time to reach the goal.

In our EA, we focus on designing a multi-objective fitness function to evolve optimal force law parameters. Since we do not change our objectives during optimization of Newtonian and LJ force laws, we use the same evaluation function, The biggest challenge is how to develop a compromise between the three objectives, so that the

evaluation function is capable of providing us with an optimal solution that is not biased toward any of the objectives.

4.4.1 Pareto Optimization

When there are multiple competing objectives, evolving an optimal solution is challenging. Multi-objective fitness functions are capable of providing compromising solutions in the solution space. One of the most common techniques to multi-objective optimization is to obtain a set of non-dominated solutions or a Pareto front. This method is also referred to as Pareto Optimization.

Since an EA searches a population space for an optimal solution in a parallel fashion, the EA seems amenable to Pareto optimization (Menczer, Degeratu, and Street 2000). Deb has shown the difficulties with Pareto optimization (Deb 1999). The EA does not converge on a true Pareto optimal set. Deb indicates that multimodality, deception, isolated optimum and collateral noise are reasons for this failure. Multi-modality refers to the multiple Pareto optimal fronts; deception refers to EA getting stuck in a local optima; an isolated optimum occurs when the solution space is flat; and collateral noise occurs when partially good solutions are discarded due to poor performance in another objective of the solution.

Numerous research articles have presented solutions to overcome difficulties with Pareto optimization. One technique targets a specific solution in the solution space by eliminating the need to consider the whole Pareto front by introducing a weighted fitness function. The weighted fitness function allows the target solution to get a higher fitness value, improving its probability of surviving to reproduce. The weights are decided based upon the importance of different objectives (Sbalzarini, Müller, and Koumoutsakos 2000).

4.4.2 Fitness Evaluation

We introduce a weighted fitness function with penalties to evaluate the force law individuals in our population of force laws. Fitness evaluation occurs at every time step for every individual within the permitted time limit.

$$fitness = w_1 P_{Collision} + w_2 P_{NoCohesion} + w_3 P_{NotReachGoalInPermittedTime}$$

The weighted fitness function consists of three componerts:

a penalty for collisions,

a penalty for lack of cohesion,

a penalty for robots not reaching the goal.

The fitness function uses positive penalties, and it is a minimizing function.

4.4.3 Penalty for Collisions

For all of the swarm tasks that we discussed in Chapter 1, avoiding collisions with obstacles in the task environment is important. In our simulation world, there are no safety zones around the obstacles as presented in (Balch and Hybinette 2000). The maximum sensing distance of the repulsive force on a robot from the center of an obstacle is set at $R_o + 20$. A collision occurs if the center of the robot is within the perimeter of an obstacles. We add a penalty to the fitness score if the robots collide with obstacles. All robots in the simulation world are evaluated for collisions and a penalty is added at each discrete time step within the permitted time interval.

4.4.4 Penalty for Lack of Cohesion

Maintaining a cohesive formation is another important aspect, especially during CPT and terrain mapping. With cohesive formations, robots maintain uninterrupted communication paths between one another allowing efficient distribution of resources. Additionally, this is important in mobile sensor networks where area coverage should be maximized.

The cohesion penalty is derived from the fact that in a good hexagonal lattice (as shown in Figure 3.1), interior robots should have six local neighbors at a distance of R. A penalty occurs if a robot has more or less neighbors and the value of the penalty is proportional to the error in the number of neighbors. This fitness pressure prevents the robots from forming tight clusters that may cause overlapping, or separation of the entire formation which in turn may cause the swarm to form sub-swarms when navigating through obstacles. All the robots in the simulation world are evaluated for cohesion and a penalty is added at each discreet time step within the permitted time interval.

4.4.5 Penalty for Robots not Reaching the Goal

We also introduce a penalty to our fitness function for robots not reaching the goal within a permitted time interval. In time critical search and rescue missions or defense related missions, having the swarm achieve the goal in a limited permitted time interval is extremely important. We added a penalty, if less than 80% of the robots from the initial swarm did not reach the goal within the permitted time interval. At the end of the permitted time interval, the EA evaluates the number of robots that reach the goal, and if this number is less than 80%, a penalty is added. A robot has reached the goal if the robot is within a 4R radius from the certer of the goal.

4.4.6 O ine vs. Online Learning

In our simulation world, a global observer or the tness evaluation module assigns a tness value to each force law based on its performance in the task environment. We refer to this traditional approach of optimizing parameters as \o ine learning". In static environments, individuals are trained repeatedly until a termination criteria is met and the desired behavior is achieved. The tness evaluation is done using an explicit multi-objective tness function. The force laws are trained using our obstacle avoidance environment. We present an empirical analysis of o ine learning in Chapter 5.

In real life, future scenarios such as environmental changes are unpredictable, and the robots may have to face novel situations. A robot's abilit y to react to novel situations is a part of the intelligent reaction that we expect to achieve with our robot swarms. We refer to this approach as \online learning". One of the most challenging problems in online learning is tness evaluation. Robots that learn a force law in a speci c environment are unable to adapt to another environment using the same force law. Thus, it is not feasible to use the same tness evaluation function in the new environment. We propose an implicit tness evaluation paradigm for \online learning" and present an empirical analysis in Chapter 6.

4.5 Parameter Optimization

For our robots to successfully accomplish their task, we optimize the two force laws using an EA. The Newtonian force law's parameters are di erent from the LJ force law's parameters. Our EA is not designed to optimize both force laws at the same time. Therefore, we executed our EA twice to achieve optimal parameter settings, once for the Newtonian force law and once for the LJ force law. Each force law

contains parameters for robot-robot, robot-obstacle, and robot-goal interactions.

4.5.1 Methodology

To optimize the force law parameters, we use the training module of our simulation tool. This training module allows the user to specify the type of force law, minimum and maximum parameter value bounds, the population size, the termination criteria, the mutation rate, and the crossover rate. The training module allows the user to either predefine the EA random seed or use the system time.

Our 2D simulation world is 900 × 700 in size, and contains a goal, obstacles and robots. Though we can use up to a maximum of 100 robots and 100 static obstacles with one static goal, we placed a compromise figure of 40 robots and 90 obstacles in the environment when using the training module. The goal is always placed at a random position in the right side of the world, while the robots are initialized in the bottom left area. The obstacles are randomly distributed throughout the environment, but are kept 50 units away from the initial location of the robots and the goal to avoid proximity collisions. Each circular obstacle has a radius R_o of 10, and the square shaped goal is 20 × 20. When 90 obstacles are placed in the environment, roughly 4.5% of the environment is covered by the obstacles (similar to (Balch and Hybinette 2000)). The desired separation between robots R is 50, and the maximum velocity V_{max} is 20. Figure 4.7 shows 40 robots navigating through randomly positioned obstacles. The larger circles are obstacles and the square to the right is the goal. Robots can sense other robots within a distance of 1:5R, and can sense obstacles within a distance of R_o + 20. The goal can be sensed at any distance.

The permitted time interval for the robots to reach the goal from their initial position is set at 2000 simulation time steps. This accounts for approximately 47 seconds of clock time (we use a Linux-based dual processor Dell machine with Intel

Figure 4.7: 40 robots moving to the goal. The larger circles represent obstacles, while the square in the upper right represents the goal.

Xeon 1500MHz processors). The EA was run with 100 individuals per population and was allowed to terminate after 100 generations. It takes approximately four days for our EA to achieve a parameter set that provides the desired behavior regardless of the force law that is being optimized.

4.5.2 Optimizing Newtonian Force Law

The Newtonian force law contains two parameters: gravity and the power of the force law.

$$F_{i,j} = \frac{m_i m_j G}{r^p} \tag{4.3}$$

In addition to these two parameters, we evolved the maximum force and the friction. The parameters we optimized are:

G_r - gravitational constant of robot-robot interactions,

p_r - power of the force law for robot-robot interactions,

	G	F_{max}	p	Fr
Min.	100.0	1.0	0.1	0.0
Max.	5000.0	5.0	2.0	1.0

Table 4.5: The range of values for the Newtonian parameters.

F_{max_r} - maximum force of robot-robot interactions,

G_o - gravitational constant of obstacle-robot interactions,

p_o - power of the force law for obstacle-robot interactions,

F_{max_o} - maximum force of obstacle-robot interactions,

G_g - gravitational constant of goal-robot interactions,

p_g - power of the force law for goal-robot interactions,

F_{max_g} - maximum force of goal-robot interactions,

Fr - friction in the system.

In the initial population of individuals, parameter values are randomly initialized between a predefined minimum and maximum bound. These bounds are shown in Table 4.5.

Figure 4.8 shows the evolved Newtonian robot-robot force law up to a distance of 80. A robot can sense another robot up to a distance of 1.5R, where R is 50. The force is repulsive when the distance between robots is less than 50, and it is attractive when the distance is greater than 50. The evolved F_{max_r} takes effect when the distance between robots is less than 35.

Figure 4.9 shows the evolved Newtonian robot-obstacle force law up to a distance of 80. Note that the maximum sensing distance of the repulsive force on a robot from

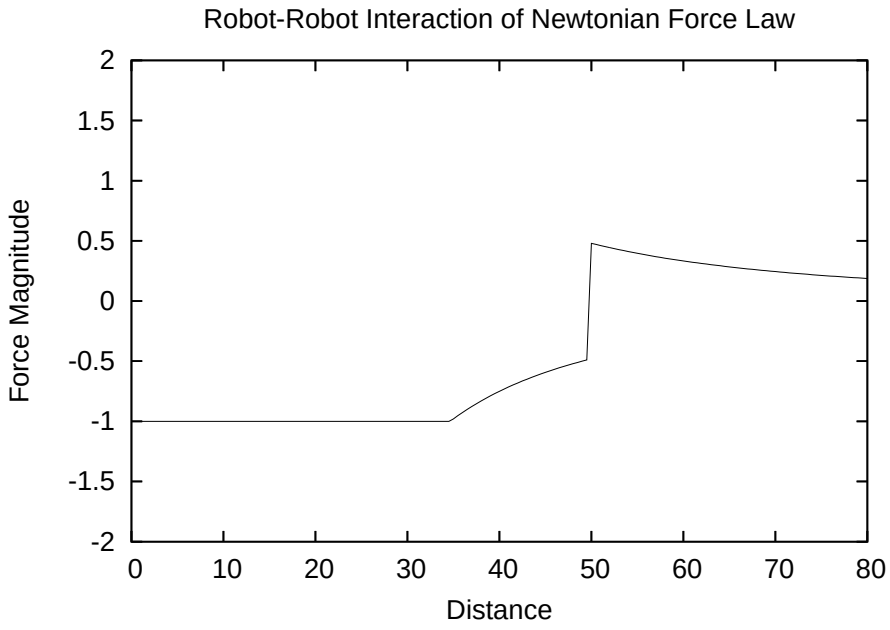

Figure 4.8: Evolved Newtonian force law for robot-robot interactions.

the certer of an obstacle is set at $R_o + 20$. At distance 0, the robot is closest to the obstacle and $F_{max\,o}$ is in effect, making the force between a robot and obstacle highly repulsive. After distance 30, the effect of the repulsive force diminishes as the robot moves away from the obstacle.

Figure 4.10 shows the evolved Newtonian robot-goal force law up to a distance of 80. A Robot can sense the goal force globally. At distance 0, the robot is closest to the goal and at distance 80, robot is further away from the goal. The evolved force law for the robot-goal interaction is constant regardless of the distance from the robot to the goal. The robots sense the maximum attractive force of $F_{max\,g}$ from the goal. It is our intention to avoid clustering of robots at the goal and our evolved robot-goal interaction is capable of avoiding this effect of the robots once they reach the goal, preserving the robot's lattice formation.

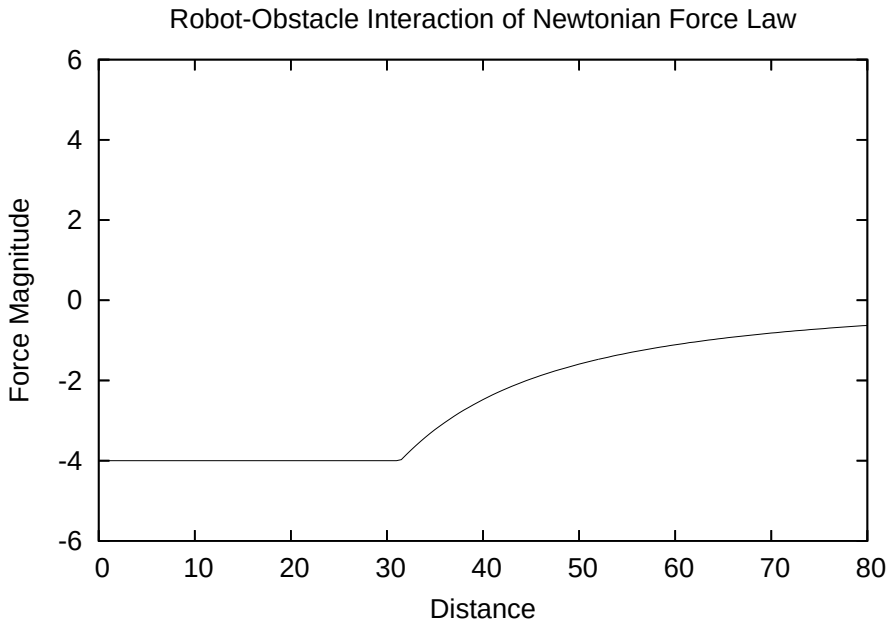

Figure 4.9: Evolved Newtonian force law for robot-obstacle interactions.

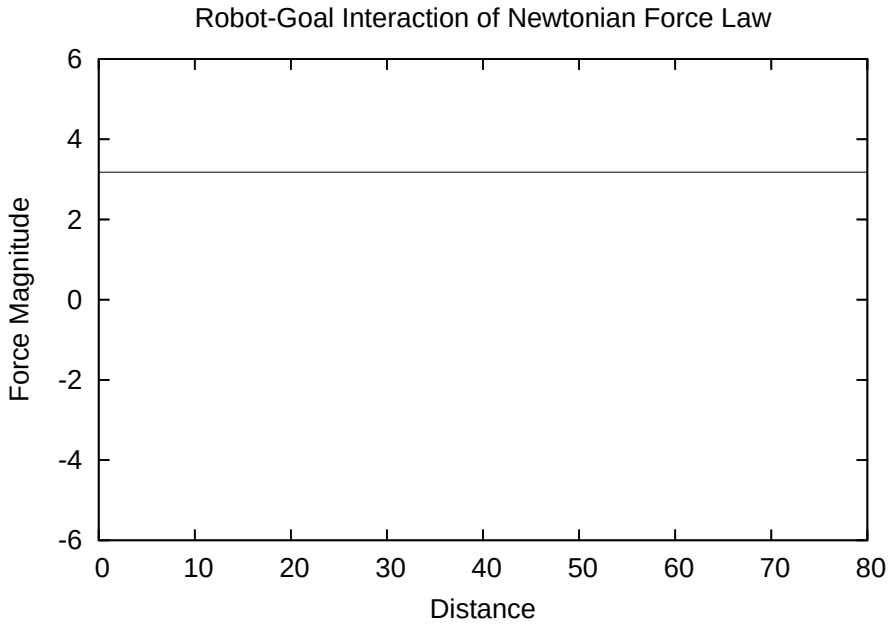

Figure 4.10: Evolved Newtonian force law for robot-goal interactions.

4.5.3 Optimizing LJ Force Law

The LJ force law contains four parameters: the strength of interaction, a non-negative attractive constant, and a non-negative repulsive constant.

$$F_{i;j} = 24 \left[\frac{2d^{12}}{r^{13}} - \frac{c^{6}}{r^{7}} \right] \tag{4.4}$$

In addition to these three parameters, we evolved the maximum force and the friction of the system. The parameters we optimized are:

$_r$ - strength of the robot-robot interactions,

c_r - non-negative attractive robot-robot parameter,

d_r - non-negative repulsive robot-robot parameter,

F_{max_r} - maximum force of robot-robot interactions,

$_o$ - strength of the obstacle-robot interactions,

c_o - non-negative attractive obstacle-robot parameter,

d_o - non-negative repulsive obstacle-robot parameter,

F_{max_o} - maximum force of obstacle-robot interactions,

$_g$ - strength of the goal-robot interactions,

c_g - non-negative attractive goal-robot parameter,

d_g - non-negative repulsive goal-robot parameter,

F_{max_g} - maximum force of goal-robot interactions,

F_r - friction in the system.

		F_{max}	c	d	F_r
Min.	1.0	1.0	1.0	1.0	0.0
Max.	20.0	5.0	10.0	10.0	1.0

Table 4.6: The range of values for the LJ parameters.

Again, in the initial population of individuals, the parameter values are randomly initialized between a predefined minimum and maximum bound. These bounds are shown in Table 4.6.

Figure 4.11 shows the evolved LJ robot-robot force law. Force is repulsive when the distance between robots is less than 50, and it is attractive when the distance is greater than 50. The evolved F_{max_r} takes effect when the distance between robots is less than 45.

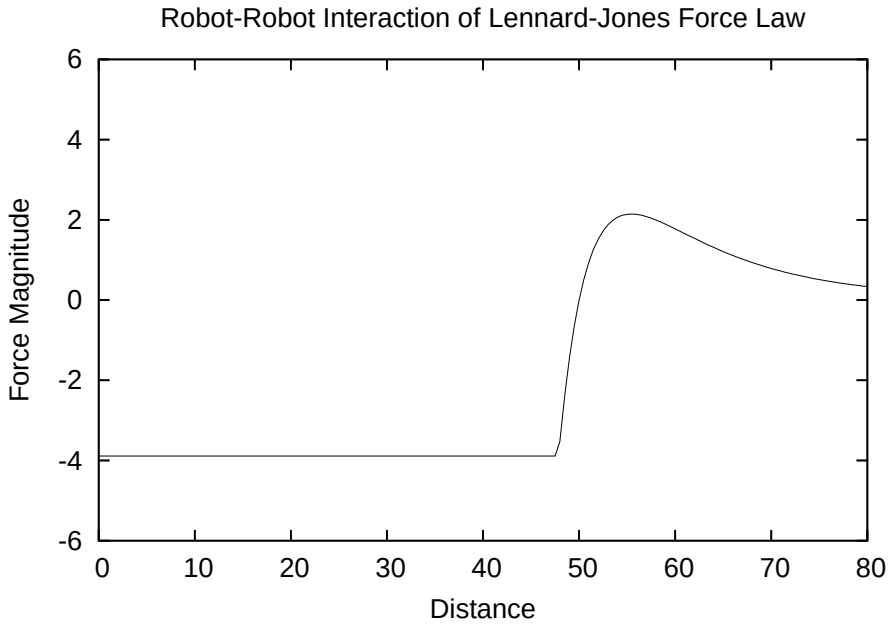

Figure 4.11: Evolved LJ force law for robot-robot interactions.

Figure 4.12 shows the evolved LJ robot-obstacle force law. The maximum sensing distance of the repulsive force on a robot from the certer of an obstacle is set at

R_o + 20. At distance 0, the robot is closest to the obstacle and F_{max_o} is in effect, making the force between a robot and obstacle highly repulsive. After distance 14, the effect of the repulsive force diminishes with the robot moving away from the obstacle.

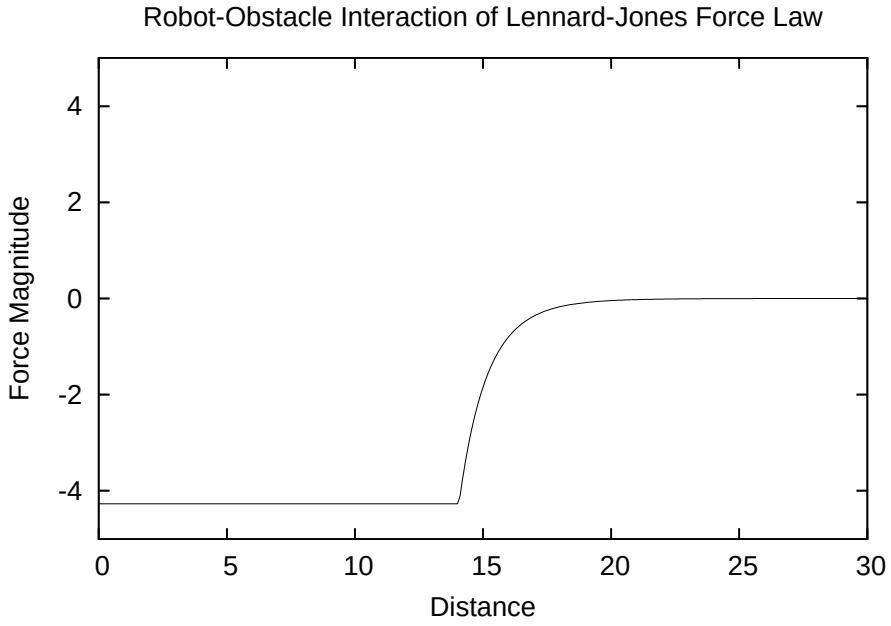

Figure 4.12: Evolved LJ force law for robot-obstacle interactions.

Figure 4.10 shows the evolved Newtonian robot-goal force law. A Robot can sense the goal force globally. At distance 0, the robot is closest to the goal and at distance 80, robot is further away from the goal. Again, the evolved force law for the robot-goal interaction is constant regardless of the distance from the robot to the goal. The robots sense the maximum attractive force of F_{max_g} from the goal. Again this evolved robot-goal interaction is capable of avoiding any clustering of the robots once they reach the goal, preserving the robot's lattice formation.

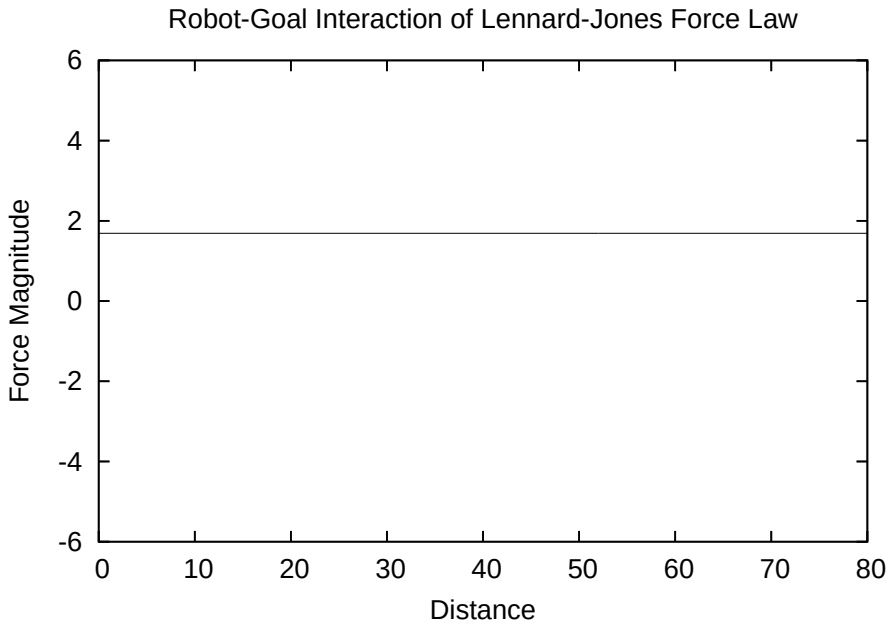

Figure 4.13: Evolved LJ force law for robot-goal interactions.

4.6 Summary

In this chapter, we first discussed the background of EL and EAs. Evolutionary Learning (EL) is motivated by Darwinian evolution and has been rapidly developing and gaining popularity as a powerful general learning approach. We use an Evolutionary Algorithm (EA) to optimize the force law parameters necessary for a swarm of robots to learn a complex environment with obstacles and a goal. Given generalized force laws, such as the Newtonian force law or the LJ force law (discussed in Chapter 3), it is necessary to optimize the force law parameters to achieve the best performance.

Then we presented a discussion on EA strategies and our motivation for selecting certain strategies. EAs are stochastic optimization algorithms that evolve an optimum solution out of a randomly initialized set of candidate solutions; slowly perturbing individuals in the solution space to perform better in the environment.

The EA uses a fitness evaluation function to determine the quality of an individual. Genetic operators such as recombination and mutation are used to create offspring from existing individuals. The relative quality of an individual in the population determines the individual's survivability and its capability of reproducing. Choosing an efficient representation (the genotype) for an individual is one of the important decisions. Our force law parameters have a real-value representation bounded by minimum and maximum limits. We use two genetic operators, Gaussian mutation and one-point crossover to produce offspring. We use a weighted fitness function with penalties to evaluate the force law performance.

Finally we introduced our simulation tool and presented the behavior of the evolved parameters. Our simulation tool consists of a Graphical User Interface (GUI), a training module with an EA and a performance evaluation module. The simulation tool allows users to select different options when setting up the environment. Our evolved parameters produced far superior results than the results presented in the literature. We present an empirical analysis of our results in Chapter 5.

Chapter 5

Offline Performance

5.1 Introduction

In Chapter 4, we presented our methodology of evolving force law parameters for the Newtonian and LJ force laws. We called this "offline learning". In this chapter, we present an empirical analysis of the two force laws. We evaluate the performance of a swarm of robots moving through an obstacle field and reaching a goal using the force laws that were optimized by the EA.

Prior research in this area has generally focused either on a small number of robots moving through a large number of obstacles, or a large number of robots moving through a small number of obstacles (Balch and Arkin 1998; Balch and Hybinette 2000). However, the more difficult task of moving a large number of robots in formation through a large number of obstacles is generally not addressed. Also, proposed metrics of performance are not complete, ignoring criteria such as the number of collisions between robots and obstacles, the distribution in time of the number of robots that reach the goal, and the connectivity of the formation as it moves. Our objective in this chapter is two-fold. The first objective is to provide a more complete set of metrics from which meaningful comparisons can be made. Second, we use these metrics, coupled with a more complete experimental methodology, to examine (a) different strategies for performing the task, and (b) trade-offs between different

criteria.

5.2 Methodology

The simulation tool consists of training and performance evaluation modules. We use the training module to evolve parameter sets for either the Newtonian or the LJ force laws. As shown in Figure 5.1, in traditional offline approaches, a simulation tool evolves force law parameters from a set of initial force laws, and the final product is the optimized force law that achieves the desired behavior.

Figure 5.1: Traditional offline approach of evolving force law rules.

In this approach, a population-based EA optimizes the force laws while the fitness function assigns a fitness value to each force law based on its performance in the environment as shown in Chapter 4.

Once the optimal force law is found, it is important to measure the quality of this force law. The performance evaluation module evaluates the optimized force laws with respect to four metrics: collisions, connectivity, reachability, and time to goal.

Again, our 2D simulation world is 900 700 in size, and contains a goal, obstacles and robots. We evaluated the performance of the optimized force laws with 20 to 100 robots, 20 to 100 obstacles, and a goal in the environment. The goal is always placed at a random position in the right side of the world, while the robots are initialized in the bottom left area. As in the training module, the obstacles are randomly distributed

throughout the environment, but are kept 50 units away from the initial location of the robots and the goal to avoid proximity collisions. Each circular obstacle has a radius R_o of 10, and the square-shaped goal is 20 20. When 100 obstacles are placed in the environment, roughly 5% of the environment is covered by the obstacles (again similar to (Balch and Hybinette 2000)). The desired separation between robots R is 50, and the maximum velocity V_{max} is 20. Robots can sense other robots within a distance of 1:5R and can sense obstacles within a distance of R_o + 20. The robots can sense the goal globally. The radius of the robot is 1.

5.3 Performance Metric

After optimization, the best force law is evaluated with our performance module. The performance module consists of four metrics and these metrics provide us with valuable measurements of the quality of the evolved force law. The force law evaluation in Chapter 4 considers three different criteria: collisions, cohesion, and time to goal. Besides these three criteria, in our performance metric we provide an additional criterion, reachability.

Collisions: our fitness function added a penalty when the robots collide with an obstacle. Thus, it is important to measure the number of robot collisions when the robots are navigating through the obstacle course using the previously evolved force law. This measurement is taken at every time step within the permitted time interval and reported as the number of robots collided within that time interval. When robots collide with obstacles we consider such robots to be damaged, but they can still move with the formation to the goal.

Swarm connectivity: One of our main objectives is to have our swarm maintains a hexagonal lattice formation until it reaches the goal. To accomplish

this objective, we added a penalty to our tness function when the force laws were trained. To evaluate the quality of the evolved force law, we measured the largest number of robots in the swarm that are connected via a communication path. The connectivity result we provide is the minimum size of the largest connected swarm as the swarm moves to the goal. Two robots are connected if their separation is 1:5R. This is an important measure of a robot swarm deployed as a sensor network, since it is extremely important to maintain communication path(s) among robots in the sensor network.

Reachability: If a considerable number of robots do not reach the goal regardless of higher swarm connectivity, then we are unable to successfully accomplish a part of our objective task. Robots not reaching the goal is an indication of lack of quality of the evolved force law. Since we believe the number of robots reaching the goal is important, we measure the percentage of robots reaching the goal at the end of the permitted time interval. We consider that a robot has reached the goal if it is within 4R distance of the goal.

Time to goal : All objectives in our o ine evaluation function depend on the amount of time permitted for the swarm to reach the goal. We set 2000 simulation time steps as the permitted time interval for the robots to reach the goal from the start position. This is also the allotted time interval during EA learning. At each time step, we measure the amount of robots reaching the goal, and if 80% of the robots are at the goal, we consider this amount of time as the \Time to goal". If this result is less than 80%, we record the \Time to goal" as 0. Though this is a strict metric of measurement, we think it is important in some critical applications where the number of robots surviving to reach the goal matters.

Figure 5.2: Performance metric; seven robots in a hexagonal lattice over the goal.

Since our objective is to provide a more complete set of metrics from which meaningful comparisons can be made, the importance of the collision, connectivity, reachability, and time to goal metrics is obvious. Although each metric provides useful information, a more complete picture arises by considering all. Figure 5.2 shows seven robots with 100% connectivity and 100% reachability at the goal.

5.4 Newtonian Experimental Results

Tables 5.1 – 5.4 show the number of collisions, connectivity, reachability, and time to goal results for the optimized Newtonian force law. A `–' entry indicates that at least 80% of the robots did not make it to the goal within the allotted time period. All experiments are averaged over 50 independent runs.

It is clear in Tables 5.1 that collisions are not a primary concern. With 100 robots and 100 obstacles, only four robots collided with obstacles and this is only a 4% collision rate which is negligible. Interestingly, the number of obstacles do not appear to be the important factor here, although the number of robots is.

With 20 and 40 robots, the connectivity remains low (see Table 5.2), but with more than 40 robots, the connectivity is amazingly high at 100%.

	Obstacles				
robots	20	40	60	80	100
20	0	0	0	0	0
40	0	0	0	0	0
60	0	0	0	2	3
80	0	0	3	3	3
100	0	2	2	4	4

Table 5.1: Number of robots that collided with obstacles using the Newtonian force law.

	Obstacles				
robots	20	40	60	80	100
20	3	5	10	8	10
40	11	17	25	21	23
60	60	60	60	60	60
80	80	80	80	80	80
100	100	100	100	100	100

Table 5.2: Minimum number of robots that remain connected using the Newtonian force law.

	Obstacles				
robots	20	40	60	80	100
20	100%	100%	95%	84%	80%
40	100%	97%	72%	55%	52%
60	0%	0%	0%	0%	0%
80	0%	0%	0%	0%	0%
100	0%	0%	0%	0%	0%

Table 5.3: Percentage of robots reaching the goal using the Newtonian force law.

When there are 20 robots, more than 80% of the robots reach the goal, but with 40 robots the number that reach the goal reduces with the increasing number of obstacles (see Table 5.3). When the number of robots is above 40, no robots reach the goal.

	Obstacles				
robots	20	40	60	80	100
20	1160	1260	1290	1530	1920
40	1680	1790	{	{	{
60	{	{	{	{	{
80	{	{	{	{	{
100	{	{	{	{	{

Table 5.4: Time taken by 80% of robots to reach the goal using the Newtonian force law.

Table 5.4 shows the amount of time taken by at least 80% of the robots to reach the goal. When there are 20 robots, more than 80% of the robots reach the goal with the time taken to reach the goal increasing with the number of obstacles. When there are 40 robots, with 20 and 40 obstacles, 80% of the robots are capable of reaching the goal within the time interval of 2000 time steps. When there are more than 40 robots, none of the robots reach the goal within the time interval of 2000 time steps.

When there are less than 40 robots, some reach the goal (Table 5.3). The time to reach the goal increases as the number of obstacles increases. However, it is clear that this is achieved by fragmenting the formation into small parts (Table 5.2). When there are more than 40 robots, none reach the goal (within the time period). Instead, the structure remains connected, but the strict rigidity of the structure prevents it from making good progress through the obstacle field. It is clear from these results that training with 40 robots does not yield a Newtonian force law that scales to a larger number of robots.

5.4.1 Solid Behavior

The Newtonian force law is effective in creating solid structures. This emergent behavior of the Newtonian force law allows us to create rigid structures with very high connectivity, but the major disadvantage is that it reduces the reachability of robots. Another disadvantage is that the Newtonian force law does not scale well to a large number of robots. Our observations show that given a longer time period, the Newtonian force law is capable of improving reachability, but this is not desirable due to time constraints in critical missions.

5.5 LJ Experimental Results

Tables 5.5 – 5.8 show the collision, connectivity, reachability, and time to goal results for the optimized LJ force law. All experiments are averaged over 50 independent runs.

	Obstacles				
robots	20	40	60	80	100
20	0	0	0	0	0
40	0	0	0	0	0
60	0	0	0	0	0
80	0	0	0	2	2
100	0	1	3	3	4

Table 5.5: Number of robots that collided with obstacles using the LJ force law.

Again, it is clear that collisions are not a primary concern. As before, the number of obstacles does not appear to be the important factor here, although the number of robots is. The differences in collision results between the LJ and the Newtonian force law are statistically insignificant.

	Obstacles				
robots	20	40	60	80	100
20	10	10	11	11	11
40	23	23	23	23	23
60	37	37	37	37	37
80	52	52	53	53	53
100	67	67	67	68	68

Table 5.6: Minimum number of robots that remain connected using the LJ force law.

Compared to the Newtonian force law, LJ connectivity remains low. Using the Newtonian force law with 60 { 100 robots, the connectivity remained 100%, but with the LJ force law the connectivity remains around 60%.

	Obstacles				
robots	20	40	60	80	100
20	100%	100%	98%	95%	95%
40	100%	100%	98%	98%	98%
60	100%	100%	98%	98%	98%
80	100%	100%	99%	98%	98%
100	100%	100%	99%	98%	98%

Table 5.7: Percentage of robots reaching the goal using the LJ force law.

Regardless of the number of robots and obstacles, the percentage of robots reaching the goal using the LJ force law remains higher than 95%. This is a signi cant improvement over the reachability results we attained using the Newtonian force law.

Comparing Table 5.8 with Table 5.4 clearly shows that the robots trained with the LJ force law reach the goal faster than the robots trained with the Newtonian force law.

Using the LJ force law, almost all of the robots make it to the goal, in all circumstances. The time to reach the goal increases slowly as the number of obstacles

	Obstacles				
robots	20	40	60	80	100
20	470	480	490	510	520
40	520	530	560	560	580
60	570	570	600	600	620
80	610	620	640	650	660
100	640	650	670	680	690

Table 5.8: Time taken by 80% of robots to reach the goal using the LJ force law.

and robots increases (with the number of robots having a larger effect). Also, swarm connectivity remains reasonably high, ranging from 50% to 68%. Interestingly, swarm connectivity increases as the number of robots increases and is almost totally unaffected by the number of obstacles. In contrast with the Newtonian force law, the LJ force law (which is trained with 40 robots) scales well with larger numbers of robots. This provides evidence that the LJ force law is a good model for the swarm behavior that we desire.

