Interaction and Intelligent Behavior

by

Maja J Matarić

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Author . . . .
Department of Electrical Engineering and Computer Science
May 12, 1994

Certified by . . . .
Rodney A. Brooks
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . .
Frederic R. Morgenthaler
Chairman, Committee on Graduate Students

Interaction and Intelligent Behavior

by

Maja J Matarić

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis addresses situated, embodied agents interacting in complex domains. It focuses on two problems: 1) synthesis and analysis of intelligent group behavior, and 2) learning in complex group environments.

Basic behaviors, control laws that cluster constraints to achieve particular goals and have the appropriate compositional properties, are proposed as effective primitives for control and learning. The thesis describes the process of selecting such basic behaviors, formally specifying them, algorithmically implementing them, and empirically evaluating them. All of the proposed ideas are validated with a group of up to 20 mobile robots using a basic behavior set consisting of: *safe–wandering*, *following*, *aggregation*, *dispersion*, and *homing*. The set of basic behaviors acts as a substrate for achieving more complex high–level goals and tasks. Two behavior combination operators are introduced, and verified by combining subsets of the above basic behavior set to implement collective *flocking*, *foraging*, and *docking*.

A methodology is introduced for automatically constructing higher–level behaviors by learning to select among the basic behavior set. A novel formulation of reinforcement learning is proposed that makes behavior selection learnable in noisy, uncertain multi–agent environments with stochastic dynamics. It consists of using **conditions** and **behaviors** for more robust control and minimized state–spaces, and a reinforcement shaping methodology that enables principled embedding of domain knowledge with two types of shaping functions: **heterogeneous reward functions** and **progress estimators**. The methodology is validated on a collection of robots learning to forage. The generality of the approach makes it compatible with the existing reinforcement learning algorithms, allowing it to accelerate learning in a variety of domains and applications.

The presented methodologies and results are aimed at extending our understanding of synthesis, analysis, and learning of group behavior.

Thesis Supervisor: Rodney A. Brooks
Title: Professor of Computer Science and Engineering

Acknowledgments

Since this is neither a Nobel Prize nor an Oscar acceptance, the acknowledgements need not be short. There are many people to thank for contributing to seven great years I have had at the MIT Artificial Intelligence Lab.

I thank Rod Brooks for being a great advisor, an excellent motivator, a rebel, a role model, and most importantly, never a conservative. I am grateful for his advice and patience, and his continuing support. I am also grateful to Gerry Sussman, Lynn Stein, and Patrick Winston for good thesis–related, job–related, and life–relate advice during my last year at MIT.

Unlimited thanks go to Mike Bolotski and Jose Robles, who have been great friends from the beginning until the end, participated in kayaking classes, heated discussions, and LaTeX hacking in the last hours of thesis preparation. Together with Brian Eberman and Sundar Narasimhan, they have kept me honest through the years by asking hard questions and not accepting mediocre answers. Special thanks to Mike for proof–reading numerous papers, encouraging me to learn Russian, introducing me to the haymarket, and being a superb friend to have in Vancouver. Special thanks to Jose for being a good and patient friend, dance partner, and source of good advice all the way back to the days of Toto and the Masters' thesis to the formalization of the basic behaviors in this document. Infinite thanks to Jose for the willingness to read the thesis again and provide numerous invaluable comments for the Tech Report.

Numerous thanks to Anita Flynn for being an inspiring force, a fantastic cheerleader for running and research, and always great to talk to about all aspects of life. Warmest thanks to Mike Erdmann for being a wonderfully supportive officemate during the first two years at MIT, for initiating me into the culture, for talking and walking and running and sailing and for unenumerable good things he has done. Endless gratitude to Nancy Pollard for being a patient, fun, and supportive officemate for four great years. Thanks for sharing everything, including running, squash, research dilemmas, and plant watering.

Many Slavic thanks to Matt Marjanović for being crazy in just the right way, for managing as my officemate during the job interview and thesis season, for making me put the ć back into my last name, and of course, for the Pig. Thanks to Ian Horswill for being a great companion for flights to far-away conferences, for being a patient source of good advice, and for introducing me to the weird world of comics. Sibling thanks to Paul Viola for being a "bro" in the early years at the Lab, for giving me outstanding advice about theses committees, research directions, people to talk to about work, and most of all for not hesitating in saying exactly what he is thinking.