5.5.1 Fluid Behavior

Observation of the system behavior shows that the formation acts like a viscous fluid rather than a solid. Although the formation is not rigid, it does tend to retain much of the hexagonal structure. Deformations and rotations of portions of the fluid are temporary manifestations imposed by the obstacles. Hence, the added flexibility of this formation (over that achieved by the Newtonian force law) has a significant impact on behavior. The optimized LJ force law provides low collision rates, very high goal reachability rates within a reasonable period of time, and high swarm connectivity.

Figure 5.3 shows a sequence of snapshots of 50 robots navigating around a large obstacle. Robots act as a viscous fluid while avoiding the obstacle.

In the first snapshot, robots are in a fully connected sensor network and are

Figure 5.3: Fifty robots navigating around a large obstacle toward a goal. Robots maintain full connectivity while avoiding the obstacle by acting as a viscous fluid, using the LJ force law.

navigating towards the goal, but the robots have not faced the obstacle. The second snapshot shows the swarm starting to flow around the obstacle on two fronts while maintaining 100% connectivity. The third snapshot shows the robots on the two fronts merging back together. In the final snapshot, the robots are back in a cohesive formation when they have reached the goal. We observe that when the swarm reaches the obstacle, it navigates around the obstacle as a viscous fluid while maintaining 100% connectivity and provides 100% reachability. This fluid type property of the LJ force law is an emergent behavior of the swarm.

5.6 Further Analysis of Force Laws

To further analyze our system, we also collected data concerning the change in the connectivity and the percentage of robots reaching the goal over time. The resulting graphs are far too numerous to present here, but we present representative examples. A complete set of reachability results can be found in Appendix I and a complete set of connectivity results can be found in Appendix II. All graphs are averaged over 50 independent runs.

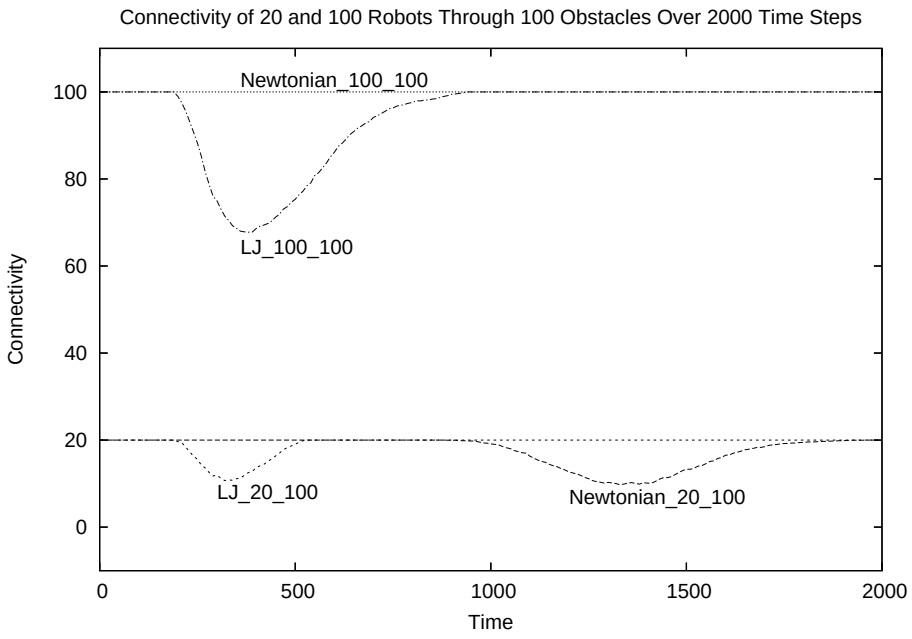

Figure 5.4: Change in connectivity over 2000 time steps for 20 and 100 robots through 100 obstacles using the Newtonian and LJ force laws.

Figure 5.4 illustrates the change in connectivity of the swarm over time. Two sets of results are presented in this graph. The curves at the top are for 100 robots moving through 100 obstacles. The robots controlled by the Newtonian force law remain fully connected (although, as we know from the prior results, this is because the formation has not succeeded in reaching the goal). However, the swarm connectivity for the LJ-

controlled robots drops after 200 time steps as the formation begins to move through the obstacle field. After 400 time steps, the formation connectivity increases as the robots reach the goal.

The curves at the bottom are for 20 robots moving through 100 obstacles. In this situation the Newtonian-controlled robots arrive at the goal, and the swarm connectivity drops after 800 time steps and increases after roughly 1300 steps. Because the LJ-controlled formation moves much more quickly, the formation connectivity drops after 200 time steps and increases after roughly 300 steps. It is interesting to note that the LJ-controlled swarm does not break apart as much as the Newtonian-controlled swarm.

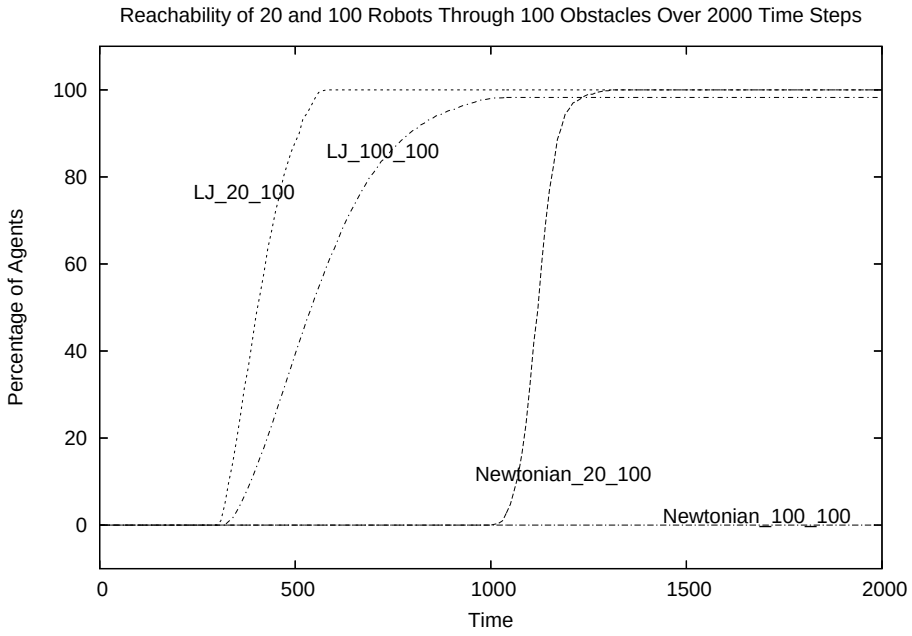

Figure 5.5: Percentage of 20 and 100 robots reaching goal through 100 obstacles over 2000 time steps using the Newtonian and LJ force laws.

Figure 5.5 shows how the number of robots reaching the goal changes with time. Again, two sets of results are presented, for 20 and 100 robots moving through 100

obstacles. The two left-most curves are for the LJ-controlled robots. Note, regardless of the number of obstacles, robots start to arrive at the goal at roughly the same time (300 time steps). With 20 robots, they all have arrived at the goal by approximately 500 time steps. This indicates that all robots arrived at the goal within a 200 time step interval { a relatively narrow band in time. Increasing the number of robots to 100 increases the time interval to only 500 steps.

The other two curves are for the Newtonian-controlled robots. With 20 robots, they start to reach the goal at 1000 time steps, and the interval is approximately 200 time steps. When there are 100 robots, none reach the goal within the allotted time period.

5.7 Safety Zone

The concept of a \safety" zone was introduced in (Balch and Hybinette 2000). The static obstacles are modeled with three layers of repulsive potentials as shown in the Figure 5.6. When the robot is beyond the sphere of influence (S), no repulsion is generated. Within the sphere of influence, repulsion increases linearly until the robot reaches the safety margin. When the robot is within the safety margin (M), the magnitude of repulsion is 1 .

Our physicomimetics framework does not assume the existence of a safety zone around obstacles. The force laws evolved with the EA produce behavior where the robots skirt the obstacles as closely as possible. This is consistent with the general physicomimetics framework, where robots move in a fashion that minimizes energy usage. However, as noted above, this can lead to collisions. In this section we examine the trade-offs induced by the addition of a safety zone. Our EA did not learn the force law with a safety zone around the obstacles. This experiment measures the

Figure 5.6: Three layered safety zone. The obstacle is represented as a black circle in the middle, and r is the distance from a robot to the center of the obstacle.

quality of the evolved force law when there is a safety zone around the obstacles.

We performed the same experiments as before, for 20 and 100 robots with varying number of obstacles. All obstacles were given a safety zone of size 5. Hence, robots can sense obstacles within a distance of $R_o + 25$, where R_o is 10. Again, the robots were allowed 2000 time units to reach the goal. If the distance to a robot from the center of the obstacle is less than 20, the robot senses the maximum repulsive force, F_{max}. When this distance is greater than 20, the robots sense the normal force as with the previous method without the safety zone. Using the safety zone, we extended the range at which obstacles are sensed and also strengthen the repulsion near the obstacle. A complete set of results with safety zone for both Newtonian and LJ force laws can be found in Appendix III.

5.7.1 Newtonian Experimental Results

The introduction of the safety zone eliminated all collisions of robots with obstacles, and the swarm connectivity results were similar. However, reachability was greatly reduced and the time to reach the goal increased (Tables 5.9 and 5.10).

	Obstacles				
robots	20	40	60	80	100
20	96%	3%	1%	0%	0%
100	0%	0%	0%	0%	0%

Table 5.9: Percentage of robots reaching the goal using Newtonian force law and safety zone.

	Obstacles				
robots	20	40	60	80	100
20	1850	{	{	{	{
100	{	{	{	{	{

Table 5.10: Time taken by 80% of robots to reach the goal using Newtonian force law and safety zone.

The results were not unexpected. Since the Newtonian force law produces a structure that acts like a solid, the addition of the safety zone makes it more difficult for the formation to rotate and counter-rotate (an emergent property of the system) through the obstacles.

5.7.2 LJ Experimental Results

As with the Newtonian force law, the introduction of the safety zone eliminated all collisions of LJ-controlled robots with obstacles, and the swarm connectivity results were similar.

	Obstacles				
robots	20	40	60	80	100
20	100%	100%	73%	60%	63%
100	100%	99%	90%	86%	80%

Table 5.11: Percentage of robots reaching the goal using LJ force law and safety zone.

	Obstacles				
robots	20	40	60	80	100
20	610	660	{	{	{
100	820	900	1050	1200	1740

Table 5.12: Time taken by 80% of robots to reach the goal using LJ force law and safety zone.

Once again, reachability was reduced and the time to reach the goal increased. However, the reduction in performance (see Tables 5.11 and 5.12) is not nearly as severe as with the Newtonian-controlled robots. The additional exibility of the viscous uid works far better.

5.7.3 Further Analysis of Safety Zone

Figures 5.7 and 5.8 show the results of the same experimert, but with the addition of the safety zones around all obstacles. As noted earlier, safety zones remove all collisions, but the impact on reachability is clear. Even with only 20 robots, the performance with the Newtonian force law is severely impacted. The performance of the LJ-controlled robots is also impacted but to a lesser extent. The time interval that the robots arrive at the goal remains relatively una ected except for the "LJ_100_100" (more than 80% reach the goal within allotted time limit of 2000). The number of robots reaching the goal is de nitely compromised. The swarm connectivity remains quite similar to earlier connectivity results with no safety zones.

5.8 Summary

This chapter presented a novel extension to our physicomimetics framework, with the use of a generalized Lennard-Jones force law. We then summarized how we tested the force laws within the context of moving robotic swarm formations through obstacle

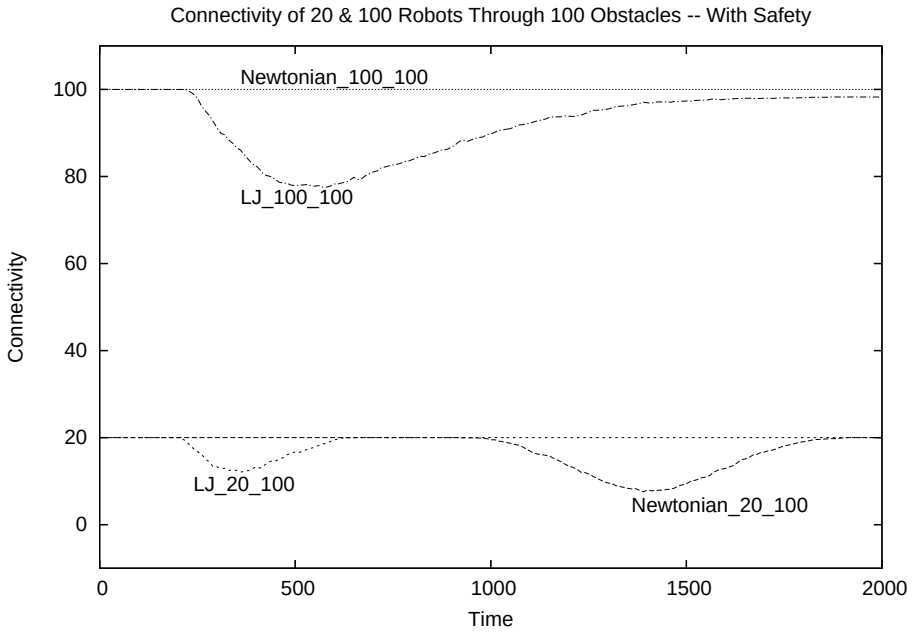

Figure 5.7: Change in connectivity over 2000 time steps for 20 and 100 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone around obstacles.

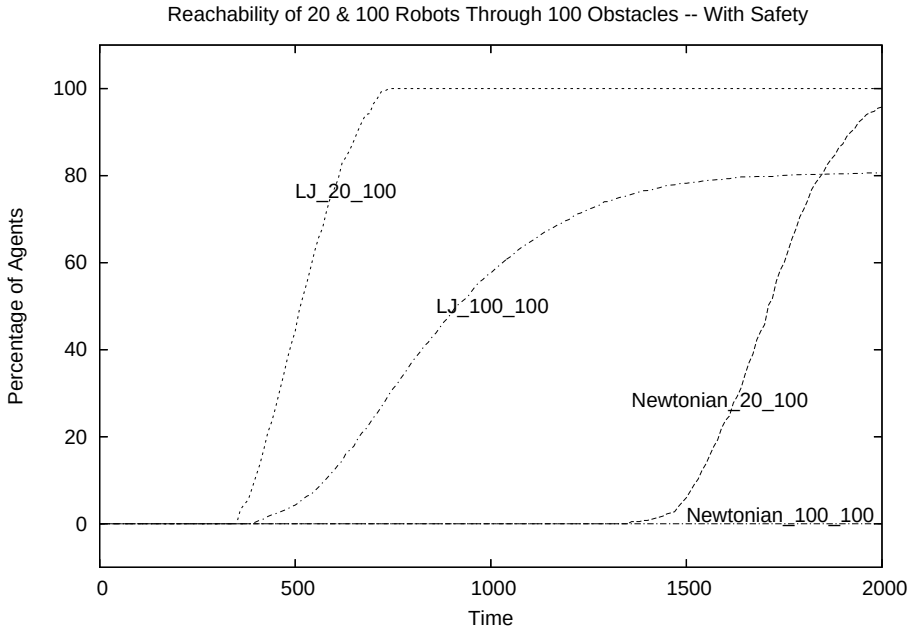

Figure 5.8: Percentage of 20 and 100 robots reaching goal through 100 obstacles over 2000 time steps using Newtonian and LJ force laws with a safety zone around obstacles.

elds to a goal.

In addition, we presented novel metrics of performance, namely, the number of robots that collide with obstacles, their connectivity, the number of robots that reach the goal, and the time taken by at least 80% of the robots to reach the goal. Although each metric provides useful information, a much better picture arises by considering all metrics. Our empirical analysis is methodical, ranging from 20 to 100 robots, and ranging from 20 to 100 obstacles.

Our results indicate that the LJ-controlled robots have far superior performance to our Newtonian-controlled robots. This is because the emergent behavior of the LJ-controlled swarm is to act as a viscous uid, generally retaining good connectivity while allowing for the deformations necessary to smoothly o w through the obstacle eld. Despite being trained with only 40 robots, the emergent behavior scales well to larger numbers of robots. In contrast, the Newtonian-controlled swarm produces more rigid structures that have more di cult y maneuvering through the obstacles. Furthermore, performance drops dramatically when there are more than 40 robots. Table 5.13 summarizes the results.

	Newtonian				LJ			
Robots	40	60	80	100	40	60	80	100
Collisions	0	3	3	4	0	0	2	4
Connectivity	23	60	80	100	23	37	53	68
Reachability%	52	0	0	0	98	98	98	98
Time to Goal by 80%	{	{	{	{	580	620	660	690

Table 5.13: Summary of results for 40 { 100 robots, with 100 obstacles.

Finally, we used the metrics to consider the trade-o s that occur when a safety zone is intro duced around the obstacles. As expected, collisions never occur, but signi cant reductions in reachability arise. The connectivity of the swarm is similar

to the results seen without the safety zone.

Chapter 6

Online Learning

6.1 Introduction

Swarm engineering is difficult due to numerous constraints, such as noise, limited range of interaction with other robots, delayed feedback, and the distributed autonomy of the robots. One potential solution is to automate the design of multi-robot swarms in simulation as we have done in Chapter 4, using evolutionary algorithms (EAs) (Grefenstette 1989; Wu, Schultz, and Agah 1999). In this paradigm, the EA evolves the behaviors of the agents (their local interactions) such that the global task behavior emerges. A global observer monitors the collective and provides a measure of performance to the individual agents. Agent behaviors that lead to desirable global behavior are hence rewarded, and the collective system is gradually evolved to provide optimal global performance.

There are several difficulties with this offline approach. First, a global observer may not exist. Second, some (but not all) agents may experience some form of reward for achieving task behavior, while others do not. Third, this reward may be delayed or may be noisy. Fourth, the above paradigm works well in simulation (offline) but is not feasible for real-world online applications where unexpected events occur. Finally, the above paradigm may have difficulty evolving different individual behaviors for different agents (heterogeneity vs homogeneity).

6.2 Constraints With Offline Learning

Offline learning in simulation provides the capability of controlling robot parameters as well as the environment parameters. These capabilities do not exist when the robots are in the field, and the robot swarm must quickly learn and adapt to a new environment. If the robots learn their aggregate behavior in simulation and they are introduced to a new environment, it is certain that the robots will not successfully accomplish the task due to several constraints such as:

non-existence of a global observer: in simulation, the global observer assigns a fitness value to each individual based on its performance in the environment. The environment is mostly static in nature and the robots evolve an aggregate behavior to accomplish a task in that specific environment. When the robots are introduced to a new environment with the rules they learned in simulation, these rules are insufficient. If a global observer does not exist in the new environment to provide performance feedback, robots will not accomplish the task successfully

noise: noise in the system could be introduced by several sources such as sensors, non-deterministic action by the robots, and environmental changes. Filtering out noise in real world is extremely difficult and challenging. Noisy environments may cause the robots to delay their response, and this may even jeopardize time critical missions.

only a subset of the robots receive a reward and reward may be delayed: noise and discretization of robot sensors and effectors may cause loss of information. This could either delay the reward to the robots or provide the reward to only a subset of robots. Lack of reward or delayed reward could further degrade the

swarm performance.

These constraints in online learning approaches impede the continued evolution of aggregate swarm behavior in changing environments. Thus, it is absolutely necessary to design new paradigms for swarm online learning. Our objective is to design a conceptual paradigm for swarm online learning that can be used regardless of the techniques used. This online learning paradigm has the ability to allow robots to learn and adapt to unexpected scenarios in new environments. We propose a new distributed online learning paradigm for robot swarms; called "Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios" or DAEDALUS.

6.3 Distributed Agent Evolution with Dynamic Adaptation to Local Unexpected Scenarios - DAEDALUS

With the DAEDALUS paradigm, we assume that agents (whether software or hardware) move throughout some environment. As they move, they interact with other agents. These agents may be of the same species or of some other species (Spears 1994). Agents of different species have different roles in the environment. The goal is to evolve agent behaviors and interactions between agents, in a distributed fashion, such that the desired global behavior occurs[1].

Let us further assume that each agent has some procedure to control its own actions in response to environmental conditions and interactions with other agents. The precise implementation of these procedures is not relevant; thus they may be programs, rule sets, finite state machines, real-valued vectors, force laws, or any other procedural representation. Agents have a sense of self-worth, or "fitness". Agents that experience direct performance rewards have higher fitness. Other agents may

[1] The work by (Watson, Ficici, and Pollack 2002) is conceptually similar and was developed independently.

not experience any direct reward but may in fact have contributed to the agents that did receive direct reward. This \credit assignment" problem can be addressed in numerous ways, including the \bucket brigade" algorithm or the \profit sharing" algorithm (Grefenstette 1988). Assuming that a set A of agents has received some direct reward, both algorithms provide reward to the set B of agents that have interacted (and helped) those in A. Further trickle-back rewards are also given to those agents in set C that helped those in B, and so on. Agents that receive no rewards lose fitness. If fitness is low enough, agents stop moving or die.

6.3.1 Distributed Evolution

Evolution in multi-robot systems is challenging in both a conceptual and applied perspective. Evolving the aggregate behavior of the collective through deliberations with neighbors and with limited sensory capabilities is much more difficult. These deliberations among robots may occur at different stages of the evolution under different constraints, but preserving the aggregate behavior and accomplishing a task is of vital importance to us. Our focus is to provide our robots with intelligent capabilities so that the robots execute these capabilities in distributed fashion.

In our DAEDALUS paradigm, evolution occurs when individuals of the same species interact. Those robots with high fitness give their procedures to agents with lower fitness. Evolutionary recombination and mutation provide necessary perturbations to these procedures, providing increasing performance and the ability to respond to environmental changes. Different species may evolve different procedures, reflecting the different niches they fill in the environment.

6.3.2 DAEDALUS for Obstacle Avoidance

Each robot of the swarm is an individual in a population that interacts with its neighbors. Each robot contains a slightly mutated copy of the optimized force law rule set found with offline learning. This ensures that our robots are not completely homogeneous. We allowed this slight heterogeneity because when the environment changes some mutations perform better than others. The robots that perform well in the environment will have higher fitness than the robots that perform poorly. When low fitness robots encounter high fitness robots, the low fitness robots ask for the high fitness robot's rules. Hence, better performing robots share their knowledge with their poorer performing neighbors.

When we apply DAEDALUS to obstacle avoidance, we focus on two aspects of our swarm: reducing obstacle-robot collisions and maintaining the cohesion of the swarm. Robots are penalized if they collide with obstacles and/or if they leave their neighbors behind. The second scenario arises when the robots are left behind in cul-de-sacs. This causes the cohesion of the formation to be reduced.

Due to the superiority of the LJ force law over the Newtonian force law, we decided to use the LJ force law to test our online learning paradigm.

6.4 Transition from Offline to Online Learning

Our prior applications of EAs to design multi-agent systems have used the offline approach – a global observer assigns fitness to agents based on their collective behavior. In the next section, we show how DAEDALUS can be applied to the obstacle avoidance task in an online environment. In this application, recombination and mutation operators provide the ability to respond to environmental changes (which can include the addition and/or removal of agents).

Figure 6.1: A long corridor with randomly placed obstacles and five goals.

6.4.1 Methodology

Each robot of the swarm contains a slightly mutated (1% mutation rate) copy of the optimized LJ force law rule set found with offline learning. Again the force law rules are mutated with Gaussian mutation (see Figure 4.2). All the robots have the same initial fitness or "worthiness" of 1000 at the start. This fitness value does not correlate with any other system parameters. There are five goals to achieve in a long corridor, and between each randomly positioned goal is a different obstacle course with a total of 90 randomly positioned obstacles. The online 2D world is 1600 × 950, which is larger than the offline world. In our changed environment, each obstacle has a radius of 30 compared to the offline obstacle radius of 10. So more than 16% of the online environment is covered with the obstacles. Compared to the offline environment, the online environment triples the obstacle coverage. We also increase the maximum velocity of the robots to 30 units/sec, allowing the robots to move 1.5 times faster than in the offline environment. The LJ force law learned in offline mode is not sufficient for this more difficult environment, producing collisions with obstacles (due to the higher velocity) and robots that never reach the goal (due to the high percentage of obstacles). The remaining system settings are kept the same as with the offline methodology. Figure 6.2 shows an example of the more difficult environment.

Robots that are left behind (due to obstacle cul-de-sacs) do not proceed to the

Figure 6.2: 60 robots moving to the goal. The larger circles represent obstacles, while the square in the upper right represents the goal. The larger obstacles make this environment far more difficult for the robots to traverse.

next goal, but the robots that had collisions and made it to the goal are allowed to proceed to the next goal. We assume that damaged robots can be repaired once they reach a goal.

6.4.2 Collision Avoidance

To measure the performance of the DAEDALUS approach, an experiment is carried out with 60 robots, 5 goals in the long corridor, and 90 obstacles in between each goal. Our focus in this experiment is to test the ability of DAEDALUS to learn to avoid robot collisions. The experiment was averaged over 100 runs with different robot, goal, and obstacle placements. Each robot is given equal initial fitness and "seeded" with a mutated copy of the optimized LJ force law learned in offline mode. If a robot collides with an obstacle, its fitness is reduced. Whenever a robot encounters another robot with higher fitness, it takes the relevant parameters pertaining to the obstacle-robot interaction of the better performing robot.

6.4.3 Experimental Results

Figure 6.3 shows the ratio of the number of robots that collided with obstacles versus the number of robots that survived to reach the goals. The graph indicates that after only 4 goals, the percentage of robots that collide with obstacles has dropped from about 38% to less than 8%. Inspection of the obstacle-robot parameters indicates that the repulsive component increased through the online process of mutation and the copying of superior force laws.

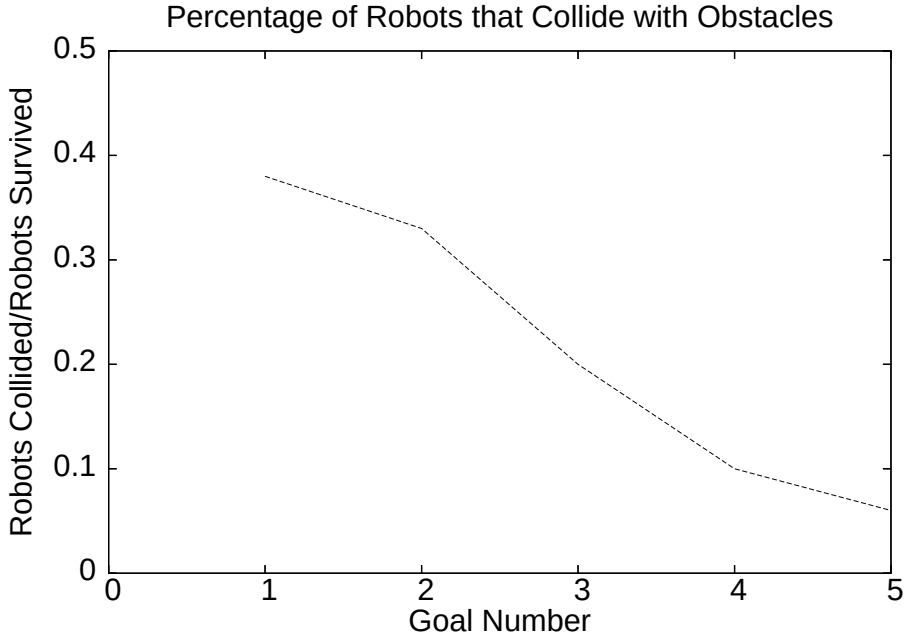

Figure 6.3: The ratio of colliding robots versus the number of surviving robots for 60 robots moving through 5 goals with 90 obstacles in between each goal.

It is apparent that robots do not have difficulty learning to avoid obstacles using the LJ force law. Online learning improves the robot's ability to adapt in the new environment and to avoid large obstacles. The slightly mutated force law rules which are learned offline provided the robots the ability to learn and adapt to the online environment more quickly.

6.5 Survivability

Survivability of our robots is extremely important for several reasons. First, if too many robots die while they are in the field, accomplishing a task becomes difficult. Second, this could affect the diversity of the swarm. Diversity is important since the robots share their genetic makeup to improve their performance. Thus, we are motivated to improve the survivability of the robots in the swarm.

Our first experiment (with collision avoidance) did not attempt to alleviate the situation where robots are left behind; in fact, only roughly 48% of the original 60 robots reach the final goal (see Figure 6.4, lower line). This is caused by the large number of cul-de-sacs produced by the large obstacle density. Our second experiment attempts to alleviate this problem by focusing on the robot-robot interactions. Our assumption is that the LJ force law needs to provide stronger cohesion, so robots aren't left behind.

6.5.1 Methodology

If robots are stuck in cul-de-sacs (i.e. they make no progress towards the goal) and they sense neighbors, they slightly mutate (1% mutation rate) the robot-robot interaction parameters of their force laws. In a situation in which they do not sense the presence of neighbors and do not progress towards the goal, they rapidly mutate (5 % mutation rate) their robot-goal interaction causing a \panic behavior". These relatively large perturbations of the force law allow the robots to escape their motionless state.

6.5.2 Experimental Results

Figure 6.4 shows the results of this second experiment. In comparison with the first experiment (with survival rates of 48%), the survival rates have increased to 63%. As a control experiment, we ran our offline approach on this more difficult task. After five goals, the survival rate is about 78%. Recall that the offline results are obtained by running an EA with a population size of 100 for 100 generations with each individual averaged over 50 random instantiations of the environment. As can be seen, the DAEDALUS approach provides results somewhat inferior to the offline approach, in real time, while the robots are in the environment.

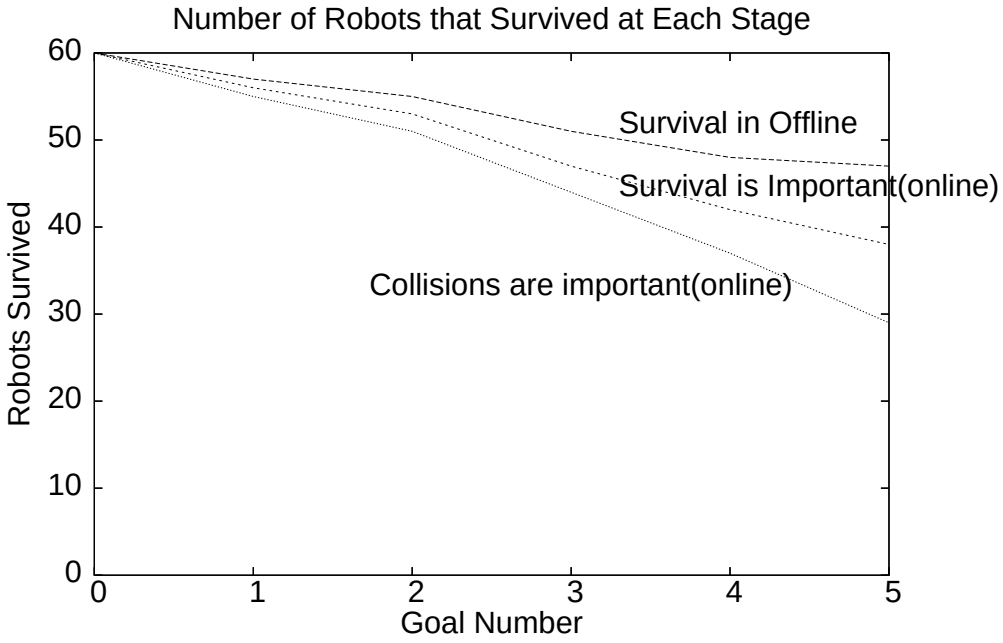

Figure 6.4: A comparison of (a) the number of robots that survive when rules are learned using offline learning, (b) the number of robots that survive when using online learning (where the focus is on reducing collisions), and (c) the number of robots that survive when using online learning (and the focus is on survivability).

Although not shown in the graph, it is important to point out that the collision rates were not affected in the second experiment. Hence, we believe that it is quite

feasible to combine both aspects in the future. Collision avoidance can be improved via mutation of the obstacle-robot interaction, while the survival rate can be improved via mutation of the robot-robot interaction and robot-goal interaction.

6.5.3 Difficulty of Survival

Surviving in one environment with the force laws learned in another environment is difficult. In the offline simulation the robots did not learn to avoid cul-de-sacs; the obstacle density did not produce cul-de-sacs, leaving sufficient space for the robots to navigate through. In the online environment robots are not capable of completely avoiding cul-de-sacs, so they get stuck behind these cul-de-sacs. Though we are able to improve the robot survival via mutation, we were unable to achieve the survival of all of the robots that are initially fielded. Our intention is to maintain the survival at a healthy level so that our swarm still maintains diversity.

6.6 Summary

Traditional approaches to designing multi-agent systems are offline, and assume the presence of a global observer. However, this approach will not work in real-time online systems. This chapter presented a novel approach to solving this problem, called DAEDALUS, where we showed how concepts from population genetics can be used with swarms of agents to provide fast online adaptive learning in changing environments.

Our obstacle avoidance case study is used in this chapter to illustrate the feasibility of this approach. We tested our robots in a long corridor with five goals placed among randomly initialized obstacles. We initialized our robots with slightly mutated copies of the LJ force law optimized using the offline approach. Our first experiment

attempted to alleviate the increased number of collisions due to the size of obstacles in the new environment. We presented results showing that robots were capable of quickly adapting to the new environments (by learning how to avoid obstacles) by sharing and mutating their force laws.

Though we were able to alleviate the number of collisions, this did not help robot survival in the new environment, due to a large amount of cul-de-sacs. We presented our results for collision reduction and online survivability and compared this results to the results of robots trained in the o ine environment.

Chapter 7

Obstructed Perception

7.1 Introduction

When robots are in complex environments and are required to interact with their neighbors, it is important for the robots to have adequate sensor capabilities. One possible problem is that even the most sophisticated sensors may not guarantee satisfactory interactions among the robots due to obstacles blocking the robot's perception of the search space. We refer to this scenario as obstructed perception.

This is quite similar to the idea of partially observable domains. In partially observable domains, robots cannot continually monitor neighbors and model their world due to the computational burden of such monitoring and modeling. The lack of such monitoring and modeling leads to increased uncertainty about the state of other agents (Varakantham, Maheswaran, and Tambe 2004). This partial observability can introduce uncertainty in agent state information, causing degradation of robot performance. We refer to this as obstructed perception. Obstructed perception follows a strict interpretation of sensor failure.

Robots face increasing amounts of sensor obstruction when they act in environments with large obstacle density. This causes the robots to either observe their environment partially or not at all. Our obstructed perception method follows the latter. Another reason for partial observation is limited sensor distance of our robots.

Sensors with limited distance can reduce the robot's interactions with other robots. A decrease in interactions among swarm members causes a reduction in population diversity making it more difficult for the swarm to improve task efficiency.

We use our DAEDALUS paradigm to improve swarm efficiency in accomplishing a task with obstructed perception. DAEDALUS is designed to improve swarm performance when the robots are in complex challenging environments. Our objective is to apply the DAEDALUS paradigm to improve robot survivability and reduce collisions in high density obstacle environments where the robots have their perception obstructed.

7.2 Learning Dynamic Environments with Obstructed Perception

When a robot can not see another robot, due to the presence of obstacles, we call this "obstructed perception." When the robot's line of sight lies along an edge of an obstacle, the robots are capable of sensing each other. Surprisingly, this is not generally modeled in prior work in this area (Balch and Hybinette 2000). Figure 7.1 shows an example scenario of obstructed perception. The larger circle represents an obstacle, and A and B represent robots. We define minD to be the minimum distance from the certer of the obstacle to the line of sight of robot A and robot B, and r is the radius of an obstacle. If $r > minD$, robot A and robot B have their perception obstructed.

We utilize a parameterized description of a line segment (Haeck 2002) to find the minD.

$$term_1 = (((1 - q) * X_a + q * X_b) - X_c)^2$$

$$term_2 = (((1 - q) * Y_a + q * Y_b) - Y_c)^2$$

Figure 7.1: The sensing capability of two robots (A; B) is obstructed by a large obstacle(C).

$$\text{minD} = \sqrt[q]{[\text{term}_1 + \text{term}_2]} \tag{7.1}$$

where X_a, X_b are the x positions of robots A and B, Y_a, Y_b are the y positions of robots A and B, X_c and Y_c are the x and y positions of the center of an obstacle, and q is the minimum function that is defined by

$$\frac{((X_c - X_a)(X_b - X_a) + (Y_c - Y_a)(Y_b - Y_a))}{(X_b - X_a)^2 + (Y_b - Y_a)^2} \tag{7.2}$$

7.3 Diversity in Swarms

Diversity refers to the non-uniformity or the variation in species. In Darwin's theory of evolution, variation is one of the principle factors that contribute to generate diverse species. The concept of diversity is fundamental in disciplines such as biology,

ecology sociology, and genetics, but there are many issues that remain unresolved when the concept of diversity is applied in multi-agent and swarm robotics research. Addressing diversity in a swarm is important; the lack of diversity or a degradation of diversity leads to tness stagnation of the swarm, causing reduced performance.

Designing and developing algorithms for diverse populations of a swarm of robots that perform well in complex environments is challenging. Addressing this issue is much more di cult if the robots are in complex dynamic environments rather than computer simulated static environments. Emergent aggregate behavior of a diverse population of robots is still desirable, and providing behavioral assurance and measuring the swarm performance is not trivial. Categorizing robots into groups and measuring their emergent properties is one way to overcome the challenges.

The robots in a swarm can be categorized into groups based on their di erences. We could relate these di erences to the robot's physical structure, algorithmic differences, mechanical properties, skills, behavioral aspects and possibly many more. These types of classi cations do not provide a guarantee that a robot belongs to an exact category due to the complex nature of a swarm that can react and behave distinctly. Though these classi cations of swarms are popular, traditionally swarms have been classi ed as either heterogeneous or homogeneous depending on the di erences previously mentioned. One of the most important units of classi cation of robots into either heterogeneous or homogeneous groups is the metric used for assessing the swarm performance.

We present a classi cation of robots in a swarm based on their genetic makeup (the force law). We classify our robots using their mutation rate. Each robot is assigned a prede ned mutation rate and the robot mutates its copy of the force law based on the circumstances it faces in the environment. As an example, a robot may mutate its force law with the assigned mutation rate if the robot is stuck behind a cul-

de-sac, but at the same time another robot may completely avoid the same cul-de-sac without any mutation due to the higher quality of its force law. In these scenarios, our internal performance metric (the robot's worthiness) decides the robot's survival.

7.3.1 Homogeneous Swarms

A homogeneous swarm consists of robots with identical software and hardware capabilities. These capabilities may vary from one swarm system to another swarm system, but the capabilities among robots within the same swarm remain identical. According to Li homogeneous systems represent a special case of heterogeneous systems (Li, Martinoli, and Abu-Mostafa 2003). Depending on the environment and task constraints, a homogeneous solution may not be a system that achieves the best results. We have presented our o ine results with homogeneous swarms in Chapter 5. Our homogeneous swarm was capable of producing superior results in an environment similar to the one that it learned, but given di erent environmental constraints the swarm fails to achieve the desired behavior (see Figure 6.4).

7.3.2 Heterogeneous Swarms

A heterogeneous swarm consists of robots that are physically di erent or have di erent software or hardware capabilities. Our robot swarm in the online learning environment is a heterogeneous swarm. We slightly mutated our LJ force law and created a heterogeneous swarm in Chapter 6. We use this heterogeneous swarm to test the feasibility of our DAEDALUS paradigm in a complex dynamic environment. In this chapter, we apply our DAEDALUS paradigm to online learning with heterogeneous swarms that are capable of exchanging their mutation rates. We also increased the task constraints by introducing obstructed perception to our heterogeneous swarm.

7.4 Swarm Learning Methodology with Obstructed Perception

To deal with obstacle avoidance, we have separate force laws for the robot-robot interactions, robot-goal interactions, and robot-obstacle interactions. Hence , c, d, and F_{max} must be optimized for all three forms of interactions, resulting in 12 parameters. Robot-robot and robot-obstacle interactions are local (i.e., robots can only sense nearby robots and obstacles). The robots are trained with an o ine EA in an o ine environment. The environment is 900 700 with 90 randomly positioned obstacles, each of radius 10. This yields about 5% obstacle coverage, which is typical of most studies in this area (Balch and Hybinette 2000). The robots move with maximum velocity 20 units/sec. The EA does not have great di cult y producing an optimized LJ force law that avoids obstacles while allowing all robots to reach the goal.

However, the online environment is far more di cult. The online 2D world is 1600 950, and each of the 90 obstacles has a radius of 30 compared to the o ine obstacle radius of 10. Therefore, more than 16% of the online environment is covered with the obstacles, tripling the obstacle density. We also increase the maximum velocity of the robots to 30 units/sec from 20 units/sec, making the robots move 1.5 times faster than in the o ine environment. Obstructed perception occurs in both the o ine and online environments.

For the online environment, each robot of the swarm contains a slightly mutated copy of the optimized LJ force law rule set found with o ine learning. There are v e goals to achieve in a long corridor, and between each randomly positioned goal is a di erent obstacle course with 90 randomly positioned obstacles. The LJ force law learned in o ine mode is not su cien t for this more di cult environment; it produces robots that never reach the goal (due to the high percentage of obstacles).

Robots that are left behind (due to obstacle cul-de-sacs) do not proceed to the next goal, but robots that collide with obstacles and make it to the goal are allowed to proceed to the next goal. We assume that damaged robots can be repaired once they reach a goal. Although noise in dynamic environments is not specifically modeled in our simulation, it has been shown with actual robots that the physicomimetics framework is robust with modest amounts of noise (Spears, Gordon-Spears, Hamann, and Heil 2004). In fact, noise can actually improve performance by overcoming local optima in the behavior space (Martinson and Payton 2005; Spears and Gordon 1999).

In Chapters 5 and 6, we have shown that the robots easily learned to avoid colliding with obstacles, so our focus in this chapter is on the survivability of the robots (i.e. the number of robots that reach a goal). When the robots are left behind in cul-de-sacs, the number of robots that survive to reach a goal is lowered, causing the cohesion of the formation to be reduced. We utilized two different methods to improve the survivability:

> if $robot_i$ is not moving (due to an obstacle) and $robot_i$ has no moving neighbors, then $robot_i$ mutates it's own robot-goal interactions. This mimics "panic behavior" seen in animals.
>
> if $robot_i$ is not moving (due to an obstacle in the way) and a neighboring $robot_j$ is moving, then $robot_i$ receives $robot_j$'s robot-robot interactions.

We addressed the first method in Chapter 6 and presented our results. We focus on the second method in this chapter.

7.4.1 Experimental Results

We compared DAEDALUS to three control studies. In the first control study, we train the robots with an offline EA on small obstacles, and then test them again on

small obstacles to verify their performance. In the second control study, we train the robots with an o ine EA on large obstacles and test them on large obstacles. The purpose of this control study is to clarify the di cult y of the task. Finally, in the third control study, we train the robots with an o ine EA on small obstacles and test them on large obstacles. The purpose of this study was to see how well the knowledge learned while avoiding small obstacles transferred to large obstacles. All results are averaged over 100 runs.

Figure 7.2 shows the results. The y-axis gives the number of robots that survived to reach the goal at each stage for the four di erent experiments. The top performance curve is for the rst control study. Note that learning with small obstacles in o ine mode is not di cult, and the robots perform very well in the online environment. This is due to the fact that the small obstacles make the environment less dense providing the robots su cient space to navigate. Out of 60 initial robots released in the online environment, 93.3% survived to reach the last goal. With such small obstacles (which is the maximum density examined in the related literature), obstructed perception is not an important issue.

In the results presented in Chapter 6, robots that learned without obstructed perception on larger obstacles had a reasonably high survival rate (78%). The bottom (dashed) performance curve shows the e ect of obstructed perception (the second control study). Learning with large obstacles in o ine mode with obstructed perception is very di cult, and the test results show that out of 60 robots released initially into the online environment only 35% (21 robots) survived to reach the last goal. This is due to the fact that the environments with larger obstacles create large numbers of cul-de-sacs that obstruct perception.

The third control study (see \NO DAEDALUS(small{large)"), where o ine training occurs with small obstacles and testing occurs with large obstacles, the results are

surprisingly good. Despite an initial drop in performance, performance at the fth goal is quite acceptable (out of the initial 60 robots, 41.6% (25 robots) survived to reach the nal goal). This is a 6.6% improvement over the robots that were trained on larger obstacles. These results run counter to accepted wisdom, which states that it is best to train on the hardest environments that you will encounter. In fact, this example demonstrates that training on simpler problems and applying the knowledge gained to harder problems can potentially provide superior results. Why is this so? As with developmental psychology, one does not train children on hard problems immediately, instead, we train them on easier problems rst, in the hopes that they will learn the \basics" (which are important building blocks for solving other, more di cult, problems) more quickly.

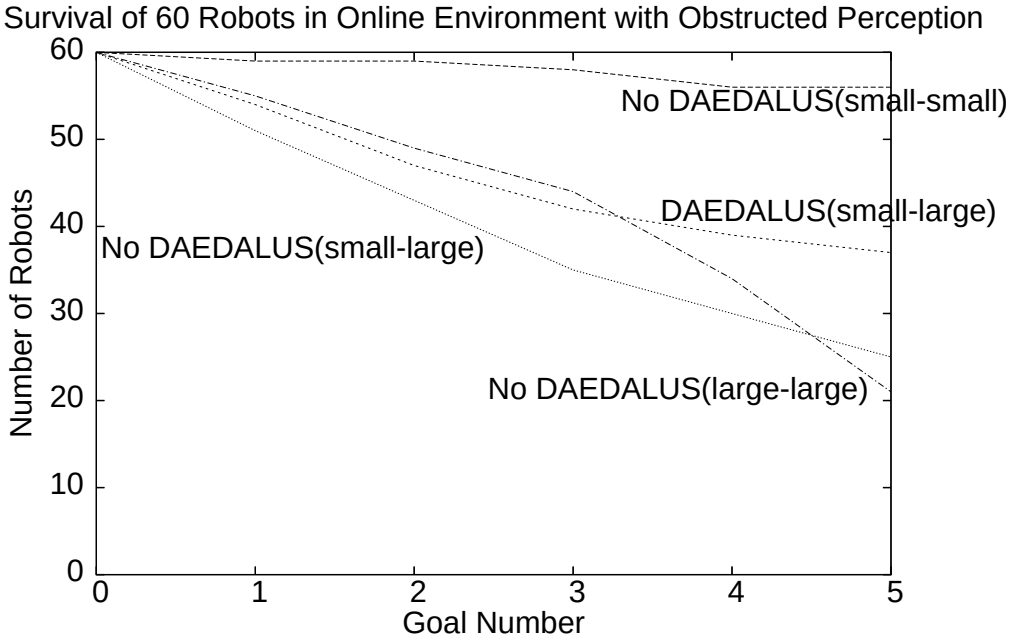

Figure 7.2: Four di erent experimerts of number of robots surviving. All robots are trained with obstructed perception and tested with and without DAEDALUS. The results are averaged over 100 independent runs.

If we extend the developmental psychology analogy further, we note that we encourage children to experiment and modify their behavior, based on changes in the environment. Furthermore, they share the lessons learned. This is precisely what the DAEDALUS system does. The final performance curve in Figure 7.2 shows the results. With an initial 60 robots, 61.6% or 37 robots survived to reach the last goal. This is a 26.6% improvement over the robots that learned in an environment with the larger obstacles, and a 20% improvement over the robots that learned with small obstacles and tested with the larger obstacles without DAEDALUS. These preliminary results are very promising. Although encouraging the robots (or children) to explore and experiment does provide an early drop-off in performance (compared to the "NO DAEDALUS (large-large)" curve), the results after three goals are superior. This is a classic example of "exploration" vs "exploitation". Pure exploitation of learned knowledge is good up to a point, but will eventually fail as the problems become more difficult. Exploration provides the key to adapt to these changing environments. DAEDALUS provides just this form of exploration.

7.5 Homogeneous Swarm Learning - Experimental Results

For the DAEDALUS performance curve given above, all robots had the same mutation rate, which was 5%. Hence, each robot had the same rate of exploration. Although the rules for each robot may differ, their mutations rates are identical, and we refer to this system as "Homogeneous DAEDALUS". However, there are numerous problems with this approach. First, the results may depend quite heavily on choosing the correct mutation rate. How is this mutation rate to be chosen? Second, the best mutation rate may also depend on the environment, and should potentially change as the environment changes. How is this to be accomplished?

Since the mutation rate may have a major effect on performance, we decided to explore this effect by conducting several experimerts with different mutation rates. Figure 7.3 shows five independent experimerts of Homogeneous DAEDALUS. Five different mutation rates were used: 1%, 3%, 5%, 7%, and 9%. The results are quite striking. Of the five different mutation rates, only 5% and 7% did well (with about 35 robots surviving to the last goal). Recall that the DAEDALUS performance curve shown in Figure 7.2 resulted from an arbitrarily chosen mutation rate of 5%. As it turns out, we were extremely fortunate in our design decision. For example, with mutation rates of 1%, 3%, and 9%, at most 23 robots survive to reach the final goal. The performance curve for the 9% mutation rate is especially interesting. Although promising at first, it appears as if the mutation rate is so high that it eventually causes an extremely deleterious mutation to appear. Mutation rates of 1% and 3% are too low to cope with the changed environment.

7.6 Heterogeneous Swarm Learning - Experimental Results

In an attempt to address the problem of choosing the correct mutation rate, we divided the robots into five groups of equal size. Each group of 12 robots was assigned a mutation rate of 1%, 3%, 5%, 7%, and 9%, respectively. This mimics the behavior of children that have different "comfort zones" in their rate of exploration. Since different robots have different mutation rates, we refer to this system as "Heterogeneous DAEDALUS". Figure 7.4 shows the results, in comparison with the three control studies shown in Figure 7.2. The label "Het. DAEDALUS (small-large)" shows the survivability of robots with pre-assigned mutation rates. Out of the initial 60 robots, 29 or 48% robots survived to reach the final goal. Although this is higher than our second and third control studies, it did not produce results as good as the

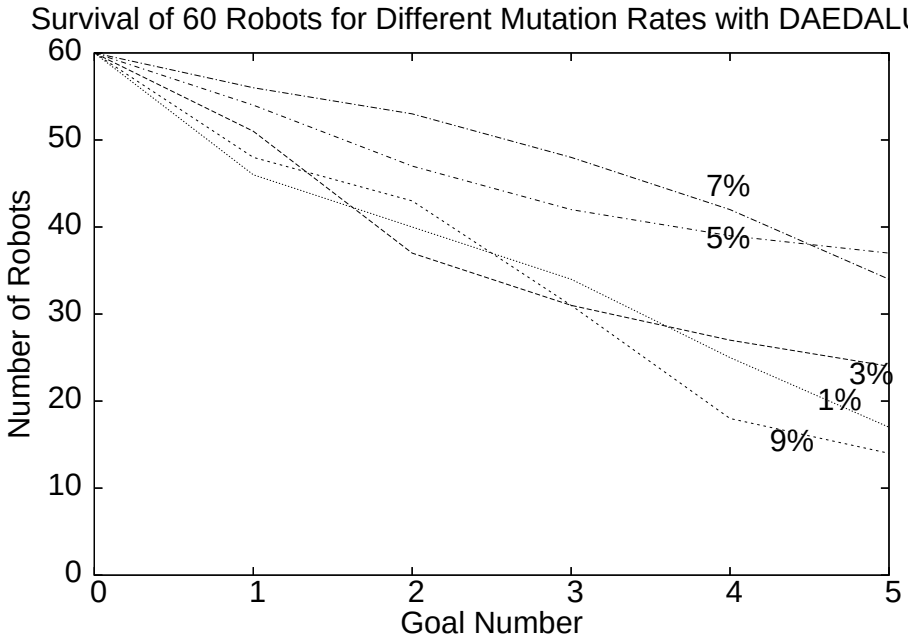

Figure 7.3: Five different mutation experiments of robots surviving. All robots are trained with obstructed perception and tested with DAEDALUS. The results are averaged over 100 independent runs.

results achieved with Homogeneous DAEDALUS using a 5% mutation rate (as shown in Figure 7.3). In fact, the result at the final goal is essentially identical to the average of the five performance curves shown in Figure 7.3.

7.6.1 Extended Heterogeneous DAEDALUS Results

In an attempt to improve performance, we again borrowed from the analogy of a "swarm" of children learning some task. Not only do they share useful information as to the rules they might use, but they also share meta-information as to the level of exploration that is actually safe! Very bold children might encourage their more timid comrades to explore more than they would initially. On the other hand, if a very bold child has an accident, the rest of the children will become more timid. In "Extended Heterogeneous DAEDALUS", five groups of children are again initialized

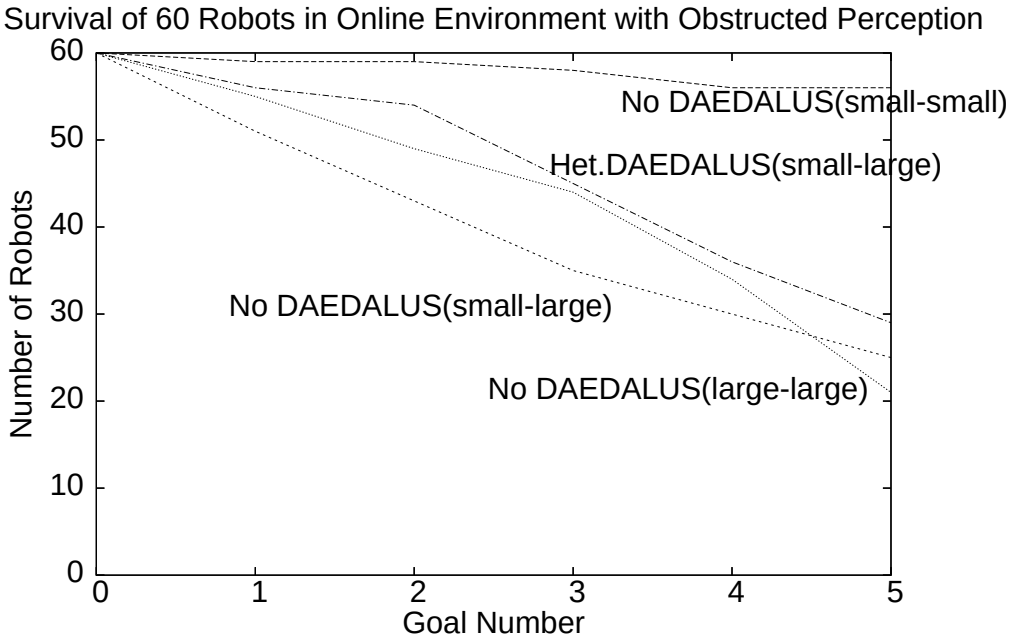

Figure 7.4: Number of robots surviving with predetermined mutation rates. The mutation rates are not exchanged. All robots are trained with obstructed perception and tested with or without DAEDALUS. The results are averaged over 100 independent runs.

with mutation rates of 1%, 3%, 5%, 7%, and 9%. However, in this situation, if a robot receives the rules from a neighbor (which, again, occurs if that robot is in trouble), it also receives the neighbor's mutation rate. In this implementation, children in trouble not only change their rules, but their mutation rate. Figure 7.5 shows the results of this study. The curve labeled with "Ex.Het.DAEDALUS(small-large)" refers to the survivability of robots with pre-assigned mutation rates that also allows the robots to receive a neighbor's mutation rate, if the robot receives the neighbor's rules. The behavior is quite good. On average, 34 robots survive to reach the final goal, which is very close to the optimum value of 37 found by the best Homogeneous DAEDALUS experiment.

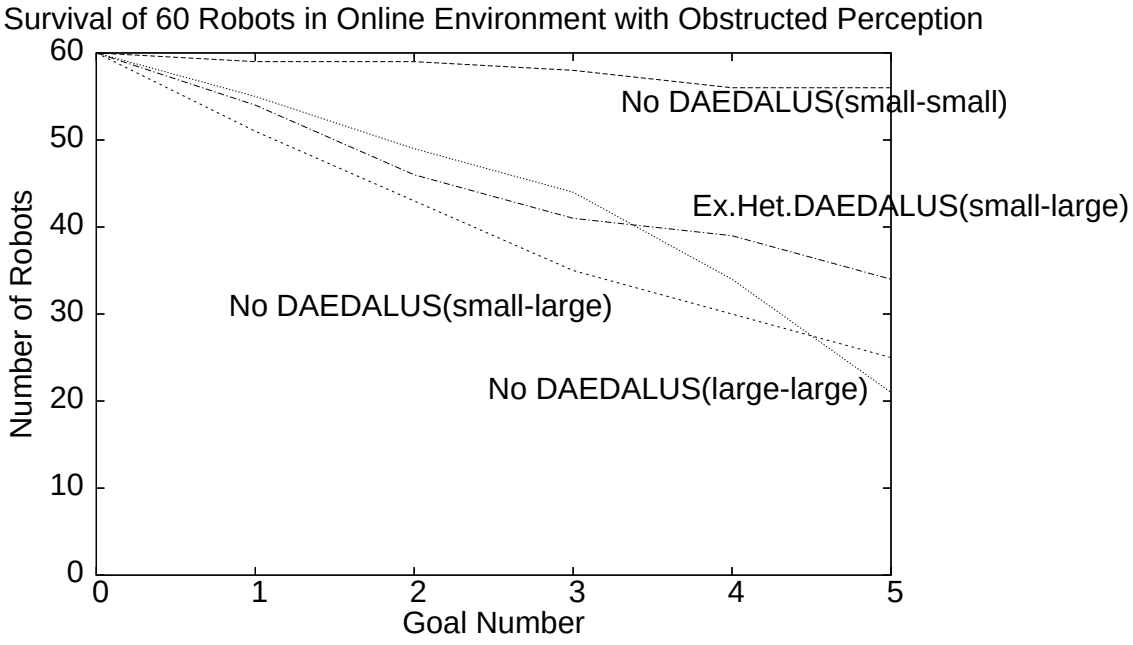

Figure 7.5: Number of robots surviving with predefined mutation rates. The mutation rates are exchanged. All robots are trained with obstructed perception and tested with or without DAEDALUS. The results are averaged over 100 independent runs.

7.7 Effect of Mutation in Swarm Learning

We explored the effect of heterogeneous swarms in an online environment and compared our results with the offline homogeneous swarms. We maintained the diversity in our heterogeneous swarm by allowing robots to exchange their predefined mutation rates. The robots learned to avoid cul-de-sacs in the online environment and maintain the diversity of the population. Table 7.1 shows the mutation rates of robots that survive to reach a goal.

At the beginning, there are five groups of robots. They are initialized with mutation rates of 1%, 3%, 5%, 7%, and 9%. The robots with 1% and 3% mutation rates had a more difficult time surviving compared to the robots with other three mutation rates. Thirty seven robots survived to reach the fifth goal, and clearly the 5% and

	Mutation Rate				
robots survive	1%	3%	5%	7%	9%
60-start	12	12	12	12	12
54-goal1	10	10	11	12	11
47-goal2	9	8	11	10	9
42-goal3	6	7	11	10	8
39-goal4	6	6	10	9	8
37-goal5	5	6	10	9	7

Table 7.1: The number of robots that survive to reach a goal and their mutation rates.

the 7% mutation rates performed better than the other three mutation rates. With 1% mutation, seven robots did not reach the fifth goal, and with 9% mutation, five robots did not reach the fifth goal. Notice that there are still robots with all five mutation rates surviving in the environment. This still maintains the diversity of the swarm.

7.8 Summary

This chapter addressed the important issue of "obstructed perception" in learning behaviors for swarms of robots that must avoid obstacles while reaching a goal. This issue has been largely absent from the literature. Our obstacle density is also three times higher than the norm, making obstacle avoidance a far more difficult task. Since obstructed perception makes the task far more difficult, DAEDALUS had to be extended. Our first extension was to allow different robots to have different rates of exploration, which affects the rate at which they change their behavioral rules. The second extension allowed the robots to also share their rates of mutation, enabling robots to find the right balance between exploration and exploitation. Results of the extended system were almost as good as the best results we were able to achieve when

the exploration rates were controlled by hand. In summary, this chapter introduced a framework that allows swarms of robots to not only learn and share behavioral rules in changing environments (in real time) and but to also learn the proper amount of behavioral exploration that is appropriate.

Chapter 8

Hardware Implementation

8.1 Introduction

In previous chapters, we presented our approach to designing and implementing robot swarms that perform the obstacle avoidance task in simulation. In this chapter, we present our approach to designing and implementing robotic hardware and algorithms for formation control and the obstacle avoidance task.

While most of our hardware modules are designed and tested in-house, we use several off-the-shelf hardware components. Our obstacle avoidance hardware module (OAM) is built in-house at the UW-DRL, and we use off-the-shelf Freescale 68HCS12 microprocessors as the central processing unit. We also design and test a physics-based control algorithm for robot navigation and obstacle avoidance. We call this algorithm AP-lite. We use the UW-DRL Maxelbot robot as our robotic platform[1]. Maxelbots are fitted with hardware and algorithms for robot localization.

Spears et al. presented a trilateration technique for robot localization (Spears, Hamann, Maxim, Kunkel, Heil, Zarzhitsky, Spears, and Karlsson 2006). In 2D trilateration, the locations of three basepoints are known as well as the distances from each of these three basepoints to the object to be localized. Looked at visually, 2D trilateration involves finding the location where three circles intersect. Thus, to locate

[1] See http://www.cs.uwyo.edu/~wspears/maxelbot/

a remote robot using 2D trilateration the sensing robot must know the locations of three points in its own coordinate system and be able to measure the distances from these three points to the remote robot. We make use of this trilateration technology and extend this work to the obstacle avoidance task using our AP-lite control algorithm.

Next, we present our hardware modules and the issues related with the hardware. Then, we present our AP-lite control algorithm and provide an analysis of our formation control and obstacle avoidance results in an outdoor setting. The work presented in this chapter is preliminary and research is ongoing.

8.2 Hardware Considerations

There are numerous issues involved with designing obstacle avoiding robots. One is to decide the type of hardware components to use, and the other is to decide the software needed to run the hardware modules.

Deciding the type of hardware to be used is not an easy task due to the availability of different hardware components. Since the trilateration technology is for localizing other robots and is not capable of localizing obstacles, we designed a new hardware module that takes obstacles into consideration. We designed a PIC micro-controller based hardware module that provides the capability of processing eight sensors. The main reasons to choose the PIC micro-controller includes its low cost, availability of support material, and the capability to program its flash memory.

We decided to use four SHARP GP2D12 IR (Infrared) proximity sensors on the front of a Maxelbot for obstacle avoidance. The decision to choose IR sensors is mainly based on the availability, cost and our prior experience with these sensors. Our previous experience in designing, implementing and testing IR sensors on a prior

generation of Maxelbots aided us tremendously. Integration of the sensor module to function with all the other hardware modules on the Maxelbots is another challenging issue.

One of the most challenging problems of integrating hardware modules is the communication between the modules that control various robotic functions. Paul M. Maxim at the UW-DRL contributed tremendously in designing, implementing and testing algorithms for hardware communication using an I^2C data bus.

8.3 Hardware Configuration

We designed one Maxelbot robot that is capable of avoiding obstacles. This particular Maxelbot consists of two MiniDRAGON boards[2] powered by a Freescale 68HCS12 microprocessor and a PIC16F7x7 processor by Microchip Technology Inc. as our OAM. We use SHARP GP2D12F6X IR sensors as our proximity sensors.

8.3.1 Maxelbot Robots

Our UW-DRL "Maxelbot" (named after the two graduate students who designed and built the robot) is modular. The platform is an MMP5, made by The Machine Lab[3]. There are two MiniDRAGONs on the Maxelbot, one for motor control and another for trilateration. They communicate via an I^2C bus and this allows us to plug in new peripherals as needed. Figure 8.1 shows the hardware modules of the Maxelbot robot. The motor control MiniDRAGON consists of algorithms that drives the motors. It also has the capability of monitoring the proximity sensors. If the sensor suite is connected to the motor control MiniDRAGON, it allows the control algorithm to take the proximity sensor readings into consideration. The trilateration

[2] Produced by Wytec (http://www.evbplus.com/)
[3] See http://www.themachinelab.com/MMP-5.html

module is shown at the right of the diagram. This module controls the RF and acoustic componerts of trilateration. Additional modules have been built for digital compasses, thermometers, and chemical plume tracing (Spears, Hamann, Maxim, Kunkel, Heil, Zarzhitsky, Spears, and Karlsson 2006).

Figure 8.1: Hardware modules of Maxelbot robot.

8.3.2 Sensor Characteristics

We use four SHARP IR proximity sensors mounted on the front of a Maxelbot for the obstacle avoidance task. The SHARP GP2D12 IR sensor is a distance measuring sensor that outputs an analog voltage proportional to the measured distance. Its effective range is normally 10cm to 80cm. The GP2D12 sensor continuously measures the distance to an object based on the voltage reading and reports this distance as an

analog voltage. The maximum output voltage of this sensor is 2.8 volts. This sensor uses triangulation to detect the distance[4]. Due to this triangulation method, the sensor output is non-linear with respect to the distance being measured. Figure 8.2 shows the typical output of these sensors.

Figure 8.2: GP2D12 sensor output voltage with distance to objects.

[4] See http://www.acroname.com/robotics/info/articles/sharp/sharp.html

```
float ConvertSensorReadingToInches(rawSensorReading)
    float temp;
    temp = (6787.0/(rawSensorReading - 3.0)) - 4.0;
    return (temp/2.54);
```

Figure 8.3: Function to convert raw sensor readings to inches.

The GP2D12 IR sensors are provided with 5 volts of input voltage and the analog to digital (AtoD) converter converts the output voltage to a digital signal based on the raw sensor reading. These raw sensor readings must be converted to distances in inches so that these distances can be used in our AP-lite control algorithm. We make use of a conversion function[5] to linearize the digital sensor reading as shown in Figure 8.3.

8.3.3 Obstacle Avoidance Module

The OAM consists of four GP2D12 IR sensors and the AtoD converter. The analog output of the sensor is converted to a digital signal, and sent via the I^2C data bus to the motor control MiniDRAGON. We provided sensors with 5 volts of input voltage and the output is an analog signal. The AtoD module converts the output voltage to a digital signal that consists of a raw sensor reading between a value of 0 and approximately 550 (the input voltage can be inconsistent), with 0 being the nonexistence of an object in front of the sensor and 550 being an object closest to the sensor. Once all sensor readings are available, the OAM stores the sensor reading in an array so that it can be used by the motor control MiniDRAGON.

The OAM communicates the sensor readings to the motor control module upon request. From the I^2C data bus point of view, this configuration has a master-slave

[5] See http://www.acroname.com/robotics/info/articles/irlinear/irlinear.html

relationship. The OAM is considered to be the slave and the motor control module that requests the information from the OAM is considered to be the master. This method of communication is more effective because all the hardware modules require I^2C data bus usage. If this master-slave communication method does not exist, there is a possibility that one module could hold on to the I^2C data bus without letting other modules use it, causing a total communication failure among modules.

A top view of the Maxelbot robot with IR sensors and the OAM is shown in Figure 8.4.

Figure 8.4: Top view of a Maxelbot robot with the OAM and MiniDRAGONs for trilateration and motor control.

8.4 Experimental Results

Numerous task-driven formations have been successfully performed with the Maxelbots indoors using trilateration. For details, see (Spears, Hamann, Maxim, Kunkel,

Heil, Zarzhitsky, Spears, and Karlsson 2006). One of the drawbacks of trilateration is that it does not allow obstacle avoidance. The OAM with trilateration allows us to maintain robot formations and to avoid obstacles.

This section presents three experiments that aid in identifying the quality of the AP-lite algorithm. Two of our experiments focus on formation control and the other experiment focuses on AP-lite obstacle avoidance with the OAM. All three experiments are conducted in an outdoor setting.

8.4.1 Control Algorithm: AP-lite

Our control algorithm "AP-lite" follows the physicomimetics approach. The physicomimetics framework allows for self-organizing swarms, while AP-lite is only capable of maintaining a formation. AP-lite is specifically designed as a leader-follower algorithm; only the leader broadcasts a signal, rather than all robots as in the physicomimetics approach. The advantage of AP-lite over physicomimetics is that AP-lite provides us with faster robot speed, and further, we can use theory to set the parameter settings. Since we are at the initial stages of our research we decided to conduct all of the experiments with one leader and two follower Maxelbots. AP-lite computes the amount of power required for the left and right motors based on the forces acting upon the robot. Here, we present our AP-lite algorithm for obstacle avoidance with both repulsive and attractive forces acting on the robot. We use Hooke's law as our force law. Hooke's law allows us to model robot interactions similar to the force laws we presented earlier.

The attractive goal force is a global component that is always active; this drives the robot forward with an equal amount of power to both motors. When the robot reaches an obstacle, AP-lite computes the repulsive forces acting on the robot, and change the power supply to the motors. If the robot senses an obstacle from the right

```
void ap_lite()
    float v, vx, vy, r, F, kmid, kside, Fx, Fy, delta_vx, delta_vy
    float theta, theta_new, delta, w, FR, STEP, mass, alpha
    int power_left, power_right
    int RANGE= 30 // reactive distance to an obstacle
    int MAXSPEED= 70 // maximum power supply to motors
    STEP= mass = 1.0 // step size and the mass of the robot
    kmid = 5.0 // Hooke's law constant for two middle sensors
    kside = 4.0 // Hooke's law constant two out side sensors
    FR = 0.5 // friction
    alpha = 1.0 // decides the amount to turn
    Fx = 100.0 // attractive goal force in x direction
    Fy = 0.0 // robot do not move side-way
    v = current_velocity // initial value is at 70%
    vx = FR * v // velocity drops with friction
    vy = 0 // No side-way velocity
    for (all four sensors)
        if (sensor_reading > 30) // filter out low readings
            r = RANGE- ConvertSensorReadingToInches(sensors_reading)
        if (r < 0) r = 0 // no obstacles seen
        theta = virtual_sensor_angle_in_radians
        if ( sensor_1 or sensor_2)
            F = kmid * r // compute force from the two middle sensors
        else
            F = kside * r // compute force from the two out side sensors
        Fx = Fx - (F * cos(theta)) // Repulsive x component
        Fy = Fy - (F * sin(theta)) // repulsive y component
        endif
    endfor
```

(i.e. right-most sensor, S0, (see Figure 8.7) reads a high value), AP-lite reacts to this repulsion by supplying less power to the left motor. If the robot senses an obstacle from the left (i.e. right-most sensor, S3, (see Figure 8.7) reads a high value), AP-lite reacts to this repulsion by supplying less power to the right motor. If both sensors in the middle, S1 and S2, are reading high values, the repulsive forces from both sensors counteract the goal force, causing the robot to come to a halt. Our robots are nonholonomic and they always move in the forward direction.

The Maxelbot's turn is decided by the turn_function (see Figure 8.6). Since our

```
delta_vx = STEP* Fx / mass // change in velocity in x direction
delta_vy = STEP* Fy / mass // change in velocity in y direction
vx = vx + delta_vx // velocity in x direction
vy = vy + delta_vy // velocity in y direction
v = sqrt(vx * vx + vy * vy) // current velocity
delta = atan2(delta_vy, delta_vx) // direction of change in velocity
current_velocity = v // reset current velocity
w = turn_function(delta) // angle of the robot turn
theta_new = atan2(vy, vx) // robot moves in first or second quadrant
if ((- /2.0 <= theta_new) and (theta_new <= /2.0))
    power_right = (int)(v + v * alpha * w) // power to right motor
    power_left = (int)(v - v * alpha * w) // power to left motor
    // proportionally cap the motor power
    if (power_right > MAXSPEED or dcl > MAXSPEED)
        if (power_right >= power_left)
            power_left = MAXSPEED* power_left / power_right
            power_right = MAXSPEED
        else
            power_right = MAXSPEED* power_right / power_left
            power_left = MAXSPEED
        endif
    endif
endif
if (dcr >= 0)
move forward right motor with power_right
endif
if (dcl >= 0)
move forward left motor with power_left
endif
end ap-lite
```

Figure 8.5: Pseudocode of the AP-lite.

```
float turn_function (float angle)
    float angle_radians = angle
    // robot moving to the second quadrant
    if (( /2.0 < angle) and (angle <= ))
        angle_radians = - angle
    endif // robot moving to the third quadrant
    if ((- < angle) and (angle < - /2.0))
        angle_radians = - - angle // only used in backward move
    endif
    return angle_radians
end turn_function
```

Figure 8.6: Pseudocode of the turn function.

robots currently do not move backward, a robot moving into the second or third quadrant is not relevant here. The turn_function returns the turning angle in radians.

The virtual angles of the positioning of the four IR sensors are shown in Figure 8.7. These sensor angles allow us to effectively model the robot's stopping behavior when all sensors detect an obstacle in front of the robot.

Figure 8.7: Positioning of sensors on the front of a Maxelbot.

8.4.2 Methodology

For all the experiments, the Maxelbots are run outside in a region in the center of the University of Wyoming campus called "Prexy's Pasture" (see Figure 1.3) at[6]. Prexy's consists mostly of grass, of average height 8 cm (3^{0}), interspersed with concrete sidewalks, trees, rocks, leaves, and other debris. The grass hits the bottom of the Maxelbot. Although generally at, the ground slope can change rapidly (within 61 cm or 2^{0}), by up to 20 , at boundaries. All the results of the formation tests are averaged over ten independent runs. All the experiments are conducted with a leader and two follower Maxelbots. The first two experiments are,

triangular formation: followers with AP-lite and leader remotely controlled (RC).

linear formation: followers with AP-lite and leader remotely controlled (RC).

The last experiment tests the reliability of the OAM working with the AP-lite.

the OAM effect on AP-lite when the Maxelbot is navigating in an outdoor terrain with obstacles.

Detailed discussions of the experiments are given in the next two sections.

8.4.3 Formation Control

To test the quality of our AP-lite algorithm, we conducted two experiments. The focus of the first two experiments is to test AP-lite accuracy with two different robot formations: triangular and linear. In both of these experiments, the leader is remotely controlled while the followers control their navigation using the AP-lite algorithm.

[6] http://www.laramie.willshireltd.com/PrexysPasture.html

The third experiment measures the performance of the OAM module working with the AP-lite. In this experiment, the leader moves in the forward direction and avoid obstacles; all three Maxelbots use the AP-lite algorithm to navigate. The leader reacts to both attractive and repulsive forces and the followers only react to the attractive force of the leader. In all experiments, the followers self-calibrate their initial x and y position relative to the leader at the beginning of every run. The leader has a maximum power (speed) of 70% and the followers move with greater than 70% of power.

In the first experiment, the followers were positioned in a triangular formation at certain positions as shown in Figure 8.8. The leader is controlled with the RC and the followers are controlled with the AP-lite but without the OAM. Dark arrows represent the initial robot headings. The initial positions shown are self-calibrated by the followers.

Figure 8.8: Three Maxelbots in triangular formation; distances shown are initial x and y positioning.

Figure 8.9 shows the results of two Maxelbots following the leader in triangular formation. It is clear that the followers maintain their position consistently using AP-lite. This also illustrates the accuracy of the trilateration technique. The robot on the left of the formation oscillates slightly more on its Y-axis more than the robot

on the right. We believe this is a hardware issuerather than AP-lite.

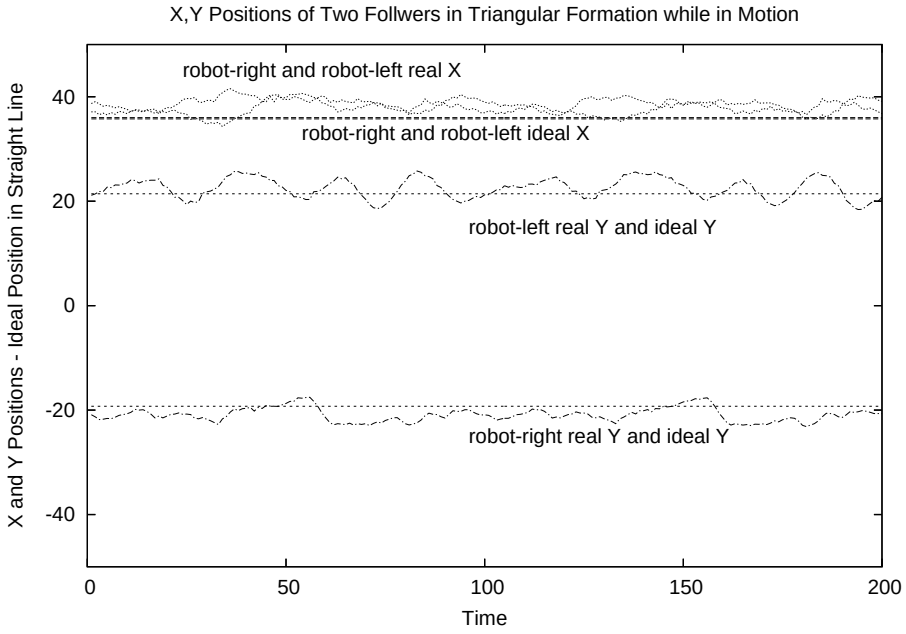

Figure 8.9: Changein position of two follower Maxelbots in triangular formation.

In the secondexperimert, the followers were positioned in a linear formation at certain positions as shown in Figure 8.10. Once again, the leader is controlled with the RC and the followers are controlled with AP-lite.

Figure 8.11 shows the results of two Maxelbots following the leader in linear formation. The mean error of follower two's real X position is larger than the mean error of follower one's real X position. Our investigation showed that follower two is the same robot that we used as the left robot in the triangular formation. This indicates that follower two's hardware problem is potentially causingthis change in behavior. Also, another possible scenariois the lack of trilateration signal strength from the leader due to distance.

The bottom two curves of the graph represent the followers real and ideal Y positions, and these two curves are overlapped in our graph. Both follower maintain

Figure 8.10: Three Maxelbots in linear formation; distances shown are initial x and y positioning.

their Y position extremely well.

8.4.4 Obstacle Avoidance

The third and last experiment tests the quality of our AP-lite algorithm working together with OAM module. Currently only the leader Maxelbot carries the the OAM. The resulting figures from Figure 8.12 through 8.14 show the corresponding relationship between the raw sensor readings and the power to the left and right motors. For all the graphs, the X-axis represents the time (the number of data points collected over a period of 15 minutes in an outdoor setting), and the Y-axis shows the raw sensor reading and the power to the motors. In all three graphs, a constant value of 200 is added to each of the raw sensor readings. Thus, a raw sensor value of 0 corresponds to a value of 200. This is done to enhance the visual effect of the graphs.

Figure 8.12 shows the correlation between the right-most sensor, S0, and the power to the left motor. The graph shows clear correlation between the repulsion that occurs from the right side of the robot and the robot turning left. Clearly,

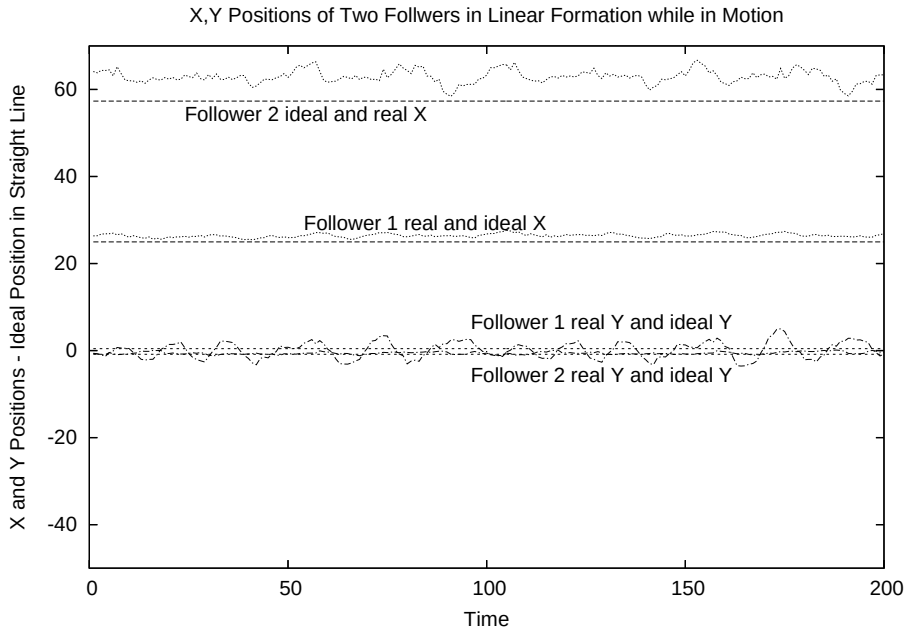

Figure 8.11: Change in position of two follower Maxelbots in linear formation.

visible at segments of the curve at value 70 represent the robot running forward at 70% of power with no repulsion acting on the robot from any of the sensors. The at segments of the bottom curve, closest to value 0, represent the left motor power at 0% causing the wheels on the left side of the robot to be stopped.

Figure 8.13 shows the correlation between the left-most sensor, S3, and the power to the right motor. Again, the graph shows a clear correlation between the repulsion that occurs from the left side of the robot and the robot turning right. Again, the clearly visible at segments of the curve, at value 70, represent the robot running forward at 70% of power with no repulsion detected from any of the sensors. The at segments of the bottom curve, closest to value 0, represent the right motor power at 0% causing the wheels on the right side of the robot to be stopped.

Figure 8.14 shows the correlation between the two middle sensors S1, S2, and the power to right and left motors. Again, as a visual enhancement, we show the average

Figure 8.12: Correlation between the right-most sensor, S0, and the power to the left motor.

Figure 8.13: Correlation between the left-most sensor, S3, and the power to the right motor.

of the two middle sensor reading and the average of the power to the two motors.

When both middle sensors detect obstacles, the repulsive forces acting on the robot are greater than the attractive force pulling it forward. This causes the robot's net force to be negative. Since our robot does not move sideways and we deliberately avoided the backward movement of the robot's motors, the robot comes to a stop when there is a negative net force acting on it. Once again, it is clear that AP-lite reacts well to both attractive and repulsive forces acting on the robot.

Figure 8.14: Correlation between the two middle sensors, S1 and S2, and the power to the left and right motors.

8.4.5 Further Analysis of Data

Figures 8.15 through 8.17 show the corresponding relationship between the distance to obstacles and the power to robot motors using scatter plots. We use scatter plots because they provide a broader picture of the data. For all the graphs, the X-axis represents the distance to obstacles in inches and the Y-axis represents the power

to the motor(s). The distance to the obstaclesis measuredusing the function in Figure 8.3. Again, all the experimerts are conducted in an outdoor setting. If the robot is lessthan 30 inchesaway from an obstacle,the robot feelsthe repulsive forces from that obstacle. Beyond 30 inches, obstacleshave no e ect on the robot. The closerthe robot getsto an obstacle,the higher the repulsive force it feels. When the robot's repulsive forcesovercomethe attractiv e goal force, the robot comesto a stop.

Figure 8.15 shows the distance to an obstacle on the right of the robot and the power to the left motor. There are v e di erent scenariosvisible in this scatter plot. First, the robot's left motor moves at a maximum constant speed of 70 when there are no obstaclesrepulsing the robot from the right. Second,the robot's left motor power decreaseswhen the robot detects an obstaclefrom its right at a distance less then 30 inches. Third, the robot is even closerto the obstaclecausingthe middle-left sensorto detect the obstacleand this further decreasesthe power to the left motor. Fourth, the robot's left motor movesat the maximum speedeven though the robot is detecting an obstacleon its right. This is becausethe robot reactsto a closerobstacle on the left beforeit reactsto the obstacleon the right. Fifth, the robot's left motor is stopped when there are no obstaclesdetected on the right. This is becausethe robot's two middle sensorsdetect an obstaclein front of the robot.

Figure 8.16 shows the distance to an obstacle on the left of the robot and the power to the right motor. The scenariothe robot facesis symmetric to the scenario seenin the previous (see Figure 8.15) graph. Interestingly, the robot detects less obstacleson its left comparedto its right. This is causedby the di erence in obstacle density along the robot's path.

Figure 8.17 shows the average(the distancesmeasuredby the two middle sensors are averaged)distanceto an obstaclein front of the robot and the averagepower to the left and right motors. The scatter plot clearly shows a linear correlation of the

Figure 8.15: Correlation between the right-most sensor, S0, and the power to the left motor.

Figure 8.16: Correlation between the left-most sensor, S3, and the power to the right motor.

motor power to the left and right motors with the distance to an obstacle in front of the robot. The points where the distance is greater than 30 while the power is reduced correspond to situations where there are obstacles to the left or right.

Figure 8.17: Correlation between the two middle sensors, $S1$ and $S2$, and the power to the left and right motors.

8.5 Summary

In this chapter, we presented our work of designing and implementing hardware and software for obstacle avoidance. We introduced our robot platforms, the OAM and a novel control algorithm, AP-lite, for robot control. AP-lite is reactive to an attractive force from a virtual goal and repulsive to forces from obstacles. We implemented our OAM using SHARP IR sensors and a Maxelbot robot. We presented three different leader-follower experiments. The first two experiments tested the quality of AP-lite using two followers in two different formations: triangular and linear. The third ex-

periment tested the quality of AP-lite using the leader with the OAM. In addition, we have presented an analysis of our results using scatter plots. Our results demonstrate the accuracy and quality of both our hardware and software.

Chapter 9

Conclusion

9.1 Accomplishments

Traditionally, accomplishing complex robotics tasks involves an expensive and possibly remotely controlled robot. This traditional approach overwhelms our robotic resources with an ever increasing complexity of task requirements and recent world events. The traditional approaches do not support complete autonomy nor the distributed computing capability of the robots. The robots depend on the human operations. The performance feedback becomes vital, and delay or perturbation of the feedback loop due to environmental constraints may jeopardize the task. The lack of a global observer could be fatal to the mission's success. Due to these disadvantages in traditional approaches, we focus on designing rapidly deployable, scalable, adaptive, cost-effective, and robust swarms. Our objective is to develop autonomous distributed mobile sensing robot swarms. Our objective is to provide a scientific, yet practical, approach to the design and analysis of swarm robotic systems.

Our specific objective within the context of this thesis is to address the related issues and concerns of a swarm of robots that reaches a goal while avoiding obstacles and maintaining a cohesive formation, even when the environment changes. We provide an empirical analysis of obstacle avoiding robot swarms that extensively contributes to our understanding of the general swarm robotic issues.

We have successfully designed and implemented an obstacle avoiding robot swarm in simulation as well as using physical robots. All our algorithms are rapidly deployable, scalable, adaptive, cost-effective, and robust. Our robots have limited sensor input but the aggregate behavior of the collective emerges through the interactions among swarm members. Our algorithms are distributed and provide computational efficiency and fault-tolerance.

9.2 Contributions

The work in this thesis makes many contributions to several areas of swarm robotics research.

- Improved performance in obstacle avoidance:
  - applied a new force law for robot control to improve performance. The thesis presents a novel extension to the physicomimetics framework, with the use of a generalized Lennard-Jones force law.
  - provided novel objective performance metrics for obstacle avoiding swarms. The metric includes the measurement of the number of robots that collide with obstacles, their connectivity, the number of robots that reach the goal, and the time taken by at least 80% of robots to reach the goal.
  - improved scalability of the swarm in obstacle avoidance. We provide methodical empirical analysis showing the scalability ranging from 20 to 100 robots, and from 20 to 100 obstacles.
  - improved performance of obstacle avoidance with obstructed perception. We address the important issue of "obstructed perception" in learning

behaviors for swarms of robots that must avoid obstacles while reaching a goal.

- Invented a novel real-time learning algorithm (DAEDALUS):
  - demonstrated that a swarm can improve performance by mutating and exchanging force laws. We invented a novel real-time learning algorithm. We show how concepts from population genetics can be used with swarms of agents to provide fast online adaptive learning in changing environments.
  - demonstrated the feasibility of DAEDALUS with obstacle avoidance, in environments three times denser than the norm. The robot's online environment is far more difficult than the offline environment due to the high density of obstacles. We provide an empirical analysis showing the feasibility of DAEDALUS in this difficult environment.
  - explored the trade-offs of mutation on homogeneous and heterogeneous swarm learning. We extend our real-time learning algorithm to allow robots to share their rates of mutation, allowing the robots to find the right balance between exploration and exploitation.
- Hardware implementation:
  - presented a novel robot control algorithm that merges physicomimetics with obstacle avoidance. We introduce an OAM to the Maxelbot robots and a novel control algorithm (AP-lite) that reacts to an attractive force from a virtual goal and to repulsive forces from obstacles.

9.3 Future Directions

Short Term Future Directions The short term future directions of our work focus on improving and extending our contributions as well as applying them to provide practical solutions to complex problems. A significant portion of this thesis is dedicated to exploring the issue related to partial observation. Still, there are significant issues that arise with respect to"wall following methods" and "local minimum trap" problems. These issues are not adequately addressed in this thesis. We have observed "local minimum trap" problem in our work, but we did not make attempts to address this issue in detail. We intend to introduce a hybrid liquid and gas model combined with our DAEDALUS approach as a solution. The robots will switch to a gas model as presented in (Kerr, Spears, Spears, and Thayer 2004) to avoid the "local minimum trap". Once the robots have escaped they can continue using the previous force law. The performance metrics we define provide future researchers with meaningful benchmarks of swarm behavior. A possible extension to these metrics could provide the distribution of sub-swarms and the number of robots in each sub-swarm.

The results of our heterogeneous swarms are promising, but we believe that robot behavior can be further improved by different mutation techniques. We intend to explore other approaches to develop more robust adaptive algorithms for online learning. We believe that we can accelerate the learning of the mutation rates. For example, currently, when a robot is in trouble, it receives the rules and mutation rate of a neighbor that is not in trouble. But this same neighbor could also query the robot in trouble to find out its mutation rate. Then the neighbor could spread this information further, to inform other robots that this particular mutation rate might be problematic. Also, another possible avenue for improving the performance of our DAEDALUS approach lies within reward sharing (i.e. credit assignment) techniques. Current work

in classifier systems uses mechanisms such as "bucket-brigade" or "profit sharing" to allocate rewards to individual "agents" appropriately (Grefenstette 1988). However, these techniques rely on global blackboards and assume that all agents can potentially act with all others, through a bidding process. We intend to modify these approaches so that they are fully distributed, and appropriate for online learning of heterogeneous swarms. Our experimental setup requires further expansion to study the feasibility of DAEDALUS in structured environments (i.e. connecting rooms separated with walls).

The results presented using the Maxelbot robots are preliminary. There are potentially numerous application areas that our algorithms and techniques can be applied. Currently, we are investigating methods to further improve our algorithms, so that all the Maxelbots in a formation carry an OAM. This requires improving the AP-lite algorithm to sense and react to all the neighboring robots in the swarm. This requires improved communication between robots. To implement DAEDALUS with physical robots, robots require efficient hardware for communication and data exchange. The current trilateration hardware and ad-hoc communication network used by trilateration are not sufficient for the greater data exchange requirements among robots.

Long Term Future Directions Recent world events have placed increasing demands on the detection and identification of threats that overwhelm current security resources. Employing robot teams to investigate for threats and foreign objects in a predefined area will greatly increase resource effectiveness. Robot teams are expected to play significant roles in future defense, law enforcement, search and rescue, disaster management, and homeland security tactical maneuvers, by providing the ability to acquire and process large amounts of detailed information over large remote areas,

enter areas unsafe for humans, and stay on-task where humans may suffer fatigue or distraction due to peripheral situational factors.

There is a general need to develop the ability of robots to interact with each other, as well as with humans. This ability is required for commercial applications such as map generation, obstacle avoidance, surveillance, chemical/biological plume tracing, chemical/biological source identification, distributed sensing grids, and mine clearing. UAVs and USVs (Unmanned Surface Vehicles) are expected to play significant roles in these applications.

Advantages of UAV and USV teams include the absence of imposed centralized control, autonomous nature of the team members, and the possible ability of achieving high connectivity between the team members. Adaptability, or easy adjustment to changing environment stimuli, resilience to failures due to strength in numbers, and effectiveness in performing multiple tasks further enhances the mission capability and reduces the human operator intervention. UAV's and USVs have been investigated for their potential to support multiple missions, ability for rapid reconfiguration, deployment endurance, and to serve as an unmanned platform to provide the human groups an off-board sensor capability that complements existing systems.

We are able to provide behavioral assurance to UAV and USV teams by

- adapting the physicomimetics framework to incorporate performance feedback for specific tasks and situational awareness.

  Based on the physicomimetics framework, autonomous cooperation will be adapted to UAV's and USVs designed to simulate entry into an area with a predefined boundary that is suspected to be penetrated by foreign objects and/or contaminated by biological or chemical hazards. The multi-USV team will apply our novel robot localization technology to unify control and positioning

and data exchange without the need for continuous global knowledge and will provide necessary performance feedback in conjunction to the environmental movement.

- extending the physicomimetics framework for sensing and performing tasks in a marine environment.

  The objectives of this research include extending the physicomimetics for a multi-robot team in a fluidic environment that simulates ocean dynamics. We intend to modify the physicomimetics framework to include a local data exchange to monitor relative positioning with respect to whether a USV is deliberately moving or idle and combine this data with the global coordinate system to map the environmental forces, i.e., fluidic motion of the system, that will subsequently be used by the physicomimetics algorithms to maintain mission success with minimum energy. The physicomimetics framework will thus incorporate a sequence of active and passive relative position monitoring in conjunction with the absolute coordinate system to provide environmental force data to the physicomimetics framework.

- introducing robot/human roles and interactions to the distributed evolution architecture.

  The objective to develop a software architecture using three critical modules: observer, command and agent. The observer module will dictate the interactions of the robot team reporting to the human team. The command module will dictate the interactions of the human team reporting to the robot team including overriding safety commands that function to terminate a maneuver in varying degrees. The agent module will dictate the interactions within the multi-robot team. The DAEDALUS framework will be introduced to these modules to

improve the situational awareness of the multi-robot team.

APPENDIX I

Complete Results of Reachability for Offline Learning

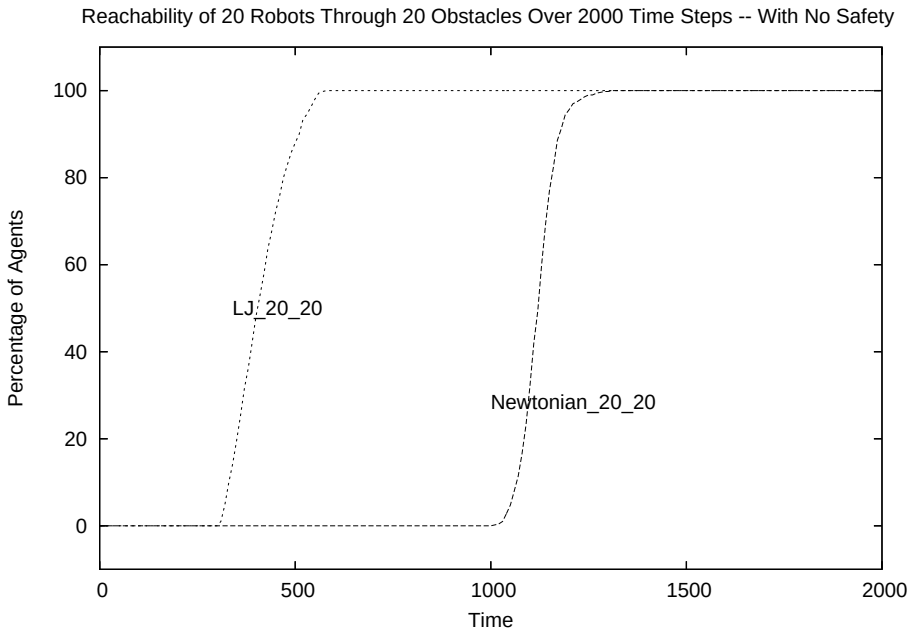

Figure 9.1: Change in reachability over 2000 time steps for 20 robots through 20 obstacles using Newtonian and LJ force laws

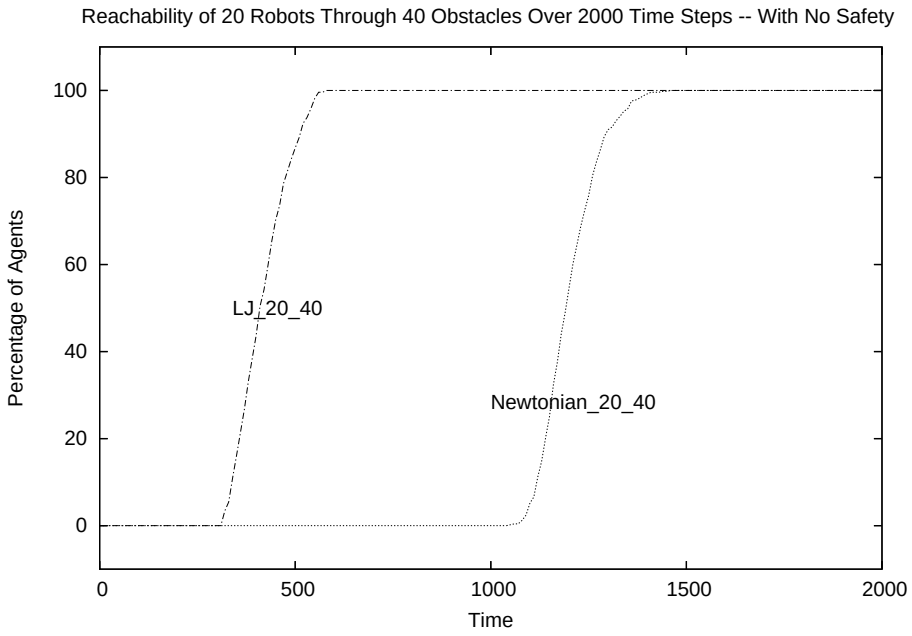

Figure 9.2: Change in reachability over 2000 time steps for 20 robots through 40 obstacles using Newtonian and LJ force laws

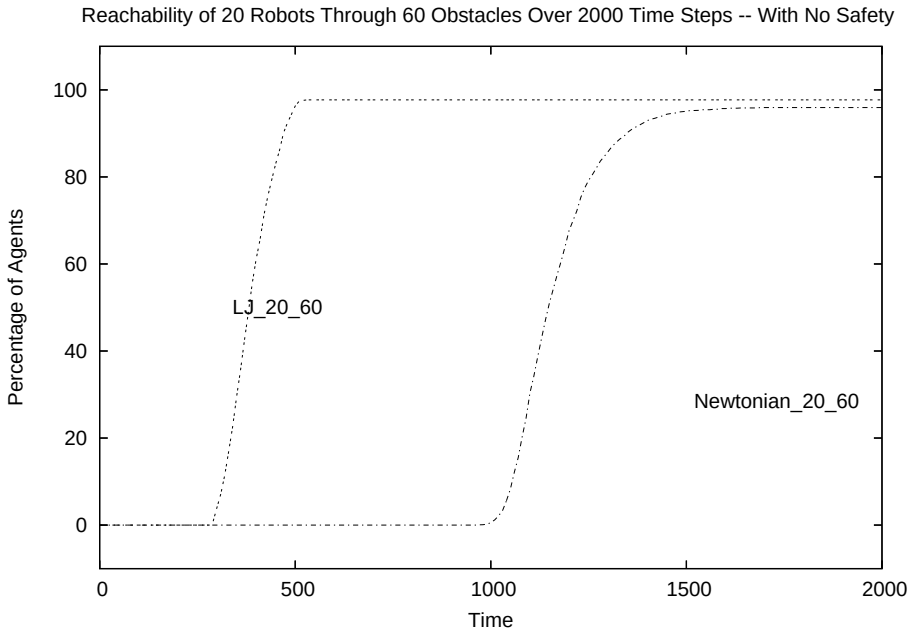

Figure 9.3: Change in reachability over 2000 time steps for 20 robots through 60 obstacles using Newtonian and LJ force laws

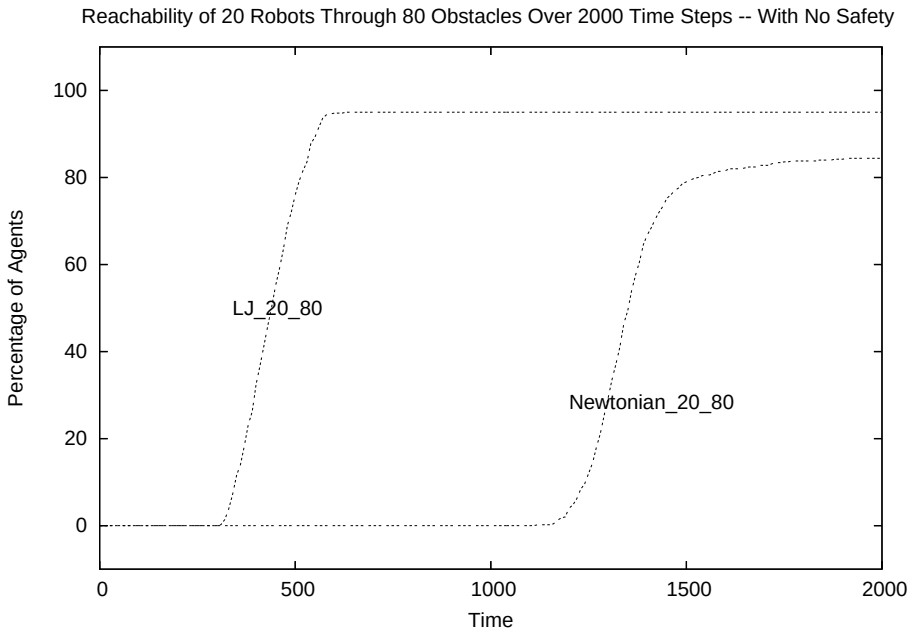

Figure 9.4: Change in reachability over 2000 time steps for 20 robots through 80 obstacles using Newtonian and LJ force laws

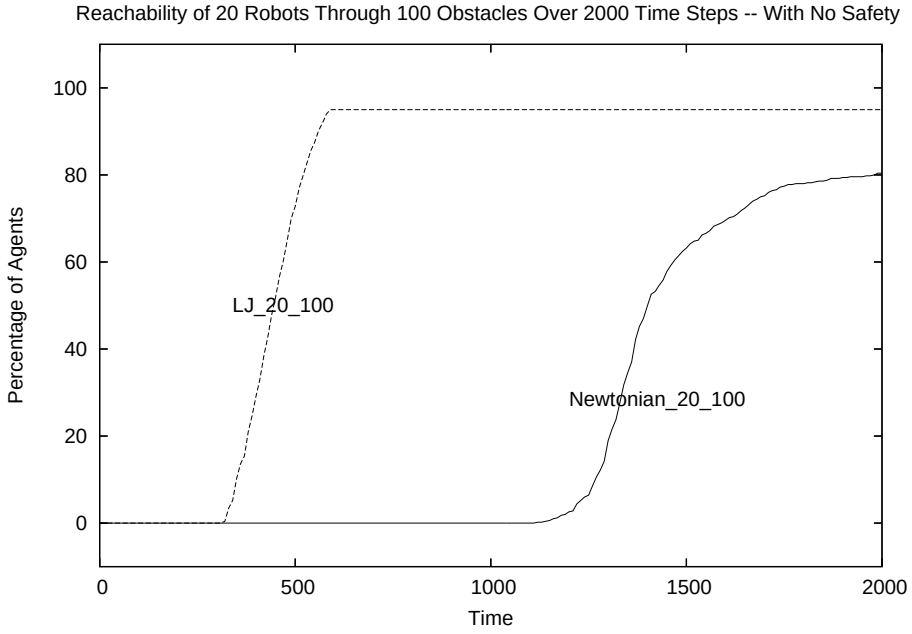

Figure 9.5: Change in reachability over 2000 time steps for 20 robots through 100 obstacles using Newtonian and LJ force laws

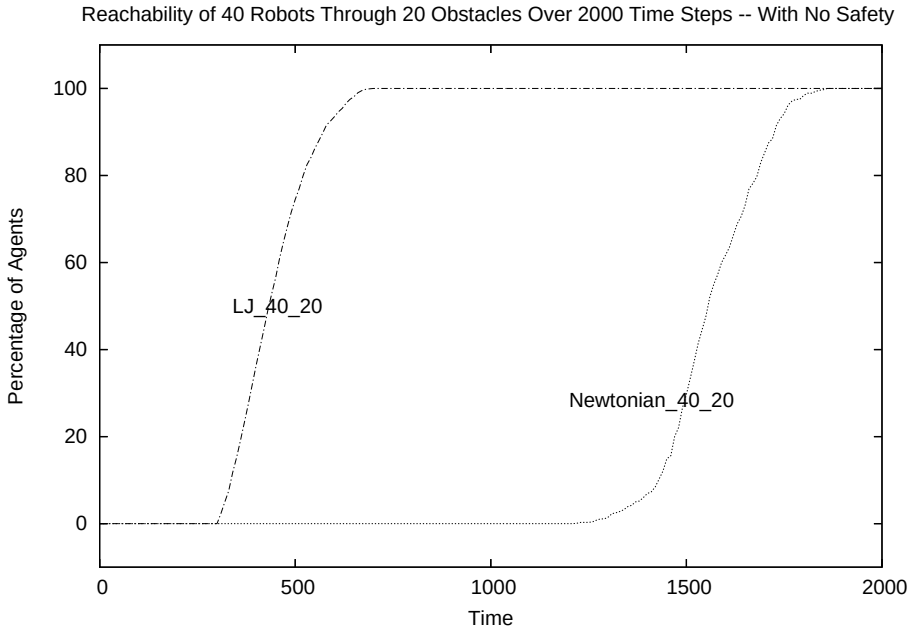

Figure 9.6: Change in reachability over 2000 time steps for 40 robots through 20 obstacles using Newtonian and LJ force laws

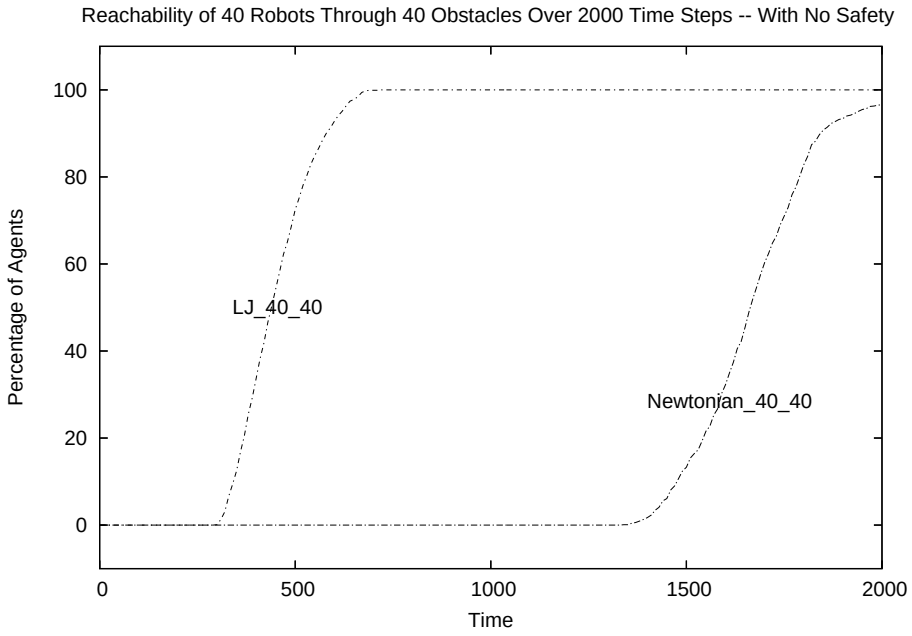

Figure 9.7: Change in reachability over 2000 time steps for 40 robots through 40 obstacles using Newtonian and LJ force laws

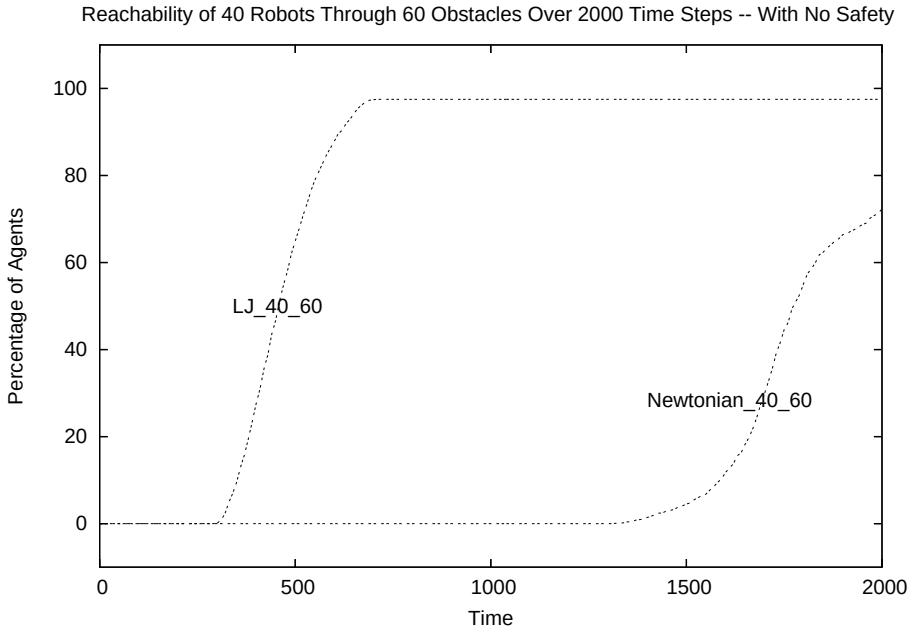

Figure 9.8: Change in reachability over 2000 time steps for 40 robots through 60 obstacles using Newtonian and LJ force laws

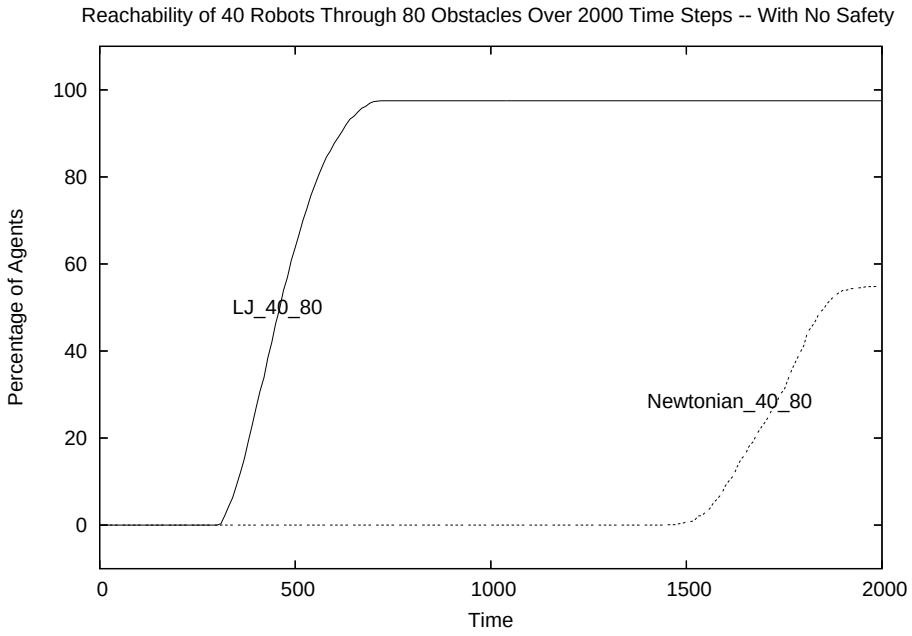

Figure 9.9: Change in reachability over 2000 time steps for 40 robots through 80 obstacles using Newtonian and LJ force laws

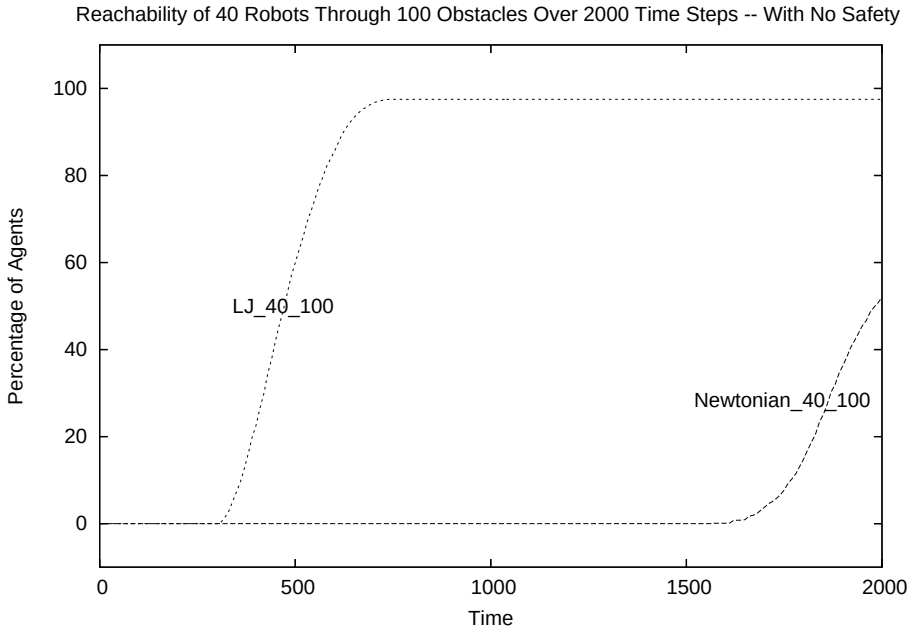

Figure 9.10: Change in reachability over 2000 time steps for 40 robots through 100 obstacles using Newtonian and LJ force laws

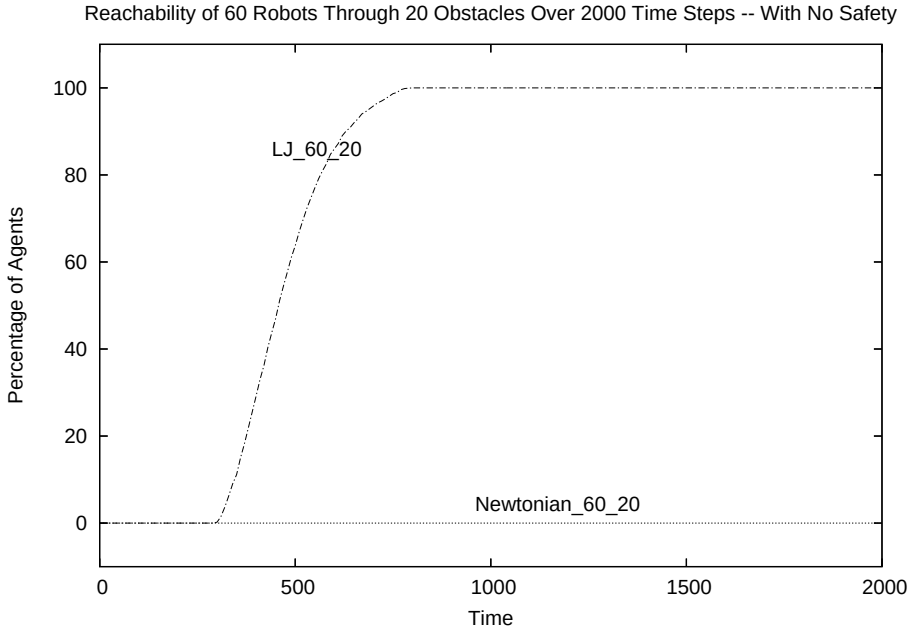

Figure 9.11: Change in reachability over 2000 time steps for 60 robots through 20 obstacles using Newtonian and LJ force laws

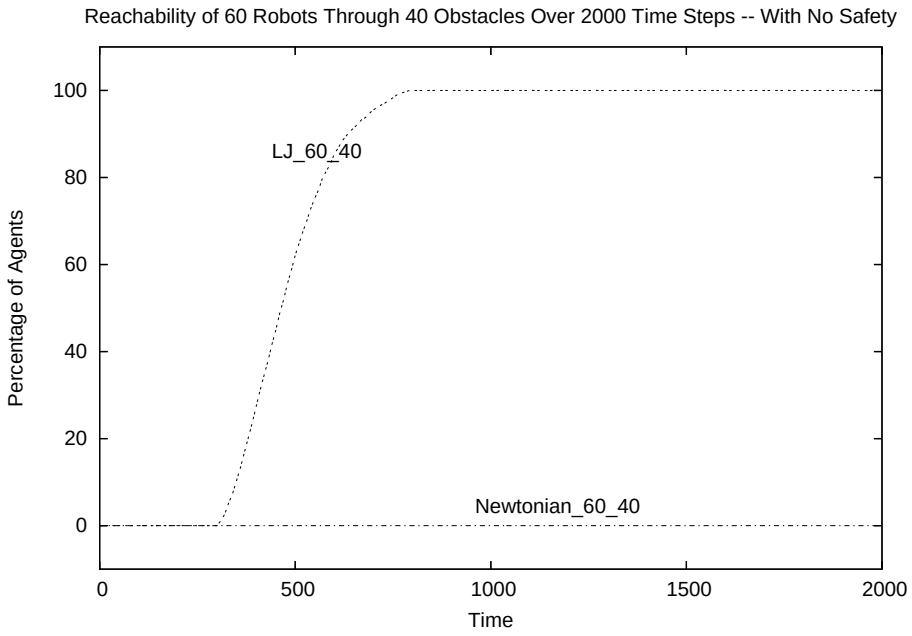

Figure 9.12: Change in reachability over 2000 time steps for 60 robots through 40 obstacles using Newtonian and LJ force laws

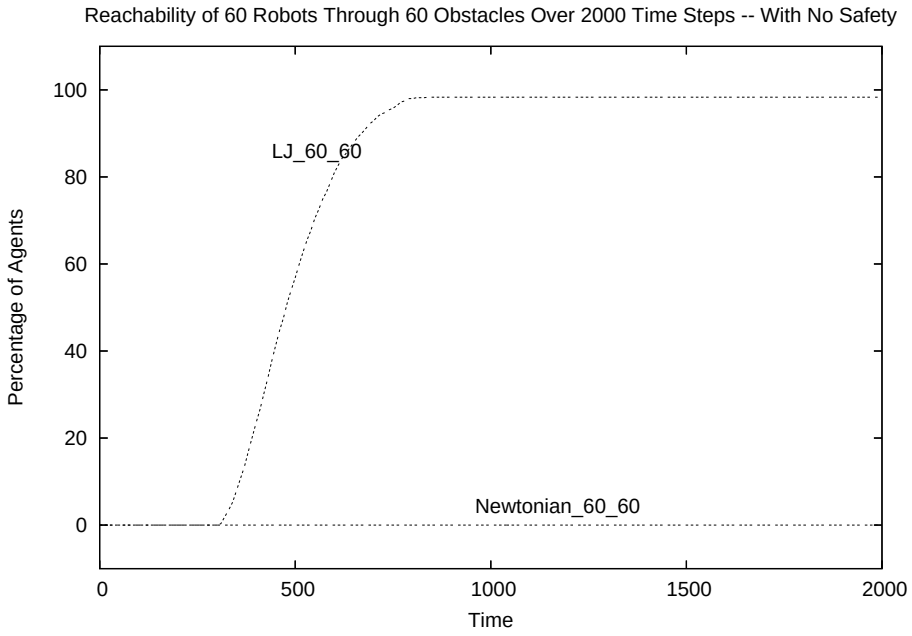

Figure 9.13: Change in reachability over 2000 time steps for 60 robots through 60 obstacles using Newtonian and LJ force laws

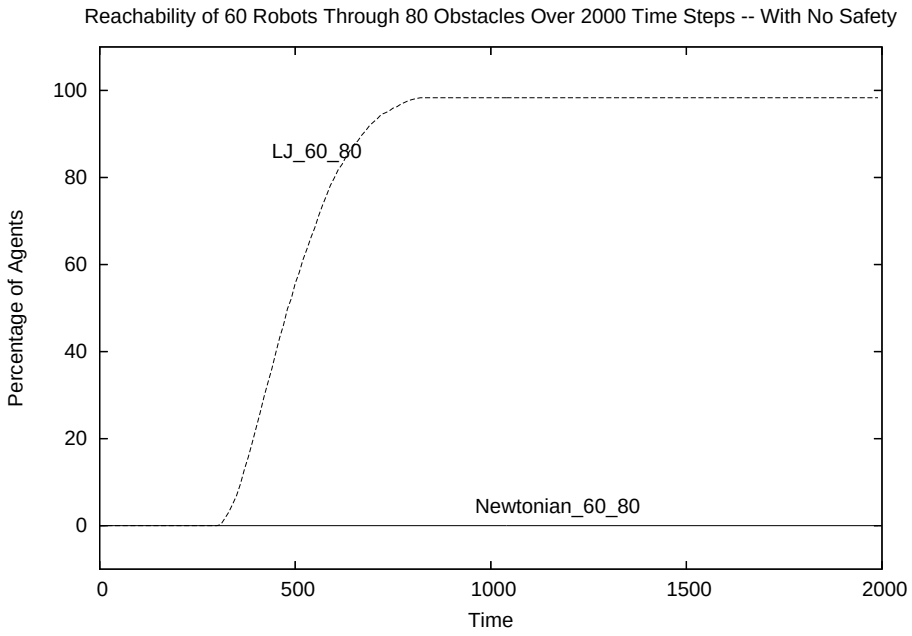

Figure 9.14: Change in reachability over 2000 time steps for 60 robots through 80 obstacles using Newtonian and LJ force laws

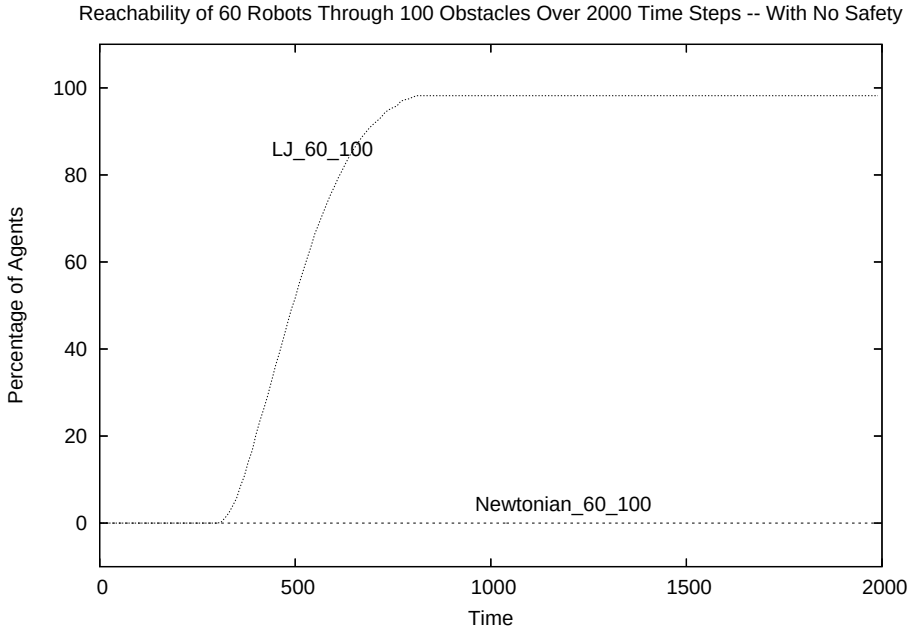

Figure 9.15: Change in reachability over 2000 time steps for 60 robots through 100 obstacles using Newtonian and LJ force laws

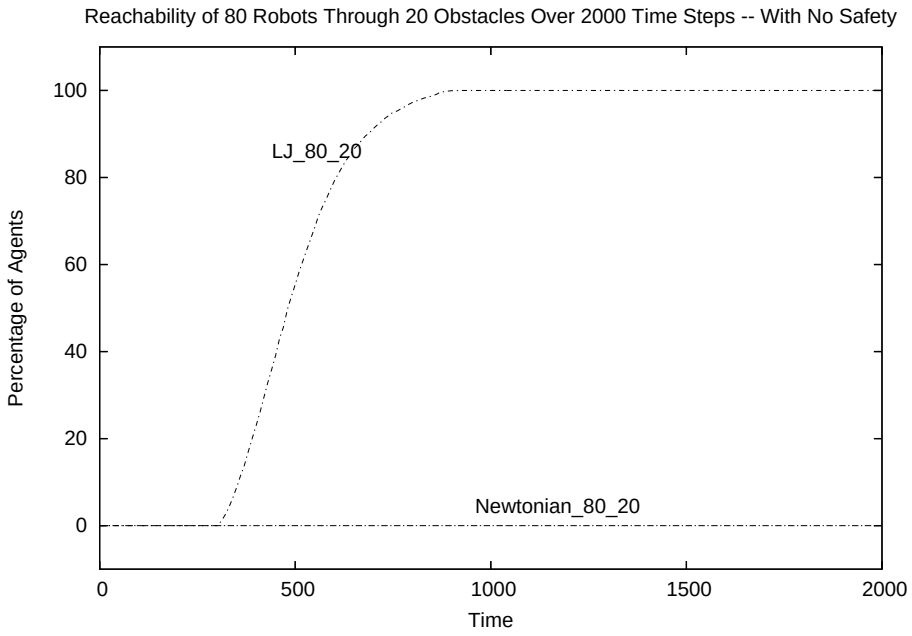

Figure 9.16: Change in reachability over 2000 time steps for 80 robots through 20 obstacles using Newtonian and LJ force laws

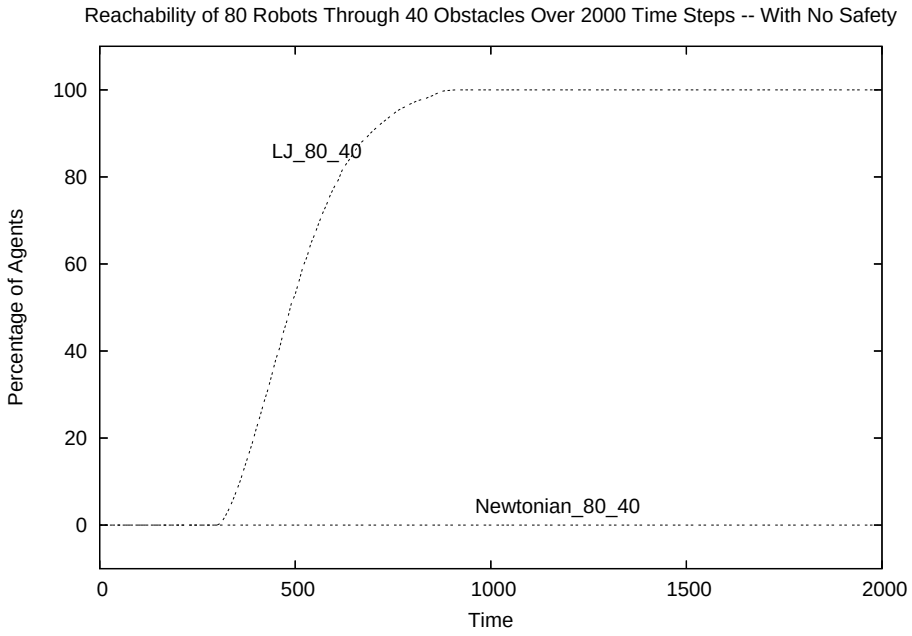

Figure 9.17: Change in reachability over 2000 time steps for 80 robots through 40 obstacles using Newtonian and LJ force laws

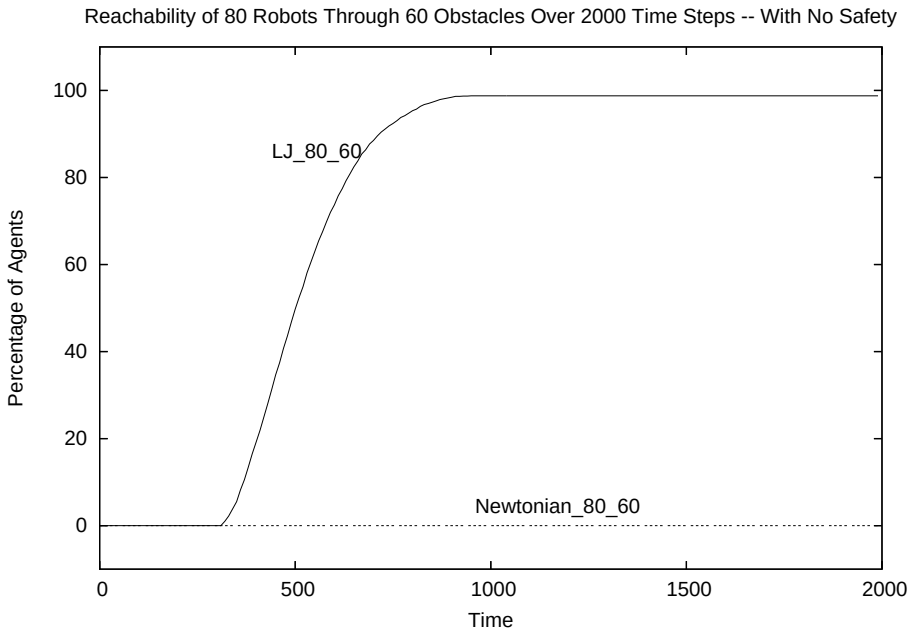

Figure 9.18: Change in reachability over 2000 time steps for 80 robots through 60 obstacles using Newtonian and LJ force laws

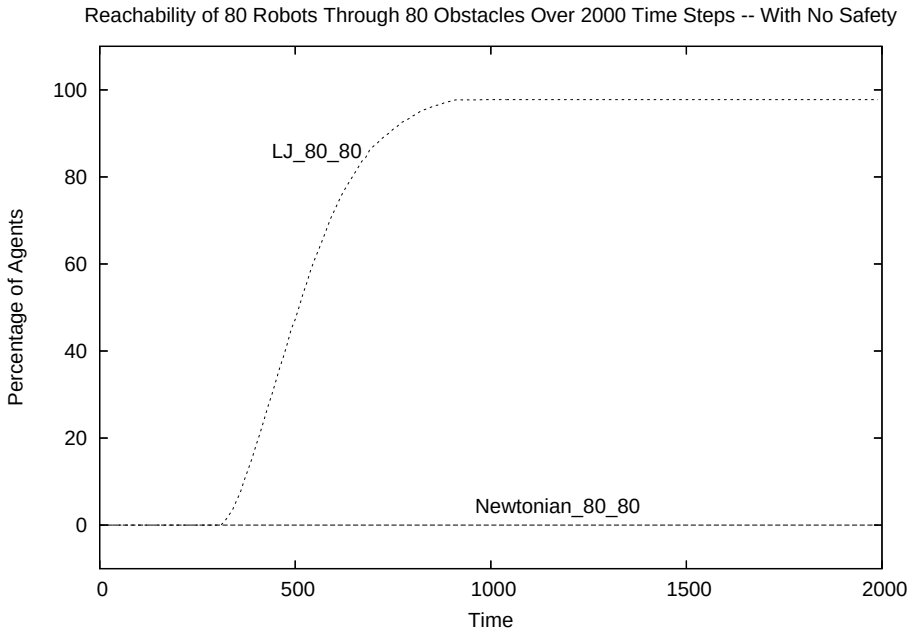

Figure 9.19: Change in reachability over 2000 time steps for 80 robots through 80 obstacles using Newtonian and LJ force laws

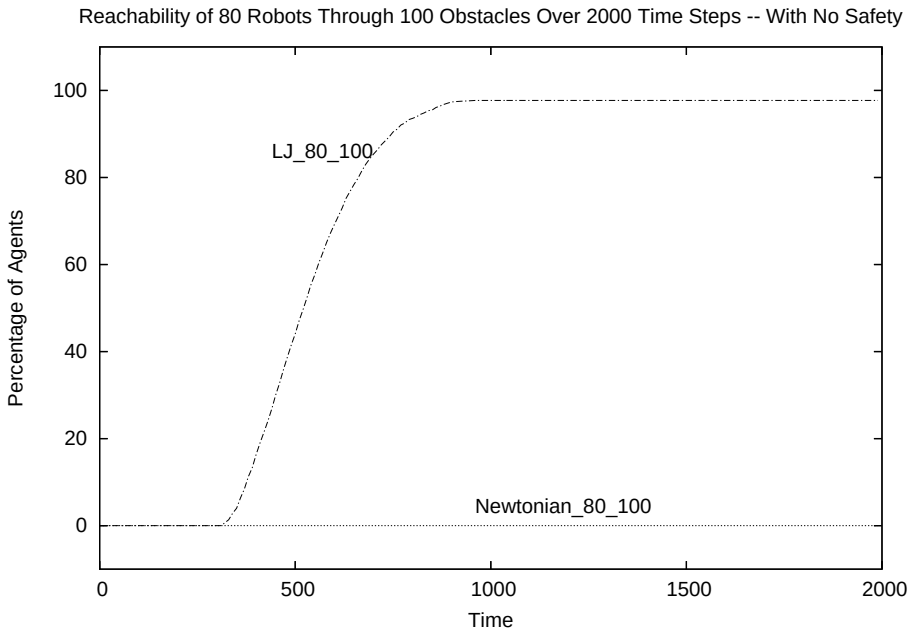

Figure 9.20: Change in reachability over 2000 time steps for 80 robots through 100 obstacles using Newtonian and LJ force laws

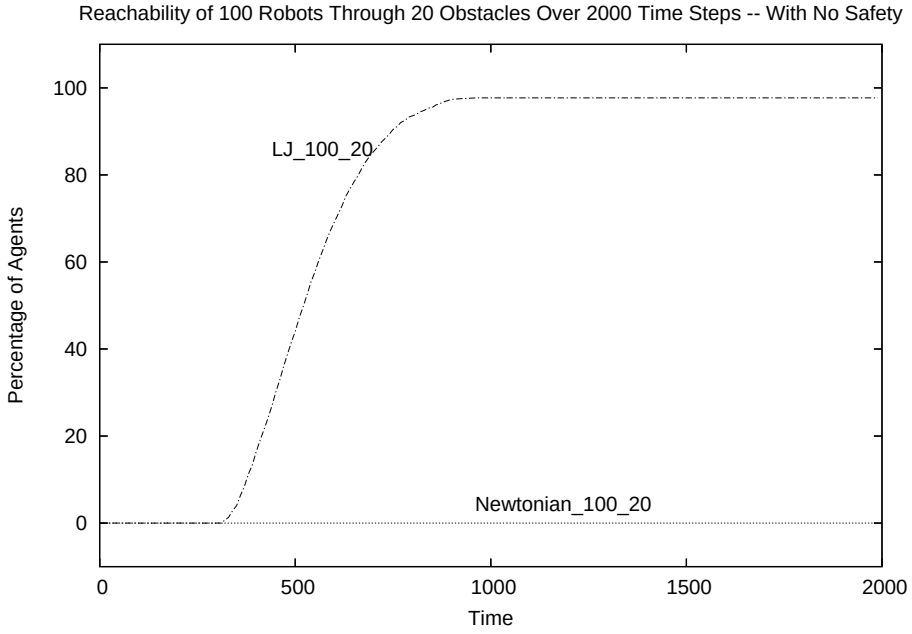

Figure 9.21: Change in reachability over 2000 time steps for 100 robots through 20 obstacles using Newtonian and LJ force laws

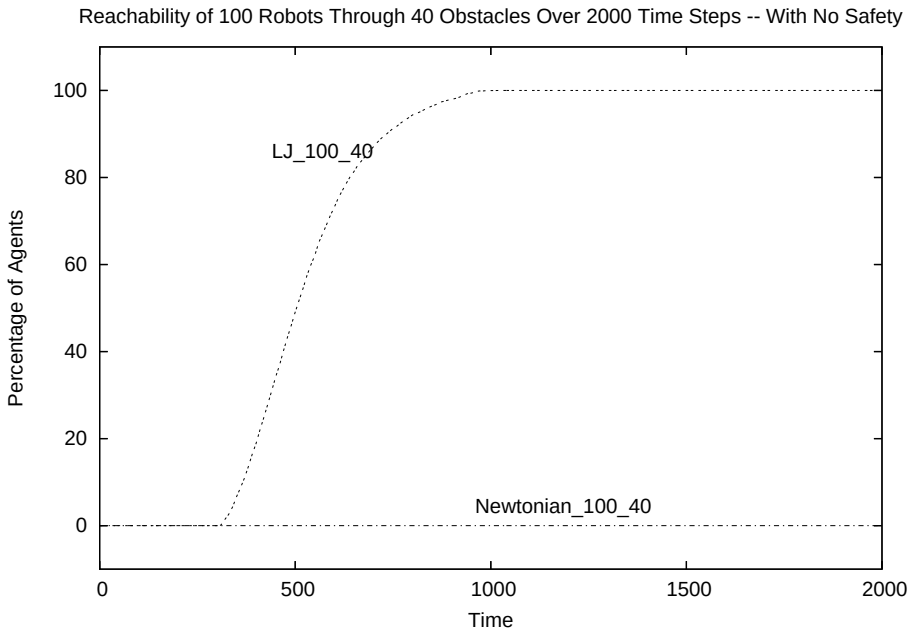

Figure 9.22: Change in reachability over 2000 time steps for 100 robots through 40 obstacles using Newtonian and LJ force laws

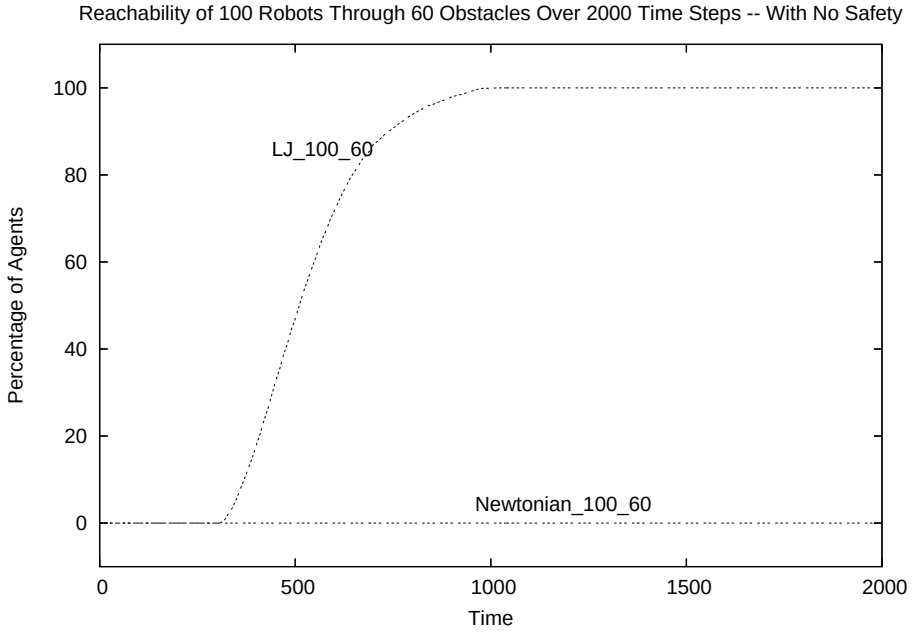

Figure 9.23: Change in reachability over 2000 time steps for 100 robots through 60 obstacles using Newtonian and LJ force laws

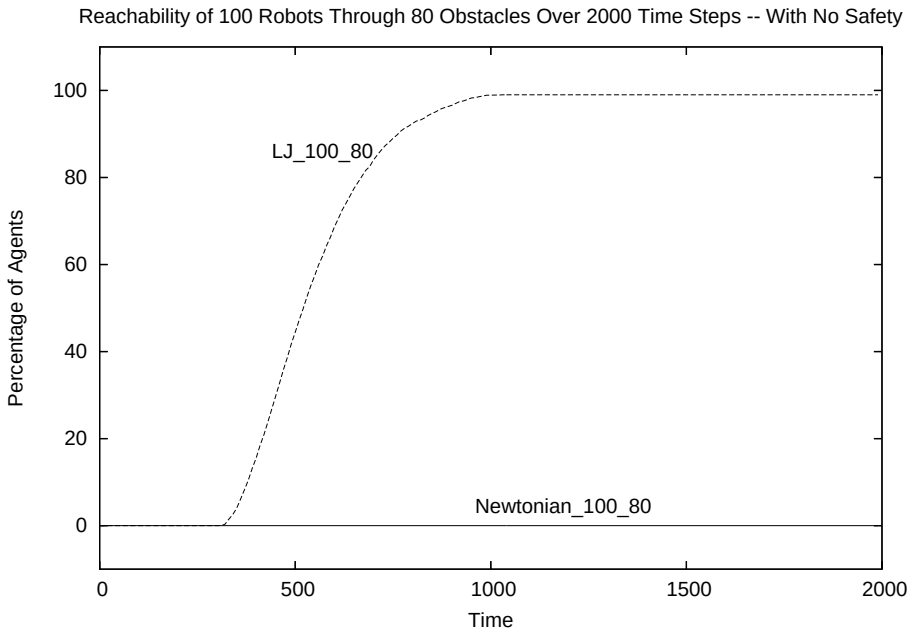

Figure 9.24: Change in reachability over 2000 time steps for 100 robots through 80 obstacles using Newtonian and LJ force laws

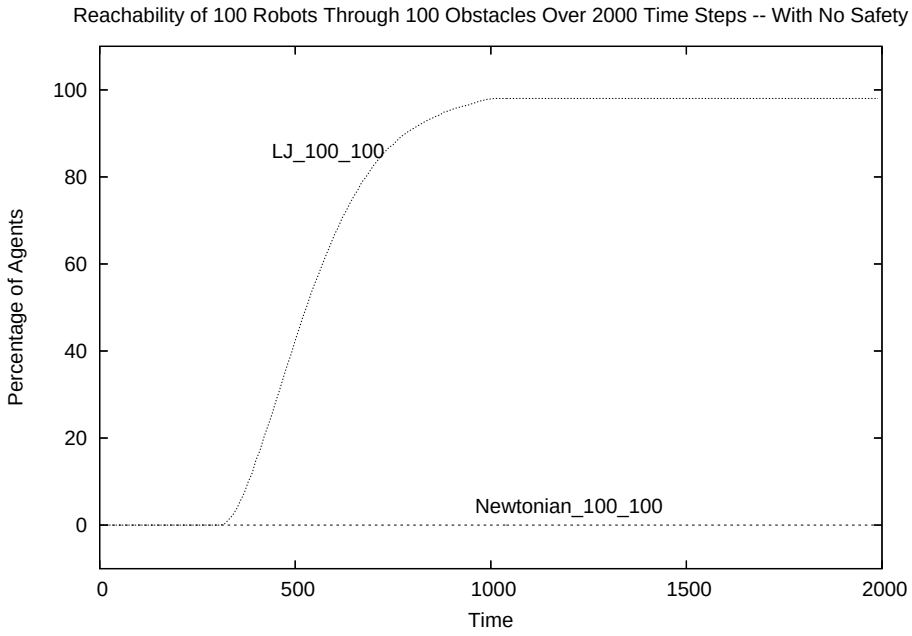

Figure 9.25: Change in reachability over 2000 time steps for 100 robots through 100 obstacles using Newtonian and LJ force laws

APPENDIX II

Complete Results of Connectivity for Offline Learning

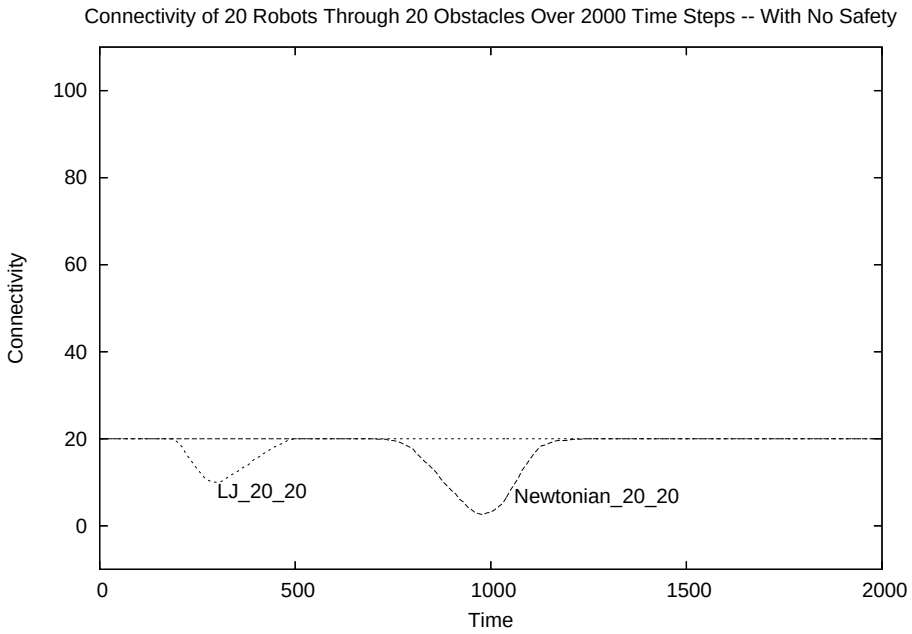

Figure 9.26: Change in connectivity over 2000 time steps for 20 robots through 20 obstacles using Newtonian and LJ force laws

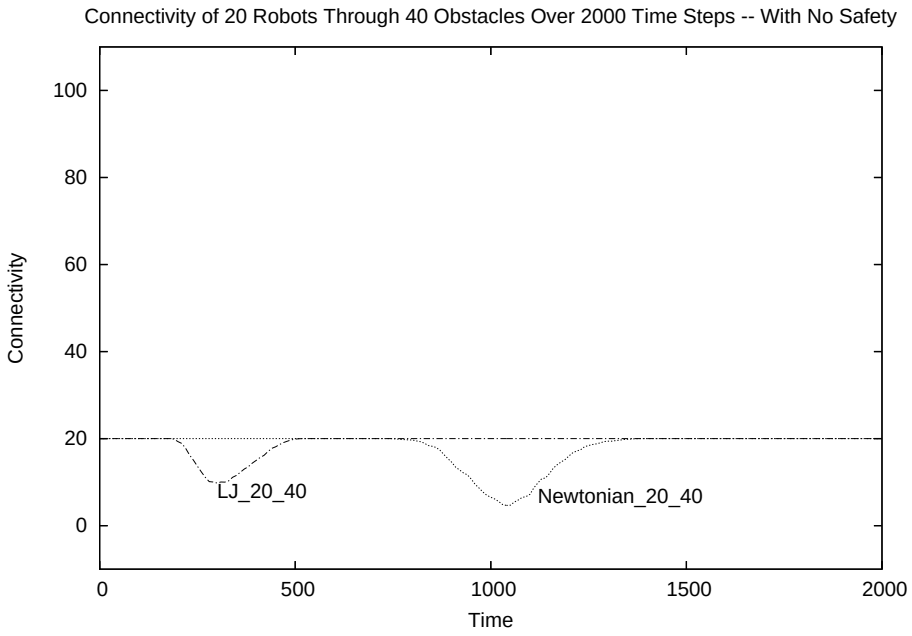

Figure 9.27: Change in connectivity over 2000 time steps for 20 robots through 40 obstacles using Newtonian and LJ force laws

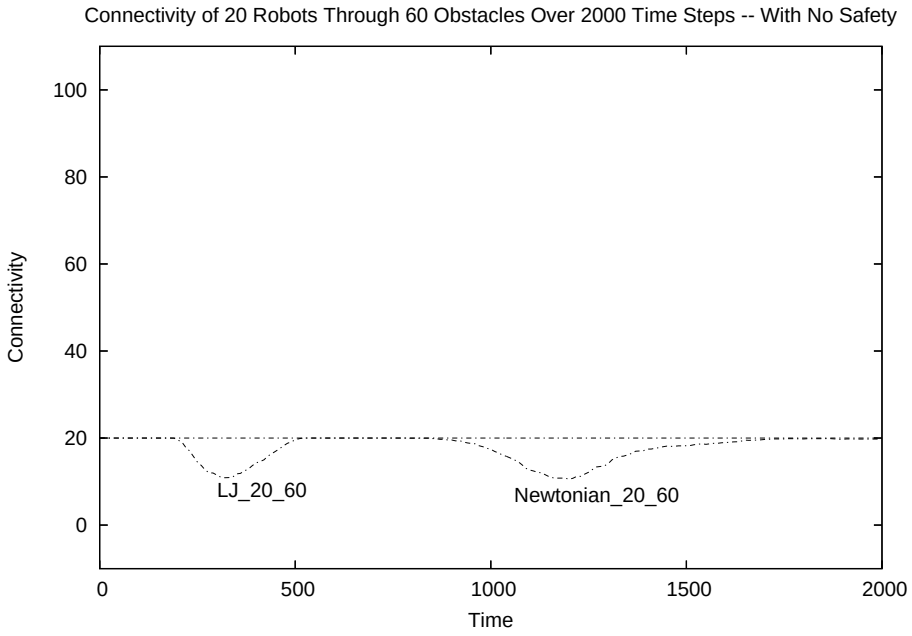

Figure 9.28: Change in connectivity over 2000 time steps for 20 robots through 60 obstacles using Newtonian and LJ force laws

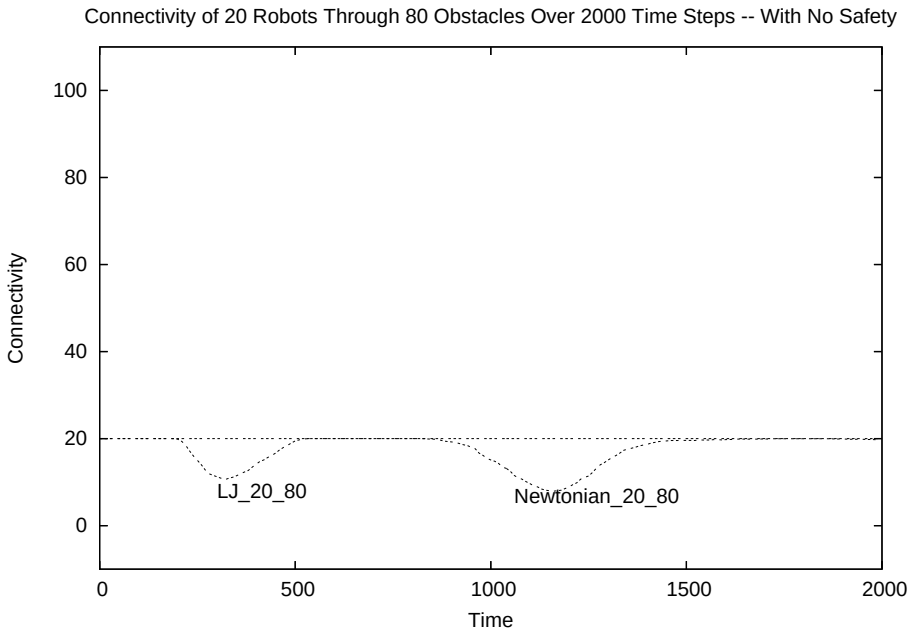

Figure 9.29: Change in connectivity over 2000 time steps for 20 robots through 80 obstacles using Newtonian and LJ force laws

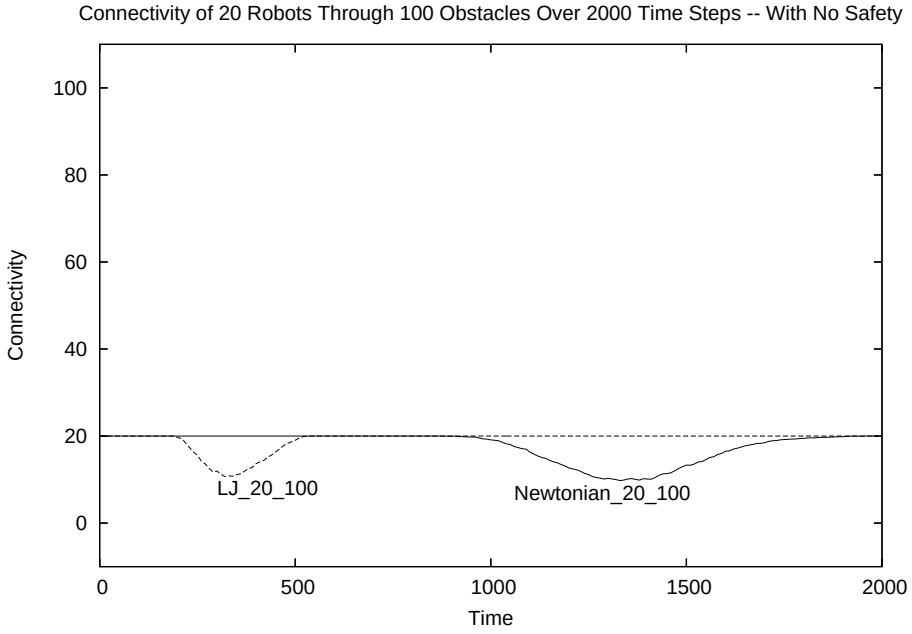

Figure 9.30: Change in connectivity over 2000 time steps for 20 robots through 100 obstacles using Newtonian and LJ force laws

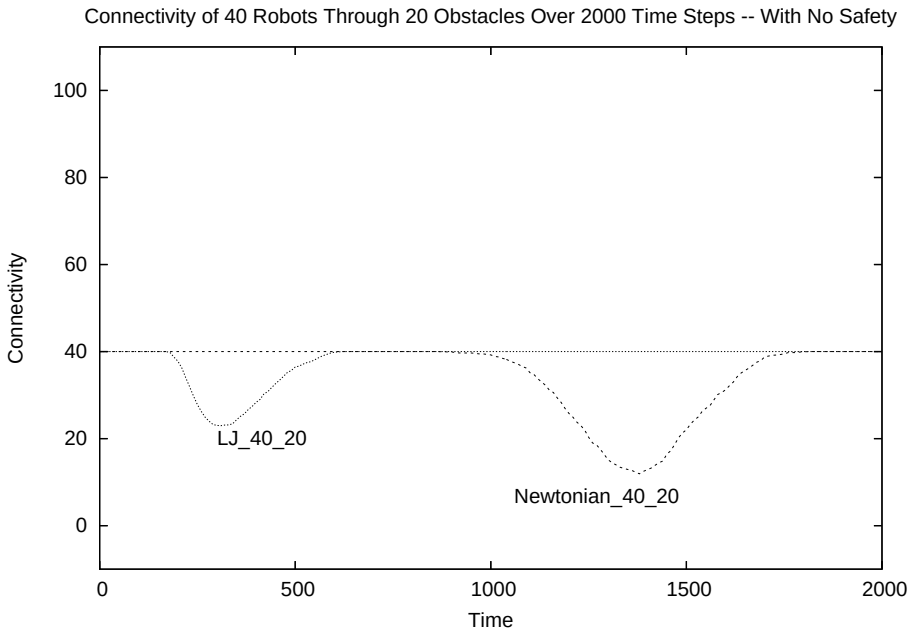

Figure 9.31: Change in connectivity over 2000 time steps for 40 robots through 20 obstacles using Newtonian and LJ force laws

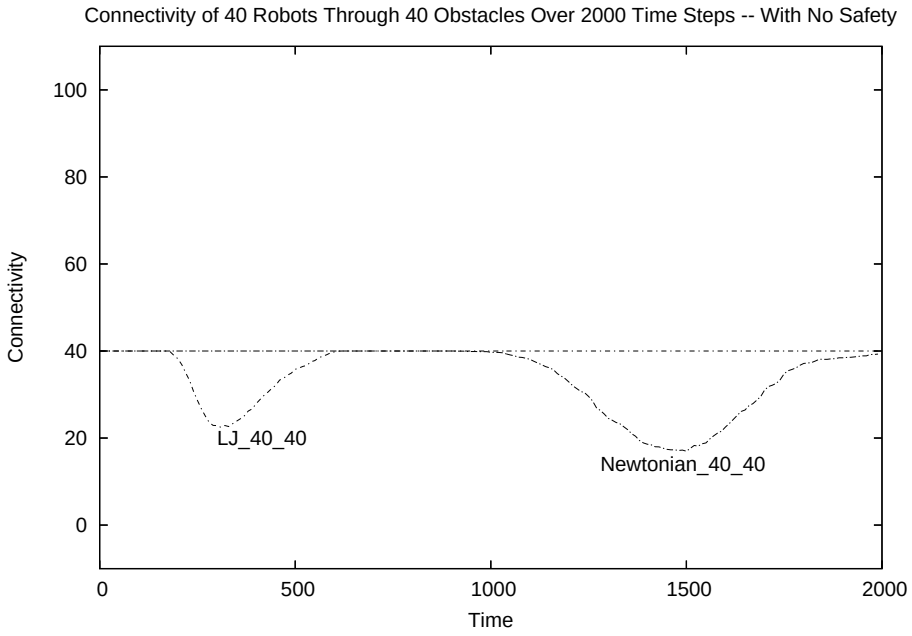

Figure 9.32: Change in connectivity over 2000 time steps for 40 robots through 40 obstacles using Newtonian and LJ force laws

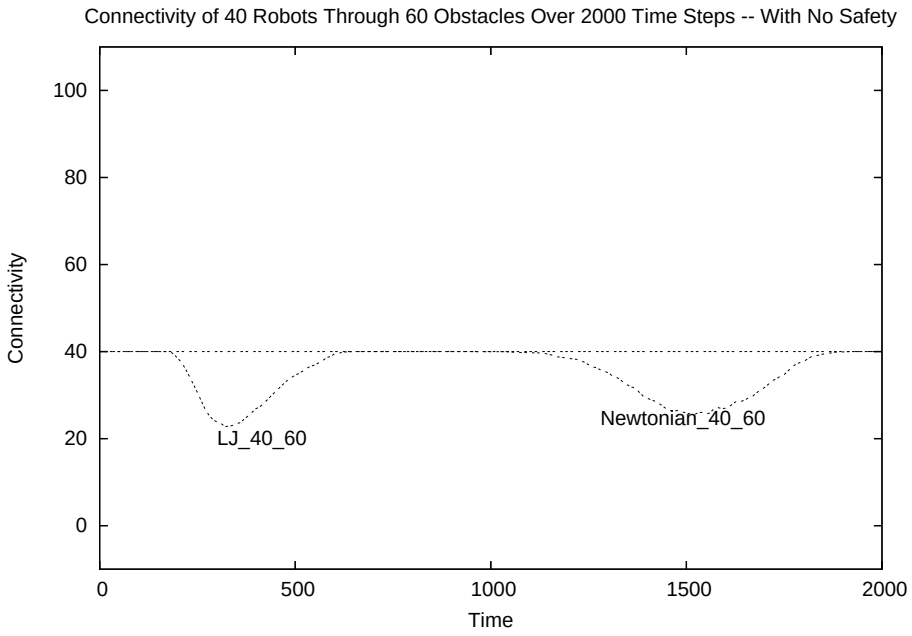

Figure 9.33: Change in connectivity over 2000 time steps for 40 robots through 60 obstacles using Newtonian and LJ force laws

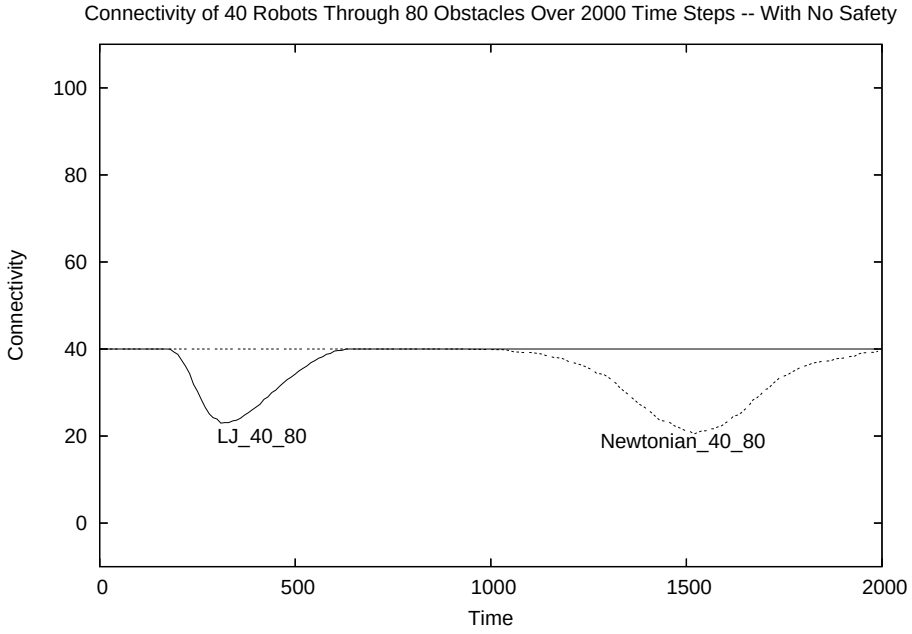

Figure 9.34: Change in connectivity over 2000 time steps for 40 robots through 80 obstacles using Newtonian and LJ force laws

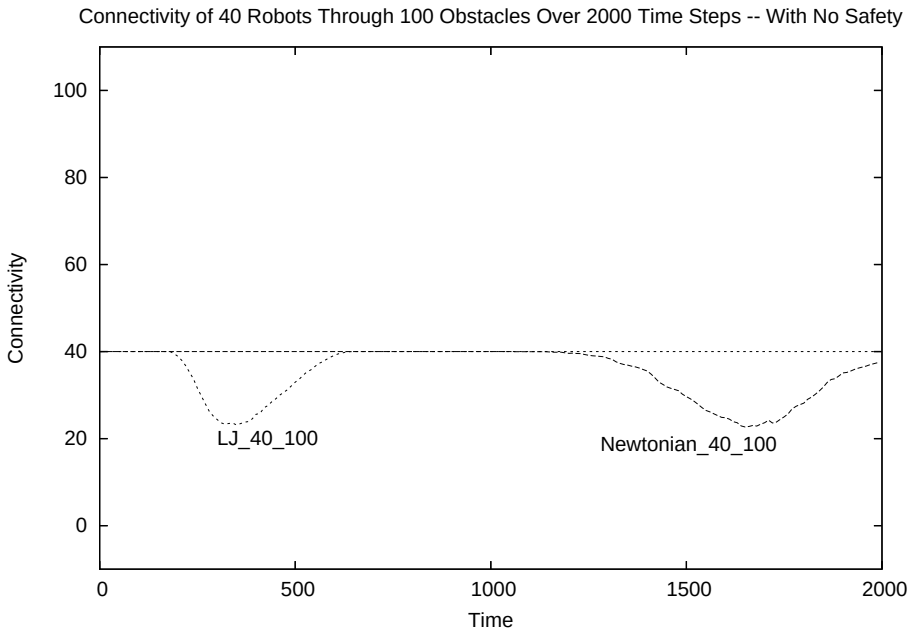

Figure 9.35: Change in connectivity over 2000 time steps for 40 robots through 100 obstacles using Newtonian and LJ force laws

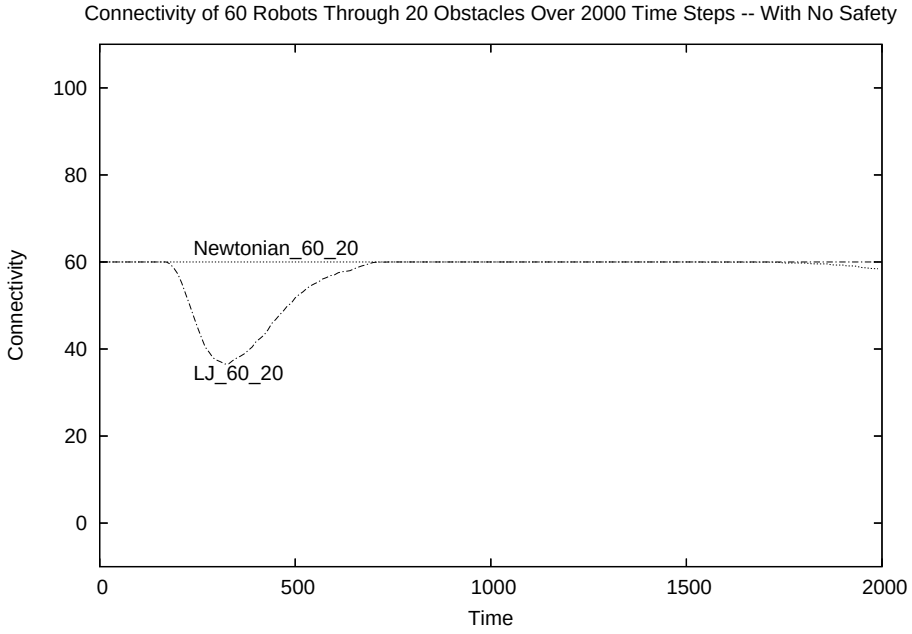

Figure 9.36: Change in connectivity over 2000 time steps for 60 robots through 20 obstacles using Newtonian and LJ force laws

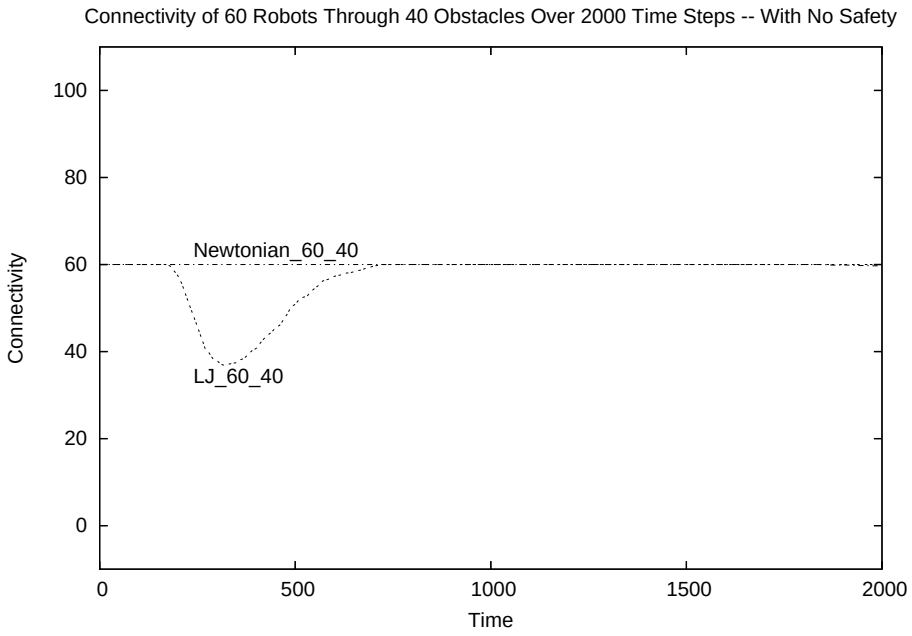

Figure 9.37: Change in connectivity over 2000 time steps for 60 robots through 40 obstacles using Newtonian and LJ force laws

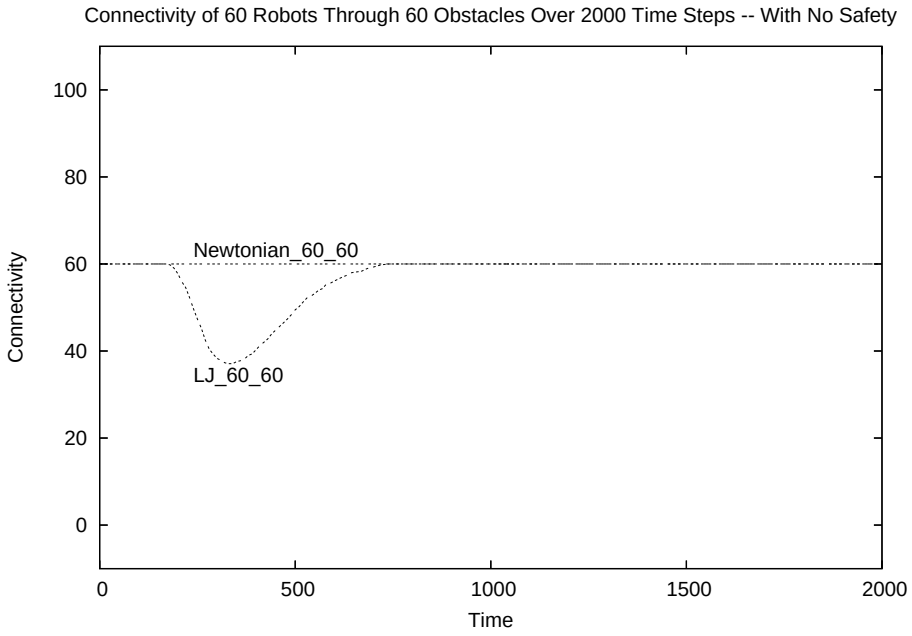

Figure 9.38: Change in connectivity over 2000 time steps for 60 robots through 60 obstacles using Newtonian and LJ force laws

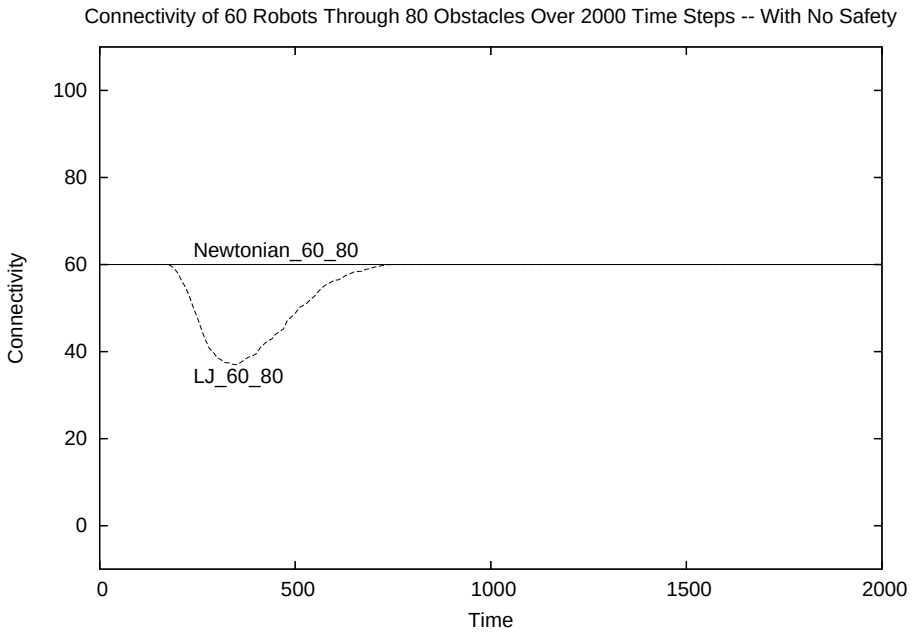

Figure 9.39: Change in connectivity over 2000 time steps for 60 robots through 80 obstacles using Newtonian and LJ force laws

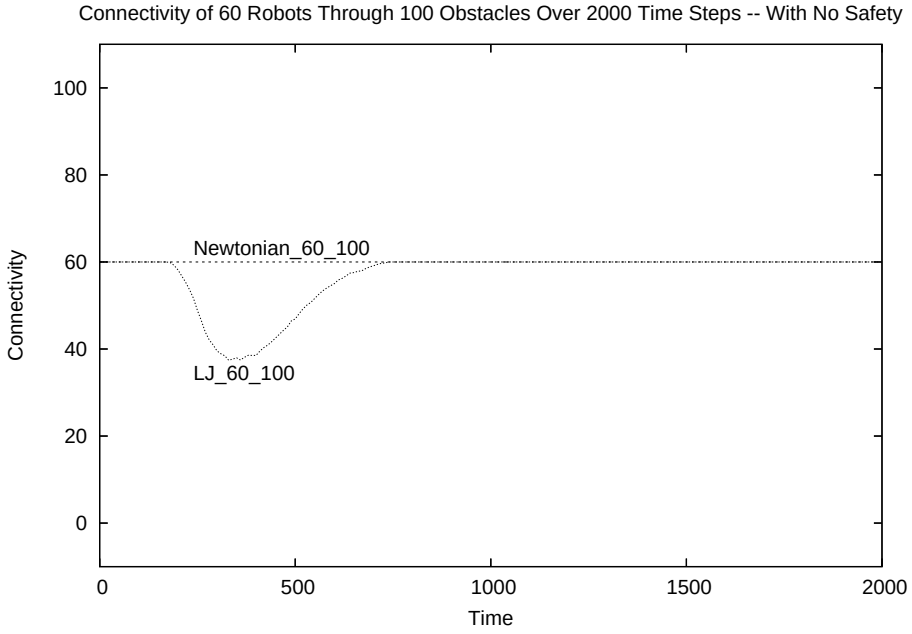

Figure 9.40: Change in connectivity over 2000 time steps for 60 robots through 100 obstacles using Newtonian and LJ force laws

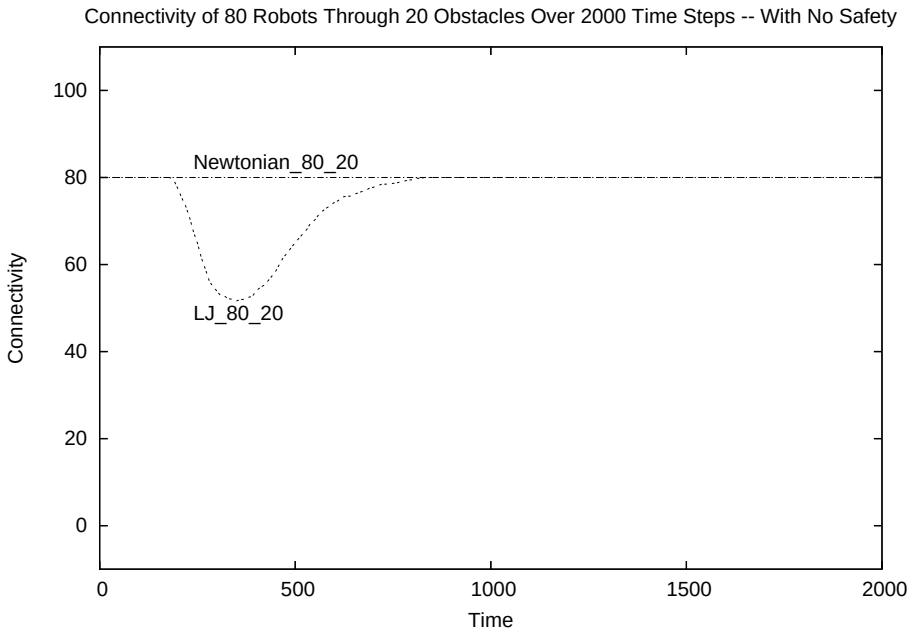

Figure 9.41: Change in connectivity over 2000 time steps for 80 robots through 20 obstacles using Newtonian and LJ force laws

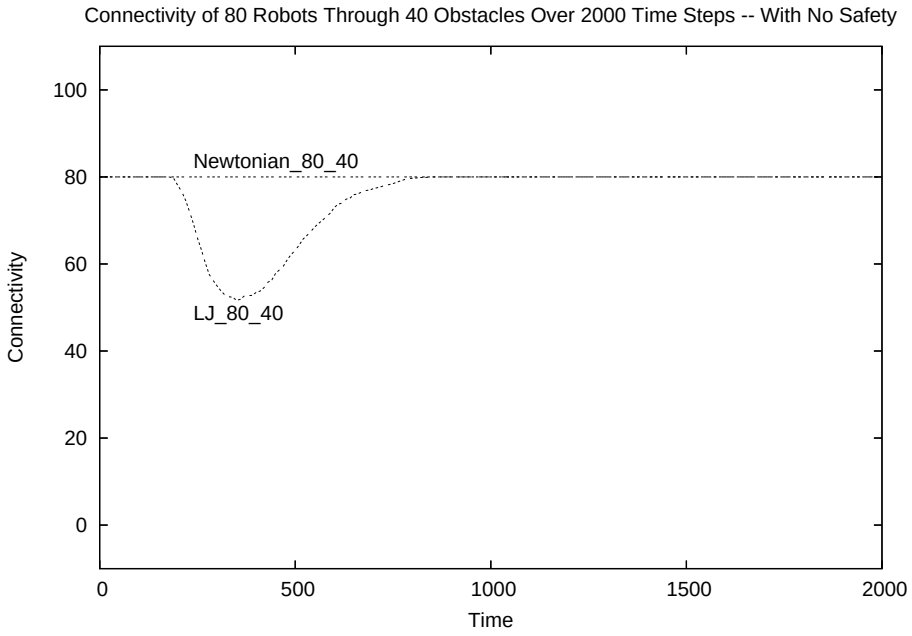

Figure 9.42: Change in connectivity over 2000 time steps for 80 robots through 40 obstacles using Newtonian and LJ force laws

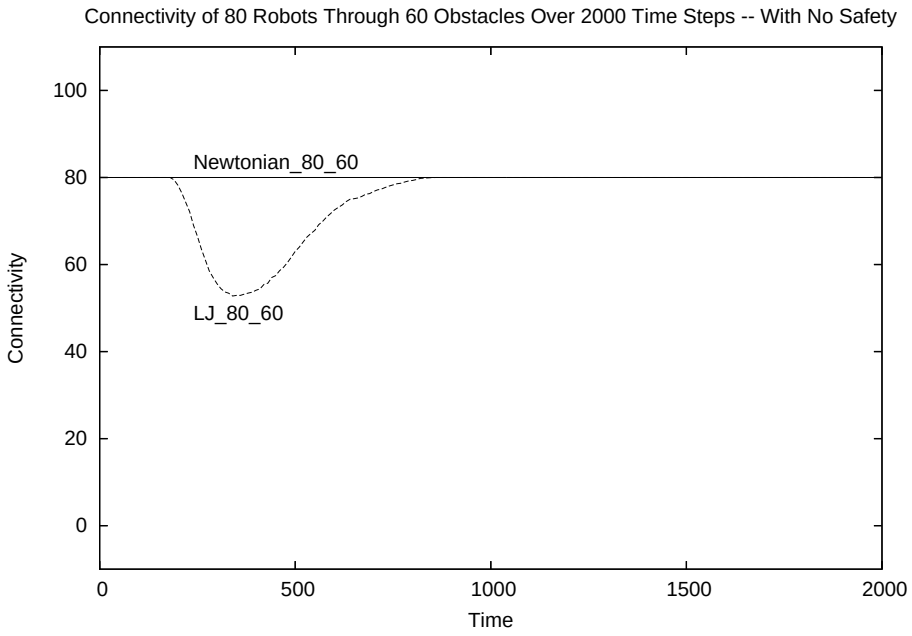

Figure 9.43: Change in connectivity over 2000 time steps for 80 robots through 60 obstacles using Newtonian and LJ force laws

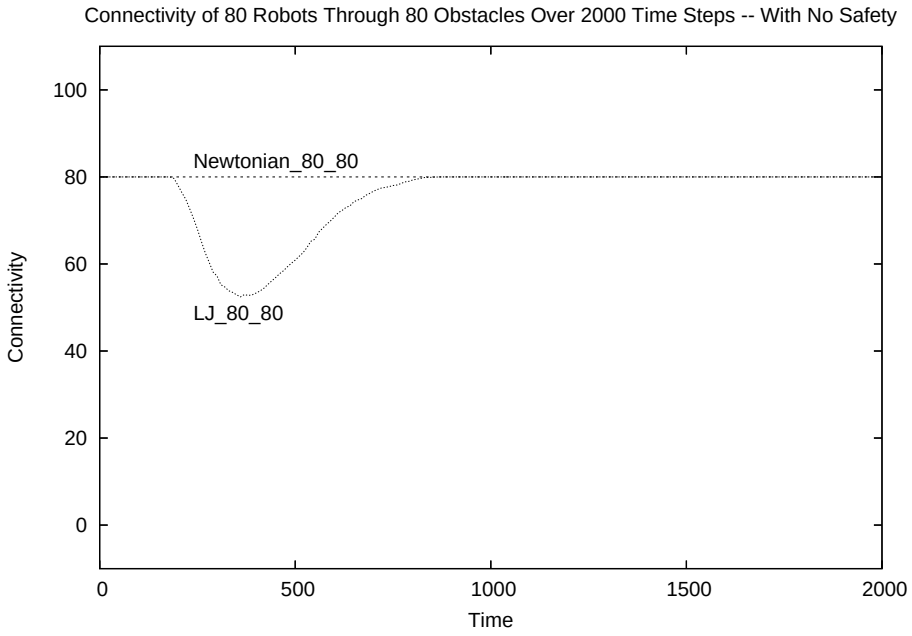

Figure 9.44: Change in connectivity over 2000 time steps for 80 robots through 80 obstacles using Newtonian and LJ force laws

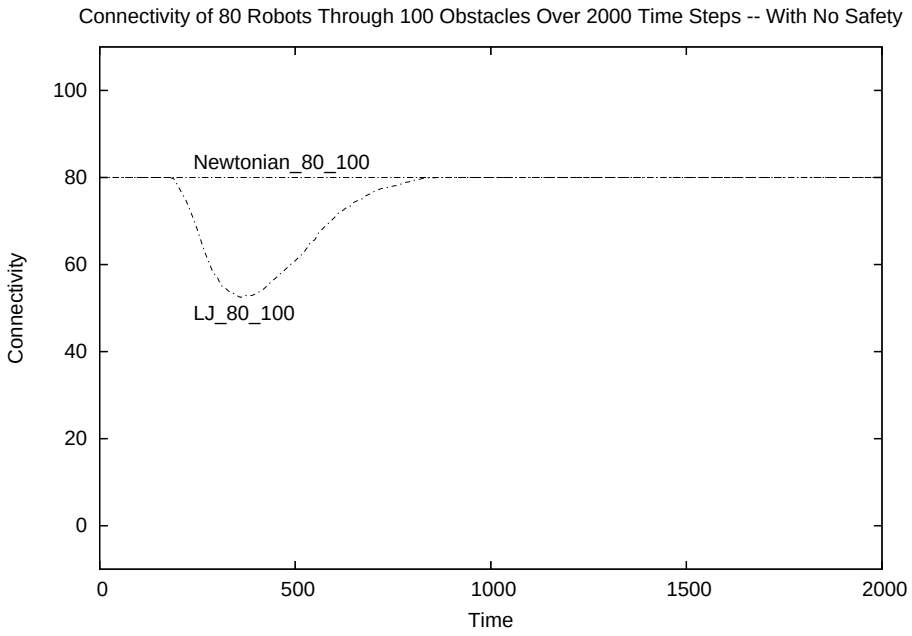

Figure 9.45: Change in connectivity over 2000 time steps for 80 robots through 100 obstacles using Newtonian and LJ force laws

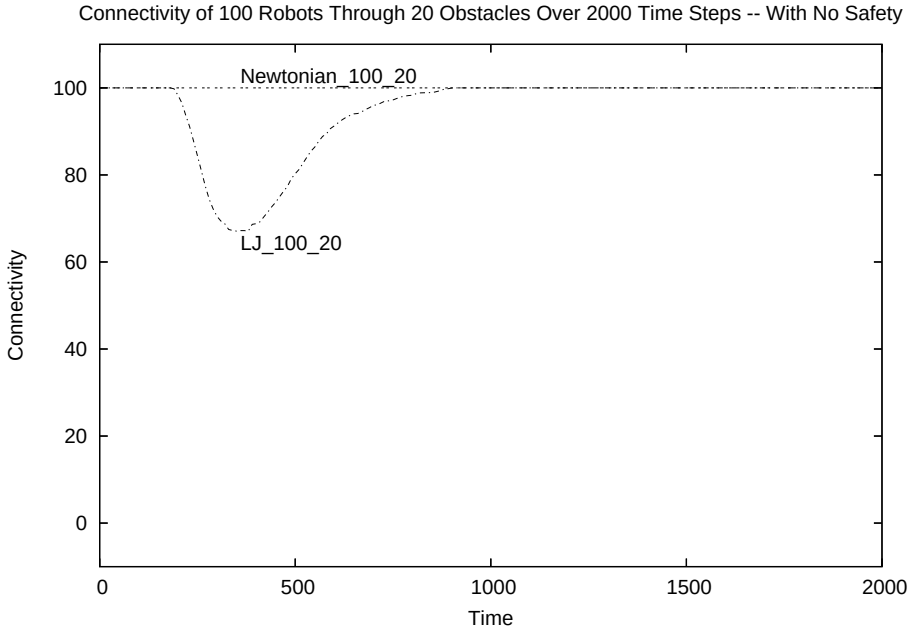

Figure 9.46: Change in connectivity over 2000 time steps for 100 robots through 20 obstacles using Newtonian and LJ force laws

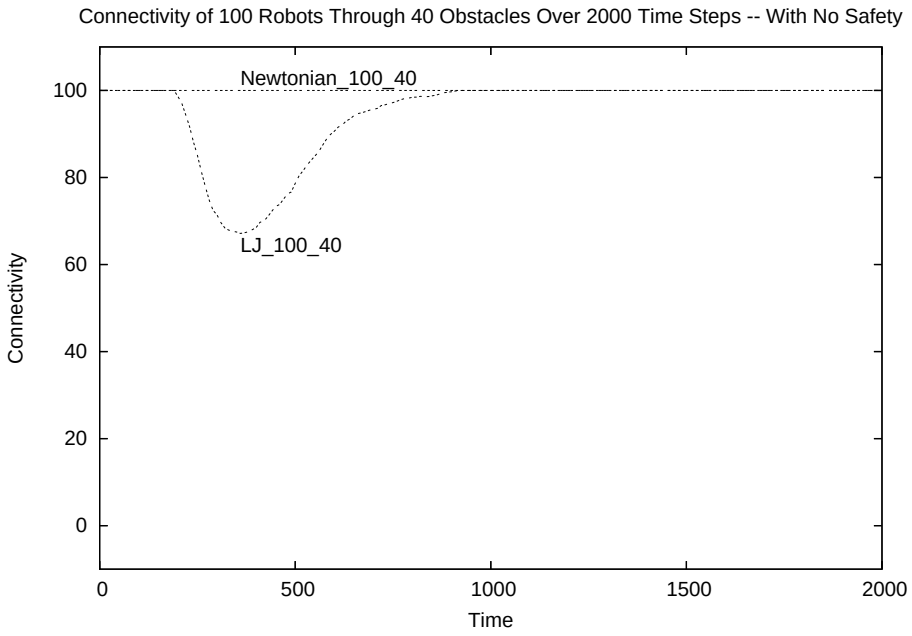

Figure 9.47: Change in connectivity over 2000 time steps for 100 robots through 40 obstacles using Newtonian and LJ force laws

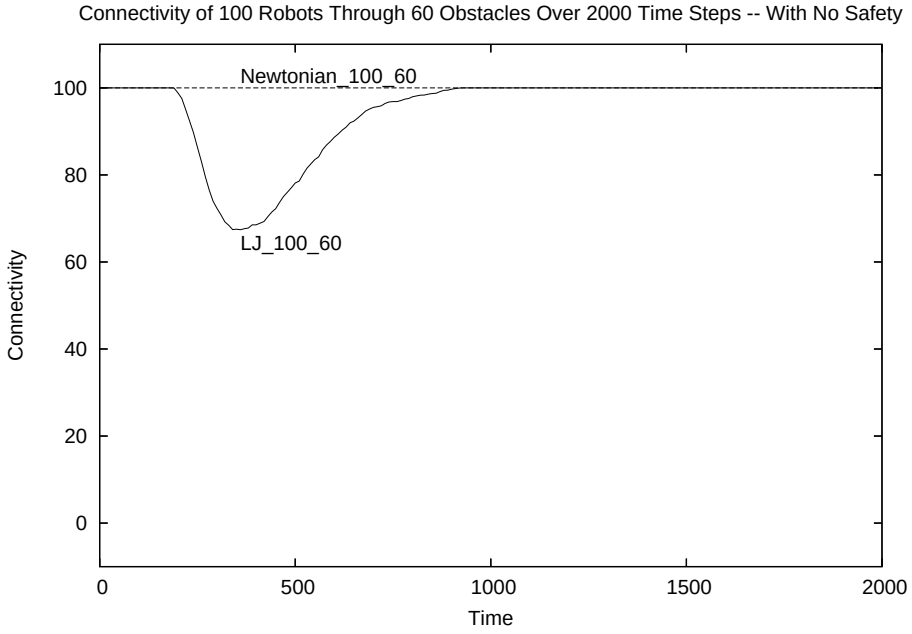

Figure 9.48: Change in connectivity over 2000 time steps for 100 robots through 60 obstacles using Newtonian and LJ force laws

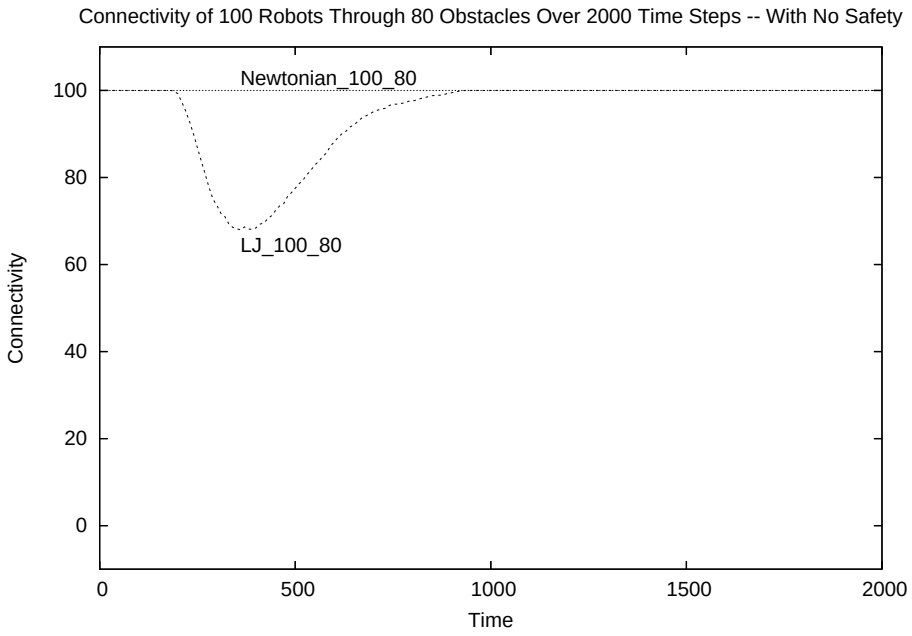

Figure 9.49: Change in connectivity over 2000 time steps for 100 robots through 80 obstacles using Newtonian and LJ force laws

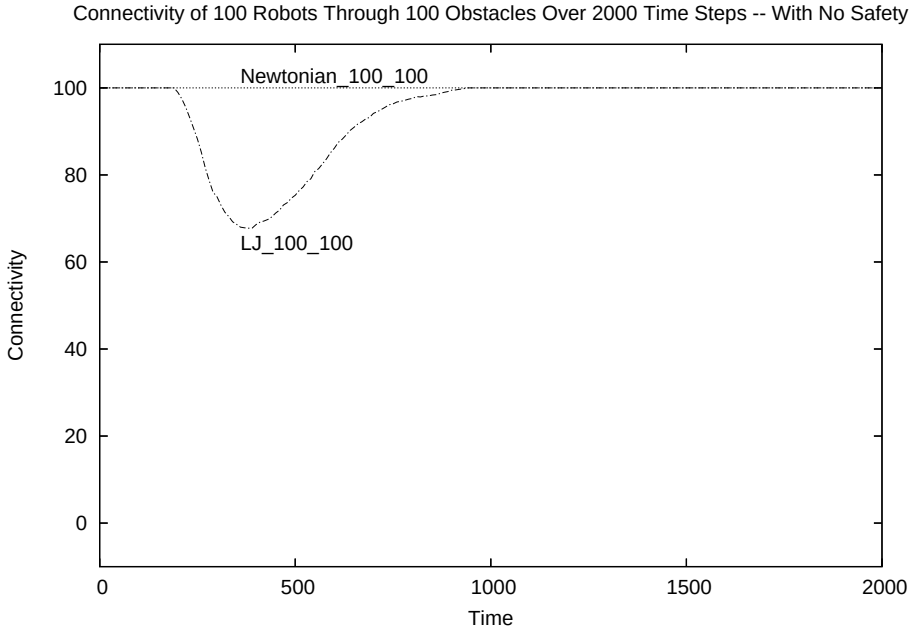

Figure 9.50: Change in connectivity over 2000 time steps for 100 robots through 100 obstacles using Newtonian and LJ force laws

APPENDIX III

Complete Results for Newtonian and LJ Force Laws with a Safety Zone for Offline Learning

	Obstacles				
robots	20	40	60	80	100
20	96%	3%	1%	0%	0%
40	5%	0%	0%	0%	0%
60	0%	0%	0%	0%	0%
80	0%	0%	0%	0%	0%
100	0%	0%	0%	0%	0%

Table 9.1: Percentage of robots reaching the goal using Newtonian force law

	Obstacles				
robots	20	40	60	80	100
20	1850	–	–	–	–
40	–	–	–	–	–
60	–	–	–	–	–
80	–	–	–	–	–
100	–	–	–	–	–

Table 9.2: Time taken by 80% of robots to reach the goal using Newtonian force law

	Obstacles				
robots	20	40	60	80	100
20	100%	100%	73%	60%	63%
40	100%	100%	85%	74%	72%
60	100%	99%	87%	81%	74%
80	100%	99%	90%	86%	79%
100	100%	99%	90%	86%	80%

Table 9.3: Percentage of robots reaching the goal using LJ force law

	Obstacles				
robots	20	40	60	80	100
20	610	660	–	–	–
40	680	740	940	–	–
60	740	810	970	1300	–
80	780	860	1000	1170	–
100	820	900	1050	1200	1740

Table 9.4: Time taken by 80% of robots to reach the goal using LJ force law

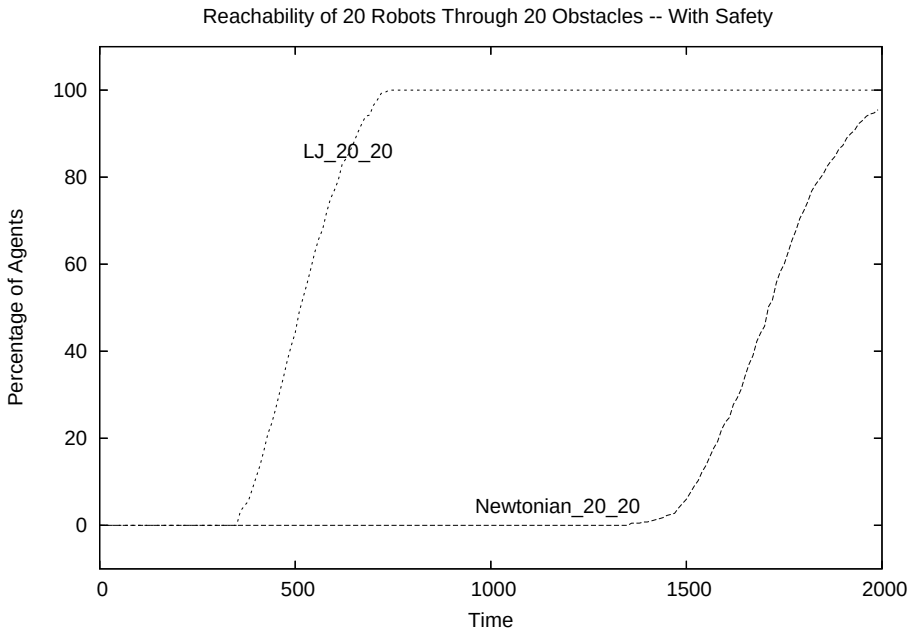

Figure 9.51: Change in reachability over 2000 time steps for 20 robots through 20 obstacles using Newtonian and LJ force laws with a safety zone

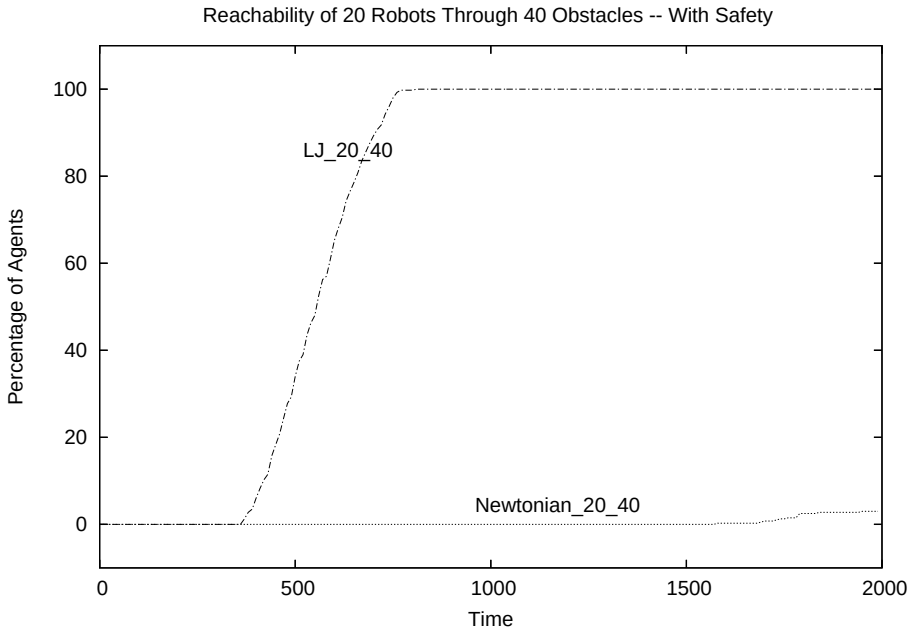

Figure 9.52: Change in reachability over 2000 time steps for 20 robots through 40 obstacles using Newtonian and LJ force laws with a safety zone

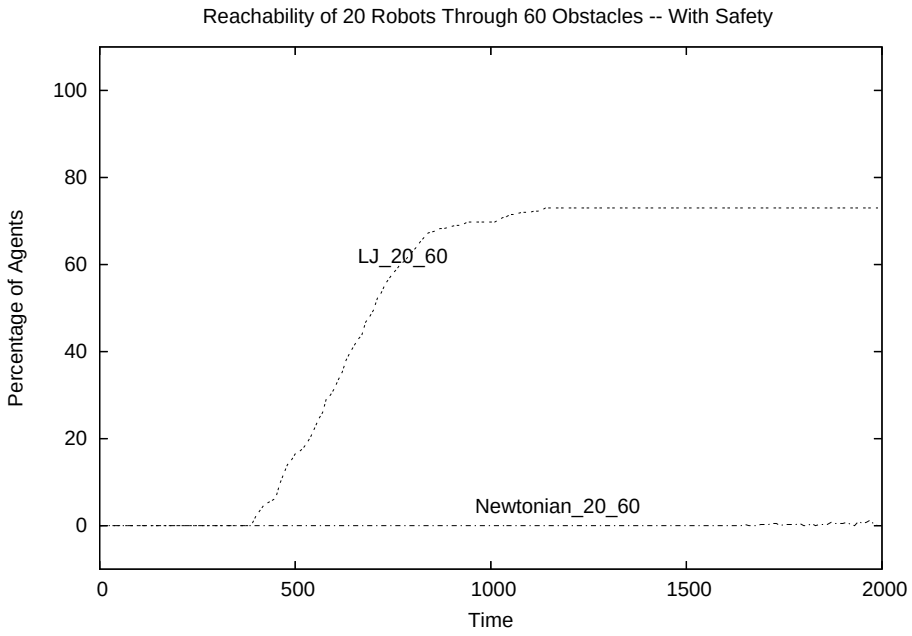

Figure 9.53: Change in reachability over 2000 time steps for 20 robots through 60 obstacles using Newtonian and LJ force laws with a safety zone

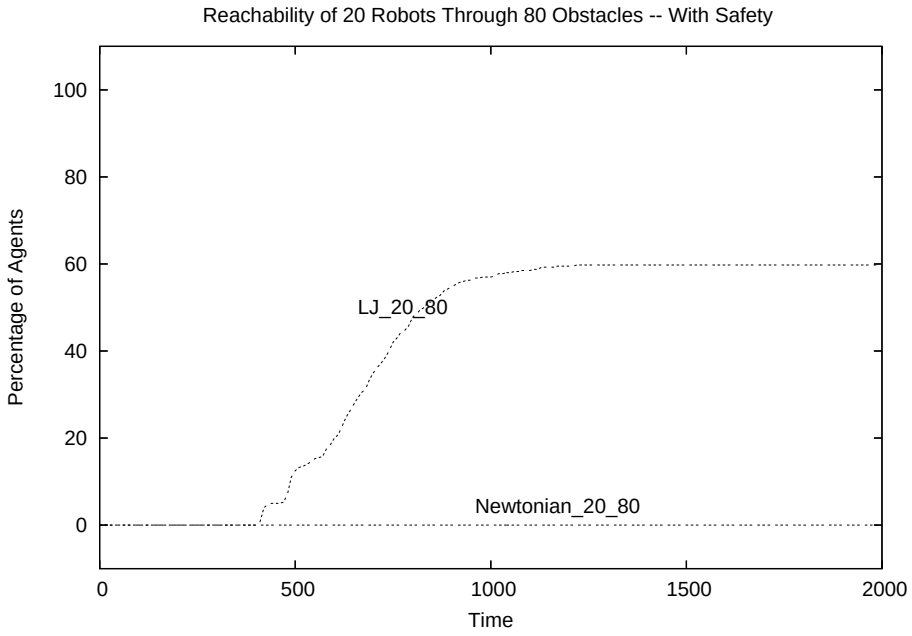

Figure 9.54: Change in reachability over 2000 time steps for 20 robots through 80 obstacles using Newtonian and LJ force laws with a safety zone

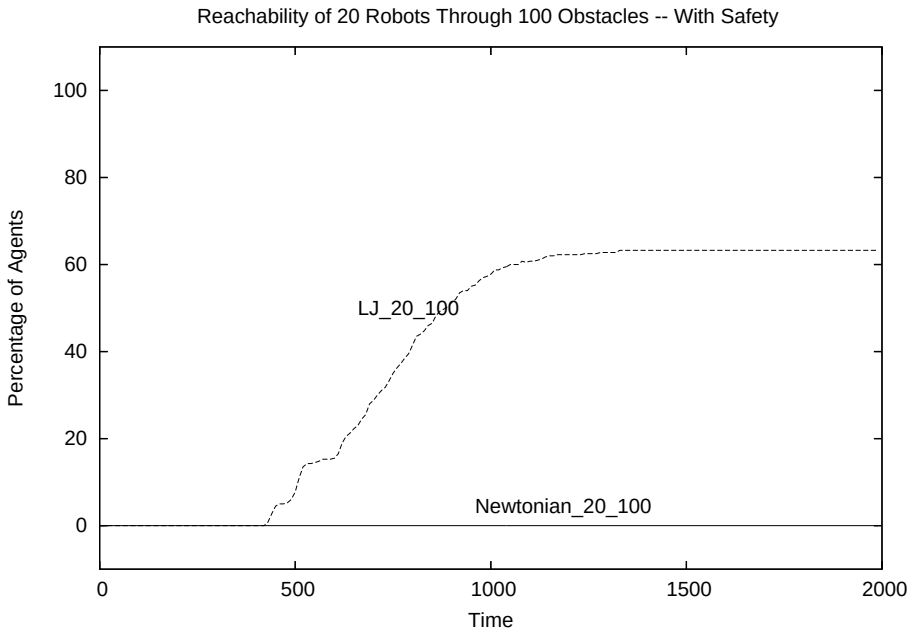

Figure 9.55: Change in reachability over 2000 time steps for 20 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone

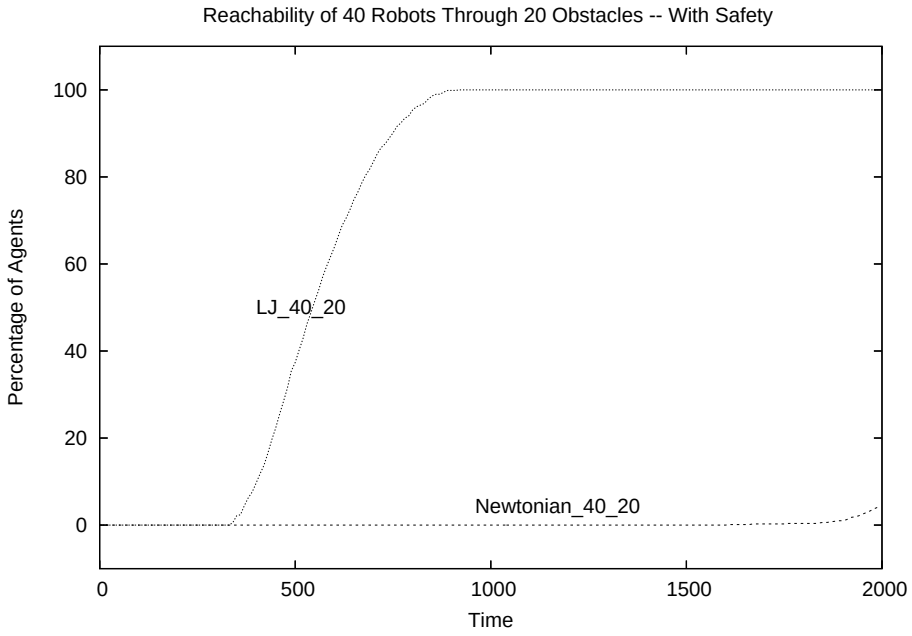

Figure 9.56: Change in reachability over 2000 time steps for 40 robots through 20 obstacles using Newtonian and LJ force laws with a safety zone

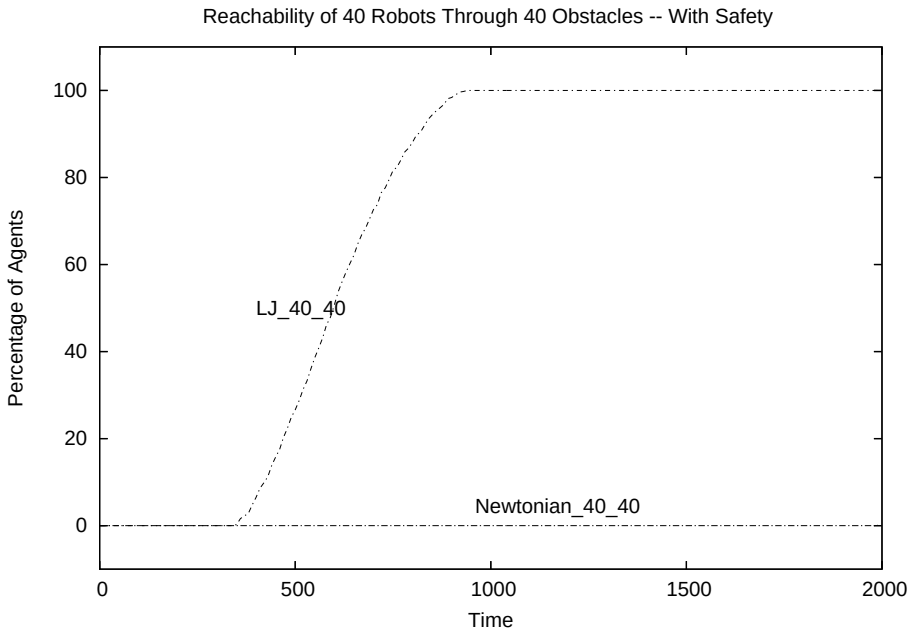

Figure 9.57: Change in reachability over 2000 time steps for 40 robots through 40 obstacles using Newtonian and LJ force laws with a safety zone

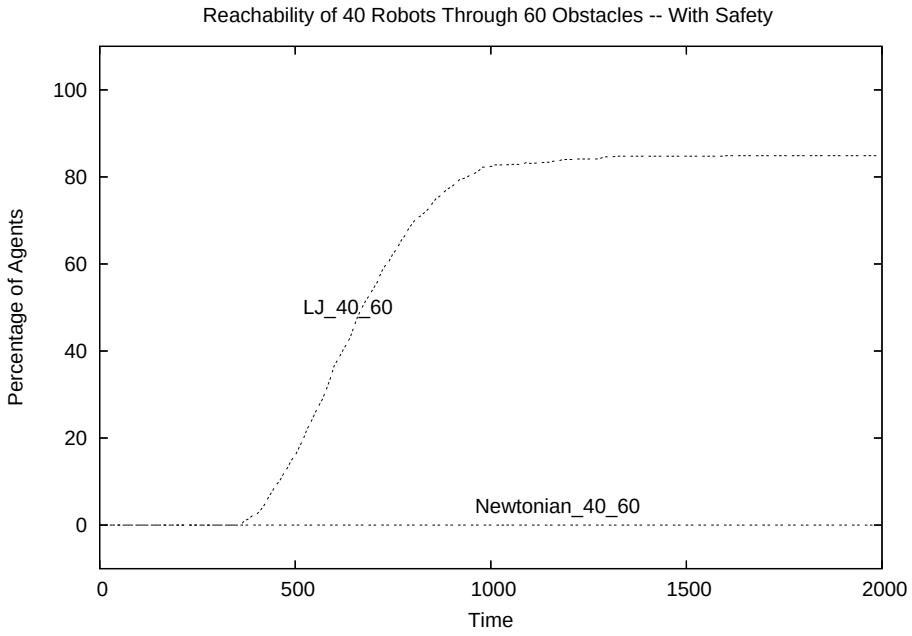

Figure 9.58: Change in reachability over 2000 time steps for 40 robots through 60 obstacles using Newtonian and LJ force laws with a safety zone

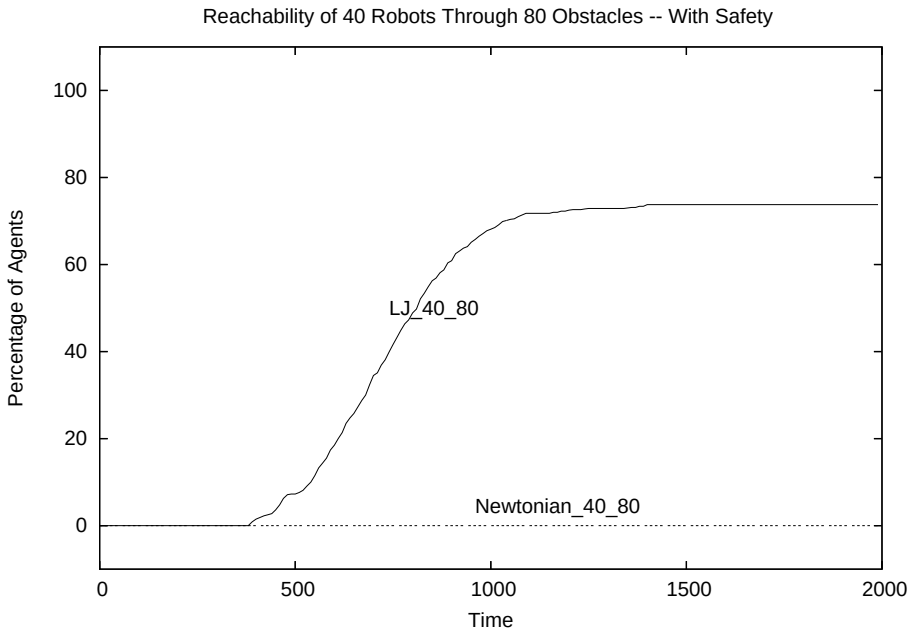

Figure 9.59: Change in reachability over 2000 time steps for 40 robots through 80 obstacles using Newtonian and LJ force laws with a safety zone

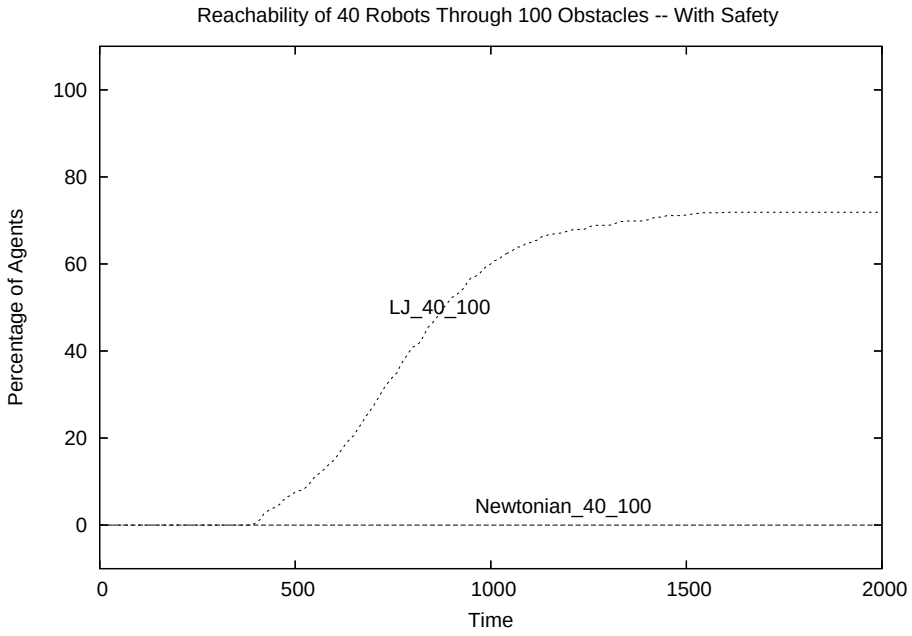

Figure 9.60: Change in reachability over 2000 time steps for 40 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone

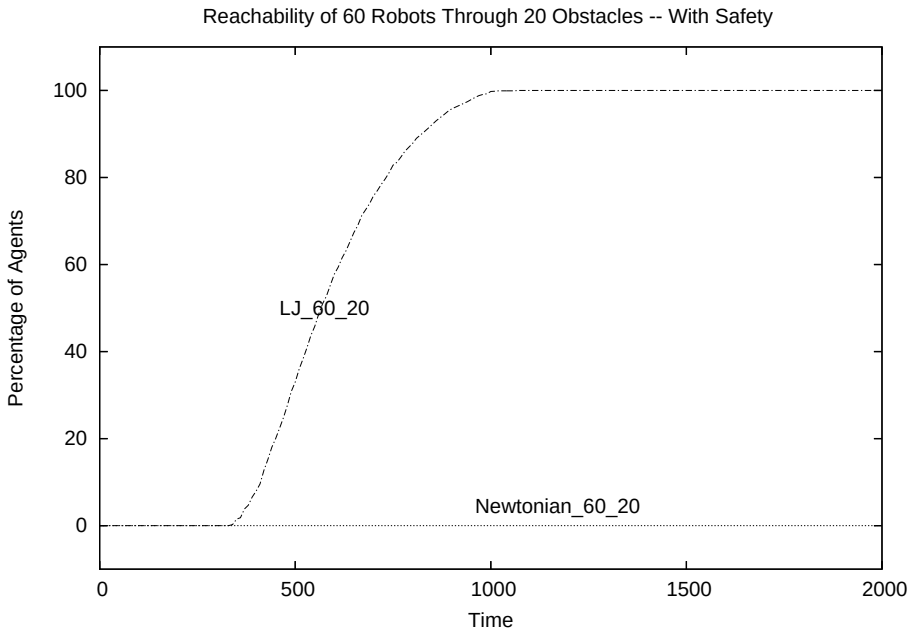

Figure 9.61: Change in reachability over 2000 time steps for 60 robots through 20 obstacles using Newtonian and LJ force laws with a safety zone

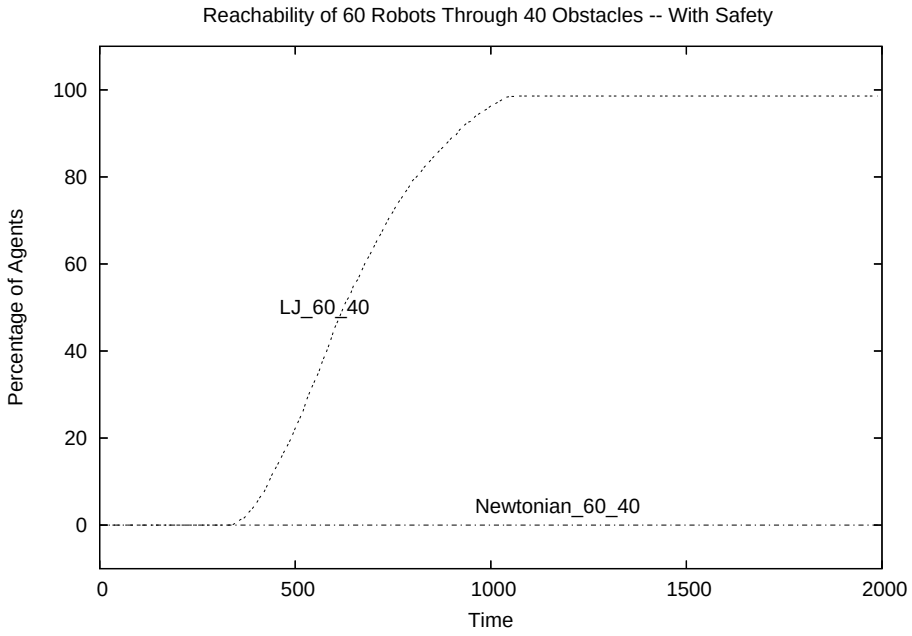

Figure 9.62: Change in reachability over 2000 time steps for 60 robots through 40 obstacles using Newtonian and LJ force laws with a safety zone

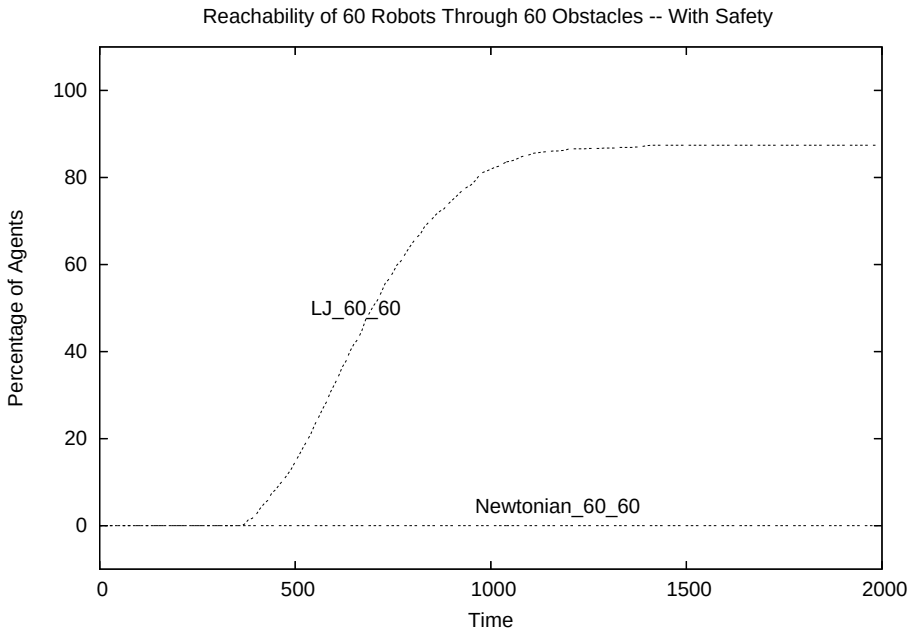

Figure 9.63: Change in reachability over 2000 time steps for 60 robots through 60 obstacles using Newtonian and LJ force laws with a safety zone

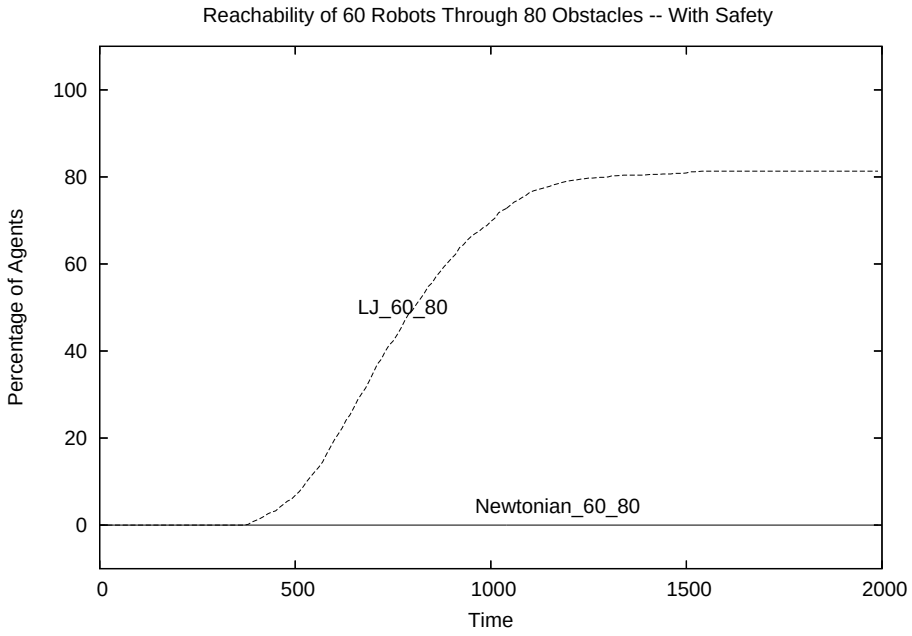

Figure 9.64: Change in reachability over 2000 time steps for 60 robots through 80 obstacles using Newtonian and LJ force laws with a safety zone

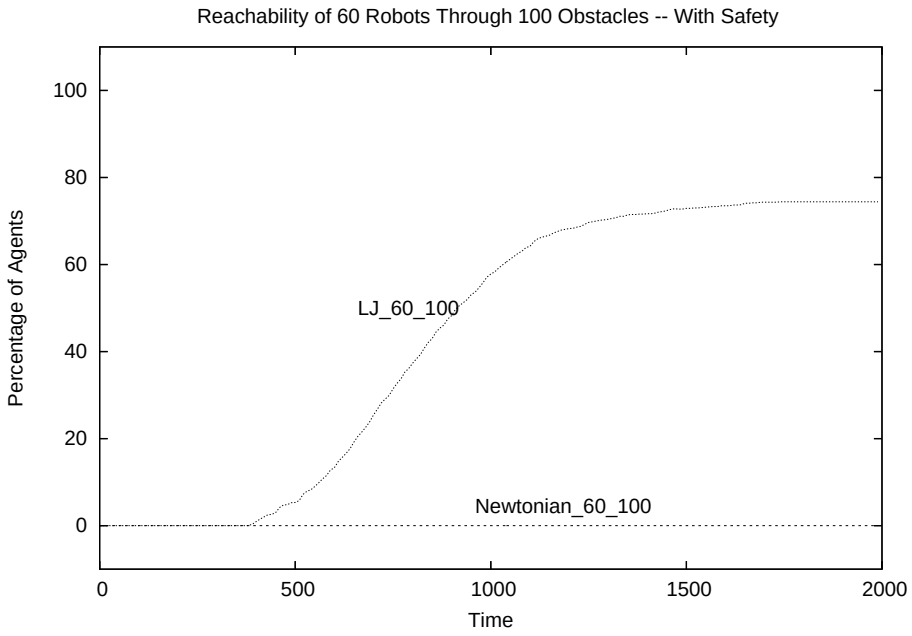

Figure 9.65: Change in reachability over 2000 time steps for 60 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone

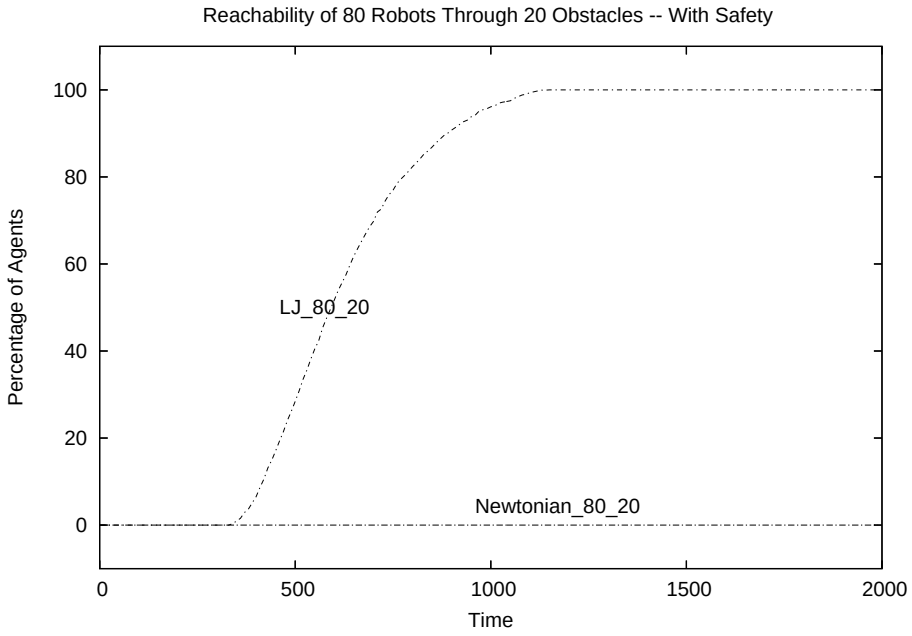

Figure 9.66: Change in reachability over 2000 time steps for 80 robots through 20 obstacles using Newtonian and LJ force laws with a safety zone

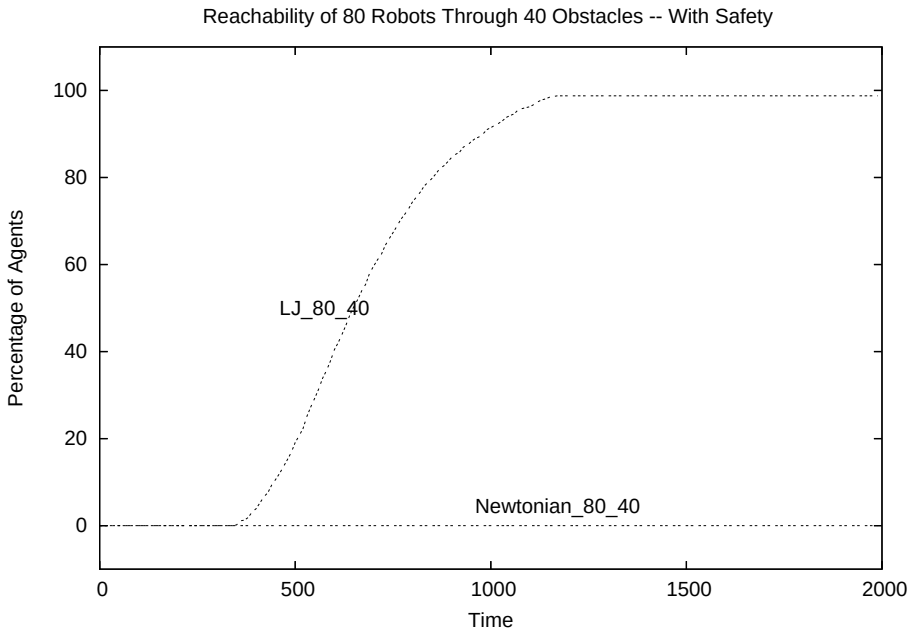

Figure 9.67: Change in reachability over 2000 time steps for 80 robots through 40 obstacles using Newtonian and LJ force laws with a safety zone

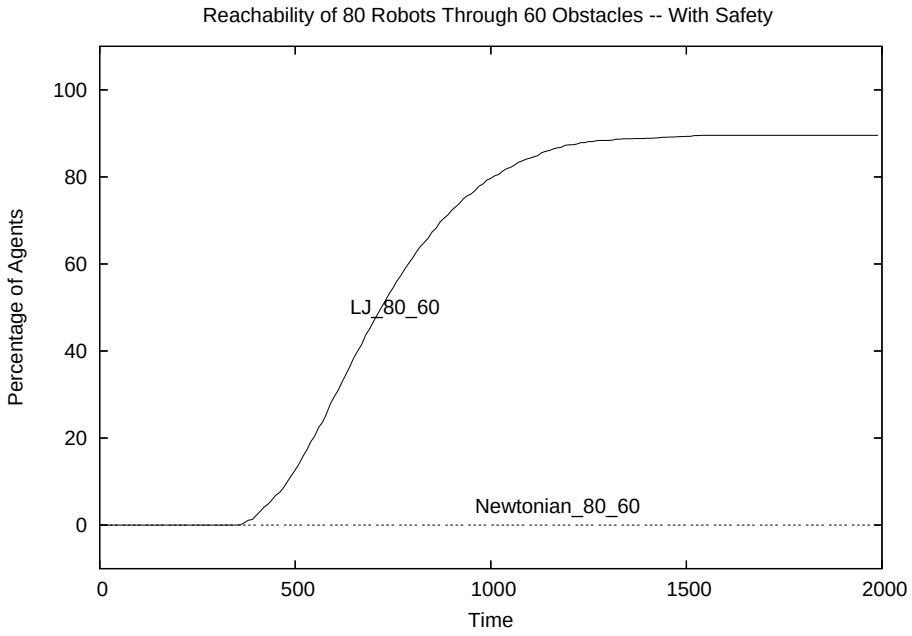

Figure 9.68: Change in reachability over 2000 time steps for 80 robots through 60 obstacles using Newtonian and LJ force laws with a safety zone

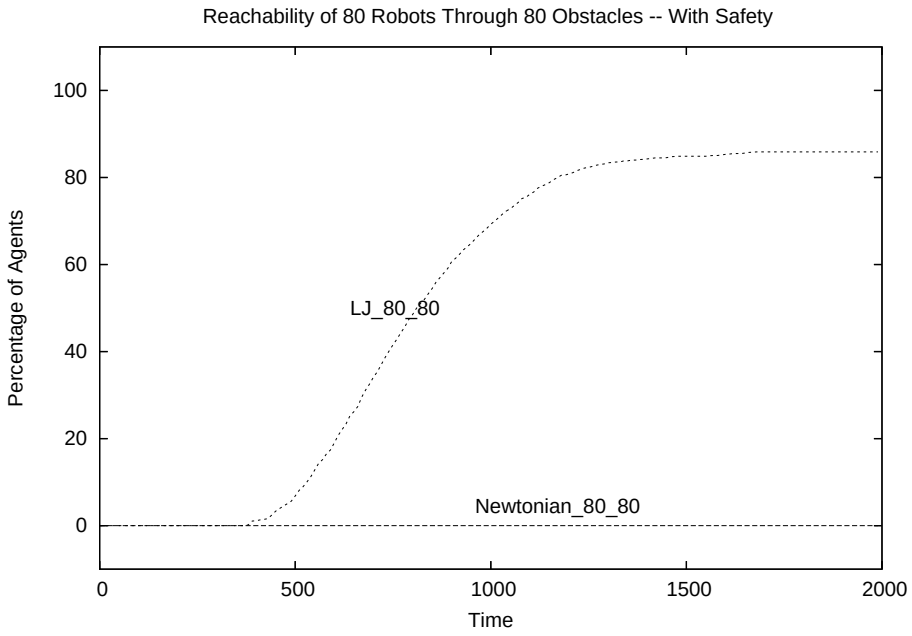

Figure 9.69: Change in reachability over 2000 time steps for 80 robots through 80 obstacles using Newtonian and LJ force laws with a safety zone

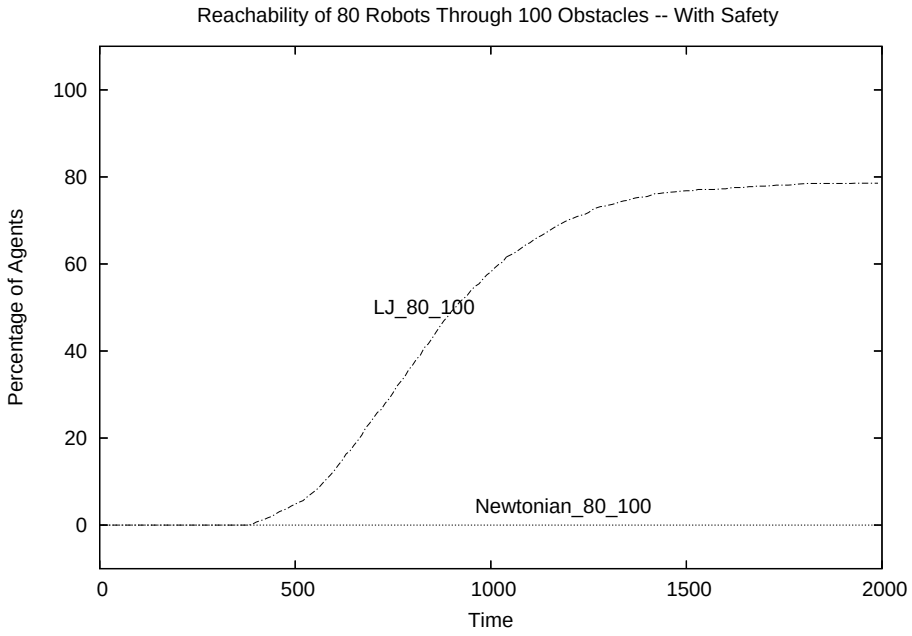

Figure 9.70: Change in reachability over 2000 time steps for 80 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone

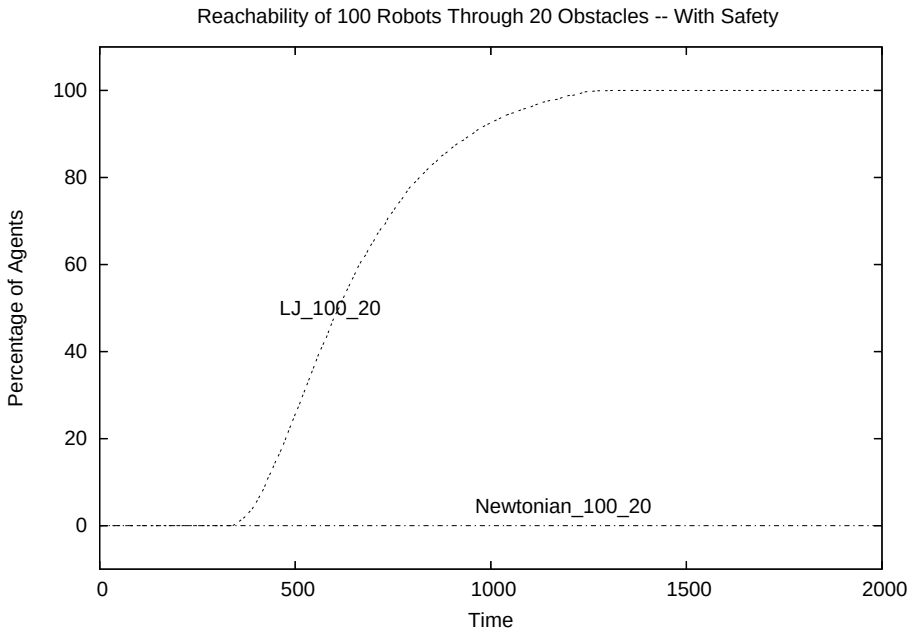

Figure 9.71: Change in reachability over 2000 time steps for 100 robots through 20 obstacles using Newtonian and LJ force laws with a safety zone

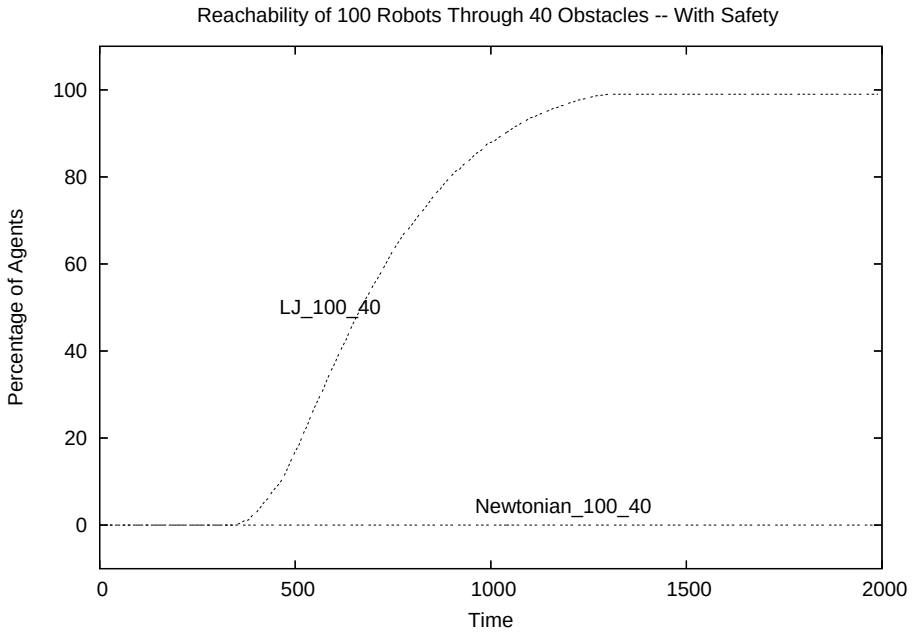

Figure 9.72: Change in reachability over 2000 time steps for 100 robots through 40 obstacles using Newtonian and LJ force laws with a safety zone

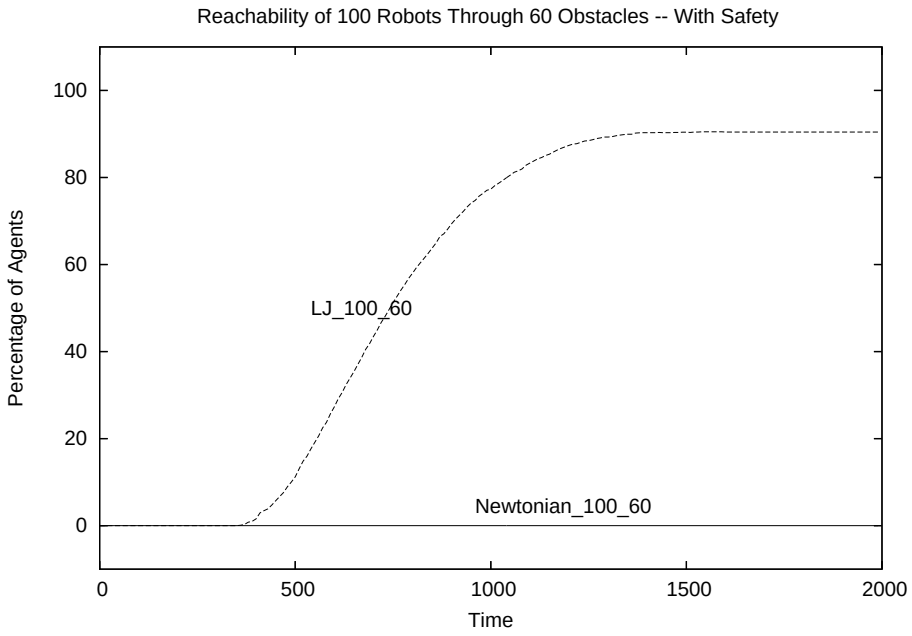

Figure 9.73: Change in reachability over 2000 time steps for 100 robots through 60 obstacles using Newtonian and LJ force laws with a safety zone

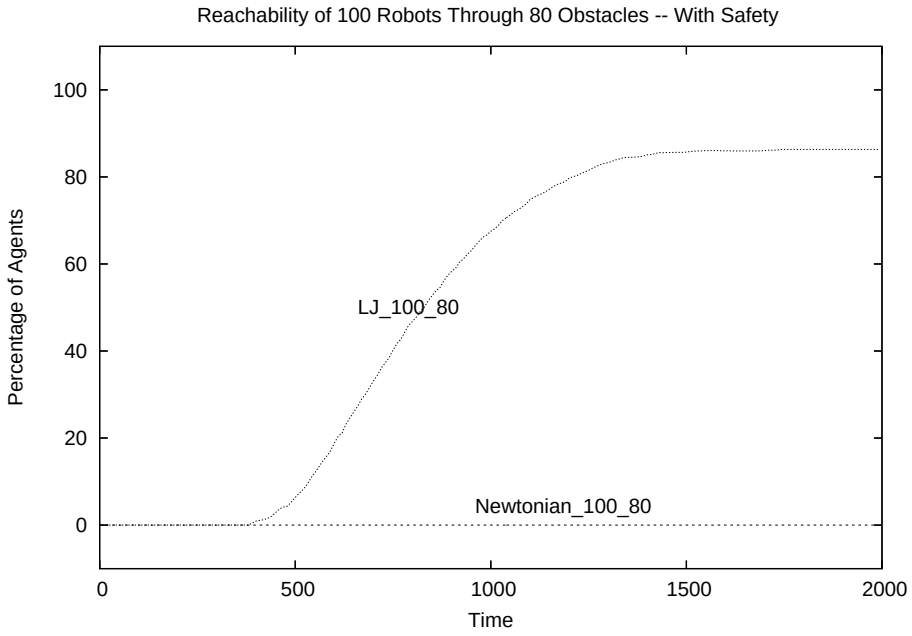

Figure 9.74: Change in reachability over 2000 time steps for 100 robots through 80 obstacles using Newtonian and LJ force laws with a safety zone

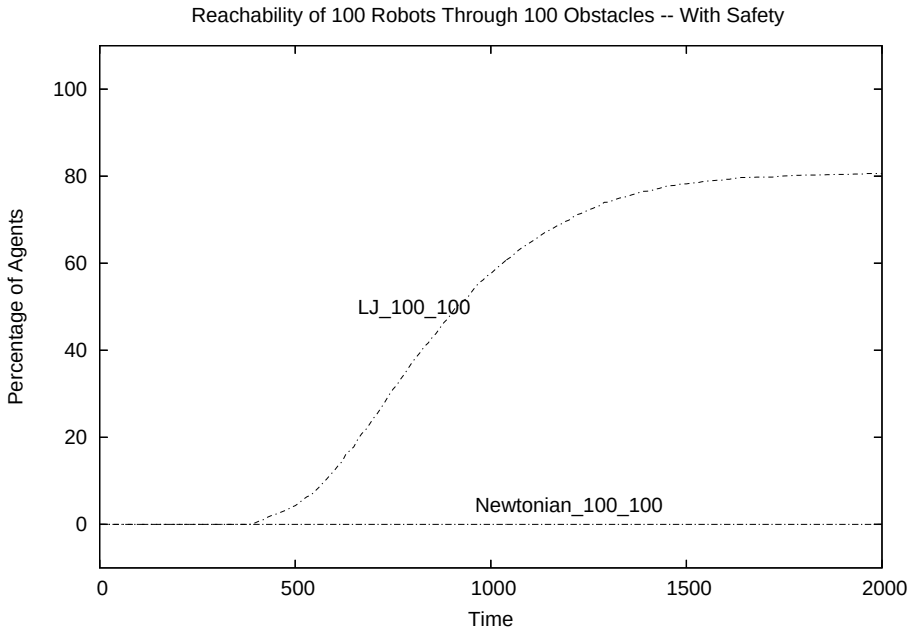

Figure 9.75: Change in reachability over 2000 time steps for 100 robots through 100 obstacles using Newtonian and LJ force laws with a safety zone

BIBLIOGRAPHY

Aoyama, H., K. Ishikawa, J. Seki, M. Okamura, S. Ishimura, and Y. Satsumi (2007). Development of mine detection robot system. *International Journal of Advanced Robotic Systems 4*(2), 229–236.

Balch, T. and R. Arkin (1998). Behavior-based formation control for multi-robot teams. *IEEE Trans. on Robotics and Autom. 14*(6), 926–939.

Balch, T. and M. Hybinette (2000). Social potentials for scalable multi-robot formations. In *IEEE International Conference on Robotics and Automation*, pp. 73–80.

Beni, G. and J. Wang (1989). Swarm intelligence. In *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, pp. 425–428.

Bonabeau, E., M. Dorigo, and G. Theraulaz (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Santa Fe Institute Studies in the Sciences of Complexity.

Borenstein, J. and Y. Koren (1989). Real-time obstacle avoidance for fast mobile robots. Volume 19, pp. 1179–1187.

Bruemmer, D., D. Dudenhoeffer, M. McKay, and M. Anderson (2002). A robotic swarm for spill finding and perimeter formation. In *Spectrum 2002*.

Cepolina, E. and M. Zoppi (2003, September). Cost-effective robots for mine detection in thick vegitation. In *Int. Conf. Climbing and Walking robots and the support technology for mobile machines CLAWAR03*, pp. 683–690.

de Croon, G., M. F. van Dartel, and E. O. Posma (2005). Evolutionary learning outperforms reinforcement learning on non-markovian tasks. In *Workshop on Memory and Learning Mechanisms in Autonomous Robots*.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test functions. *Evolutionary Computation 7*, 205–230.

Desai, J., J. Ostrowski, and V. Kumar (1998). Controlling formations of multiple mobile robots. In *IEEE International Conference on Robotics and Automation*.

Desai, J., J. Ostrowski, and V. Kumar (2001). Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation 17*(6), 905–908.

Fax, J. and R. Murray (2002). Information flow and cooperative control of vehicle formations. In *IFAC World Congress*.

Fox, D., W. Burgard, and S. Thrun (1995). The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13.

Fredslund, J. and M. Matarić (2002). A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation 18*(5).

Grefenstette, J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. Volume 3, pp. 225–245.

Grefenstette, J. (1989). A system for learning control strategies with genetic algorithms. In *Third International Conference on Genetic Algorithms*, pp. 183–190.

Haeck, N. (2002). *Minimum distance between a point and a line.* http://www.simdesign.nl/tips/tip001.html.

Hayes, A., A. Martinoli, and R. Goodman (2001). Swarm robotic odor localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Howard, A., M. Matarić, and G. Sukhatme (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Sixth Int'l Symposium on Distributed Autonomous Robotics Systems*.

Huber, D. F. and M. Herbert (1999). A new approach to 3-d terrain mapping. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 2, pp. 1121–1127.

Kazadi, S. (2005). On the development of a swarm engineering methodology. In *IEEE International Conference on Systems, Man and Cybernetics*, Volume 2, pp. 1423–1428. IEEE Press.

Kerr, W. and D. Spears (2005). Robotic simulation of gases for a surveillance task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*.

Kerr, W., D. Spears, W. Spears, and D. Thayer (2004). Two formal fluids models for multiagent sweeping and obstacle avoidance. *Lecture Notes in Artificial Intelligence 3228*.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int'l Journal of Robotics Research 5*(1), 90–98.

Kim, J. and P. Khosla (1991). Real-time obstacle avoidance using harmonic potential functions. In *IEEE Int'l Conf. on Robotics and Autom.*, pp. 790–796.

Koren, Y. and J. Borenstein (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Int'l Conf. on Robotics and Autom.*, pp. 1398–1404.

Li, L., A. Martinoli, and Y. Abu-Mostafa (2003). Diversity and specialization in collborative swarm systems. In *Second International Workshop on the Mathematics and Algorithms of Social Insects*, Volume 39, pp. 91–98. Elsevier.

Martinson, E. and D. Payton (2005). Lattice formation in mobile autonomous sensor arrays. In *Lecture Notes in Computer Science*, Volume 3342, pp. 98–111. Springer-Verlag.

Menczer, F., M. Degeratu, and W. Street (2000). Efficient and scalable pareto optimization by evolutionary local selection algorithms. *Evolutionary Computation 8*(2), 223–247.

Murphy, R. R. and J. L. Burke (2005). Up from the rubble: Lessons learned about hri from search and rescue. In *49th Annual Meetings of the Human Factors and Ergonomics Society*.

Nilsson, N. (1984). Shakey the robot. *Technical Note 323. AI Center, SRI International 323*.

O'Hara, K. J., V. L. Bigio, E. R. Dodson, A. Irani, D. B. Walker, and T. R. Balch (2005). Physical path planning using the gnats. In *IEEE International Conference on Robotics and Automation*.

Reif, J. and H. Wang (1998). Social potential fields: A distributed behavioral control for autonomous robots. In *Workshop on the Algorithmic Foundations of Robotics*.

Sarma, J. and K. de Jong (1998). Selection pressure and performance in spatially distributedevolutionary algorithms. In *IEEE World Congress on Computational Intelligence*, pp. 553–557.

Sbalzarini, I., S. Mller, and P. Koumoutsakos (2000). Multiobjective optimization using evolutionary algorithms. In *Center for Turbulence Research Proceedings of the Summer Program*, pp. 63–74.

Schoenwald, D., J. Feddema, and F. Oppel (2001). Decentralized control of a collective of autonomous robotic vehicles. In *American Control Conference*, pp. 2087–2092.

Schultz, A. C. (1991). Using a genetic algorithm to learn strategies for collision avoidance and local navigation. In *International Symposium on Unmanned Untethered Submersible Technology*, pp. 213–225.

Simmons, R. (1996). The curvaturevelocity method for local obstacle avoidance. In *In Proceedings of the International Conference on Robotics and Automation*, pp. 3375–3382.

Spears, W. (1994). Simple subpopulation schemes. In *Proceedings of the Evolutionary Programming Conference*, pp. 296–307.

Spears, W. and D. Gordon (1999). Using artificial physics to control agents. In *IEEE International Conference on Information, Intelligence, and Systems*, pp. 281–288.

Spears, W., D. Gordon-Spears, J. Hamann, and R. Heil (2004, August). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots 17*, 137–162.

Spears, W., J. Hamann, P. Maxim, T. Kunkel, R. Heil, D. Zarzhitsky, D. Spears, and C. Karlsson (2006). Where are you? In *Second Workshop on Swarm Robotics*. Springer-Verlag.

Spears, W., K. D. Jong, T. Bačk, D. Fogel, and H. de Garis (1993). An overview of evolutionary computation. In *European Conference on Machine Learning*.

Spears, W., D. Spears, and R. Heil (2004). A formal analysis of potential energy in a multiagent system. In *Proceedings of FAABS III*.

Spears, W., D. Spears, R. Heil, W. Kerr, and S. Hettiarachchi (2004). An overview of physicomimetics. *Lecture Notes in Computer Science, State-of-the-Art Series 3342*.

Spears, W., D. Zarzhitsky, S. Hettiarachchi, and W. Kerr (2005). Strategies for multi-agent surveillance. In *IEEE Networking, Sensing and Control*, pp. 929–934. IEEE Press.

Vail, D. and M. Veloso (2003). Multi-robot dynamic role assignment and coordination through shared potential fields. In *Multi-Robot Systems*. Kluwer.

Varakantham, P., R. Maheswaran, and M. Tambe (2004). Agent modelling in partially observable domains. In *Workshop on Modeling Other Agents from Observations, AAMAS04*.

Watson, R., S. Ficici, and J. Pollack (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots. In *Robotics and Autonomous Systems*, Volume 39, pp. 1–18. Elsevier.

Wiegand, R. P., A. M. Potter, D. A. Sofge, and W. M. Spears (2006). A generalized graph-based method for engineering swarm solutions to multiagent problems. In *Parallel Problem Solving from Nature*, pp. 741–750.

Wolf, A., H. B. Brown, R. Casciola, A. Costa, M. Schwerin, E. Shamas, and H. Choset (2003, October). A mobile hyper redundant mechanism for search and rescue tasks. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 3, pp. 2889 – 2895.

Wu, A., A. Schultz, and A. Agah (1999). Evolving control for distributed micro air vehicles. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*.

Yao, X., Y. Liu, and P. Darwen (1996). How to make best use of evolutionary learning. In *Complex Systems: From Local Interactions to Global Phenomena*, pp. 229–242.

Zarzhitsky, D., D. Spears, and W. Spears (2005). Swarms for chemical plume tracing. In *IEEE Swarm Intelligence Symposium (SIS'05)*. IEEE Press.

Zarzhitsky, D., D. Spears, D. Thayer, and W. Spears (2004). Agent-based chemical plume tracing using fluid dynamics. In *Lecture Notes in Artificial Intelligence*, Volume 3228. Springer-Verlag.