Thanks to Cindy Ferrell for having an endless supply of good cheer, great spirit, excellent taste (especially in tech report covers), and enthusiasm for both research, sports, and exotic vacations. Thanks to Pattie Maes for being a friend and a part–time advisor on all matters imaginable, for having impeccable taste (again in all matters imaginable), and for being a superb role model. Thanks to Lynne Parker for understanding the robots' quirks better, and for having endless patience with both them and me. She was a great person with whom to share research views as well as hardware frustrations.

Many thanks to three hard–working UROPS: Matt Marjanović, Stanley Wang, and Owen Wessling, for making the unruly Nerd Herd behave. Thanks to Phillip Alvelda for rock climbing and rappeling lessons, bad movie advice but good home movies, and endless enthusiasm. Thanks to David Beymer, Gideon Stein, Sandy Wells, and Matt Williamson for being great running and talking partners. Added thanks to Matt for the most hilarious squash games and the Scottish accent. Thanks to Carl de Marken for writing memorable Girl Scout Benefit announcements and for being an expert and a patient teacher for every outdoor activity known to mankind. Many thanks to Adonis Stassinopoulos for unsurpassed Greek desserts, great taste in movies, and for always being excellent company.

Many thanks to Bruce Blumberg, David Brock, Barb Moore Bryant, and Amy Bruckman for great discussions, to Holly Yanco and Joanna Bryson for their enthusiasm and organizational skills, to Mike Caine for bizarre humor through the years and for the broken talking clock, to the great people in the Cog Group, to Trevor Darrell for always having something interesting and new to say on a variety of topics, and for always keeping in touch, to Lisa Dron for good dinner parties, to Charles Isbell for great enthusiasm and unique quotes, to Tom Knight for being at the Lab at weird hours, knowing about and always being willing to discuss any and all topics, and for loaning me "green slime" for building a ceiling–swinging robot, to Tina Kapur and Lilly Lee for not minding me adopting the fridge in their office, to Marina Meila for being a great travel–mate and wonderful to talk to about machine learning, to Henry Minsky for being an irreplaceable cow supplier, to Annika Pfluger for all administrative help through the years and great talent for cello, to Robert Ringrose for saintly patience in helping with the Creature Library and the video equipment, to Ruth Schonfeld for being a great friend to talk to even if we loose touch for a few months at a time, to Laurel Simmons for keeping the Lab from collapsing, to Patrick Sobalvarro for great literary skills and movie advice, and to Jeanne Speckman for organizing the dance group, and to everybody I have unintentionally left out.

Deepest thanks to Liba and Bora Mikić, who served as my Boston family, and were always ready to overfeed me and listen to my stories. Great thanks to Milan

Radovanov for being the smart uncle I could talk about and talk with, and for always playing devil's advocate to keep me on my toes.

Warmest and most enduring thanks to Rich Roberts, for over a decade of love, support, and inexplicable and irresistible silliness. He made every minute more fun, more memorable, and more worthwhile.

Infinite and incomparable gratitude to Mira Matarić, the most wonderful mother I could have. She taught all lessons by example, worked harder for and at everything in life than anybody else I have known, and had superhuman energy and perseverance that inspired everything I have accomplished. This thesis is dedicated to her.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Overview of the Thesis

One of the main goals of Artificial Intelligence (AI) is to gain insight into natural intelligence through a synthetic approach, by generating and analyzing artificial intelligent behavior. In order to glean an understanding of a phenomenon as complex as natural intelligence, we need to study complex behavior in complex environments.

Traditionally, AI has concerned itself with complex agents in relatively simple environments, simple in the sense that they could be precisely modeled and involved little or no noise and uncertainty. In contrast to traditional systems, reactive and behavior–based systems have placed agents with low levels of cognitive complexity into complex, noisy and uncertain environments. This thesis describes work that attempts to simultaneously scale up along both dimensions. The environmental complexity is scaled up by introducing other agents, and cognitive complexity is scaled up by introducing learning capabilities into each of the agents (Figure 1-1).

This thesis addresses two problems:

1. synthesis and analysis of intelligent group behavior

2. learning in complex group environments

Our ideas are based on the notion of *basic behaviors*, a means for combining constraints from the agent, such as its mechanical and sensory characteristics, and the constraints for the environment, such as the types of interactions and sensory information the agent can obtain, in order to construct an appropriate abstraction for structuring primitives for control.

We will present a methodology that uses basic behaviors to generate various robust group behaviors, including following, homing, and flocking (Figure 1-2). We will also introduce a formulation of reinforcement learning based on behaviors as the unit of representation that allows a group of agents to learn complex tasks such as foraging

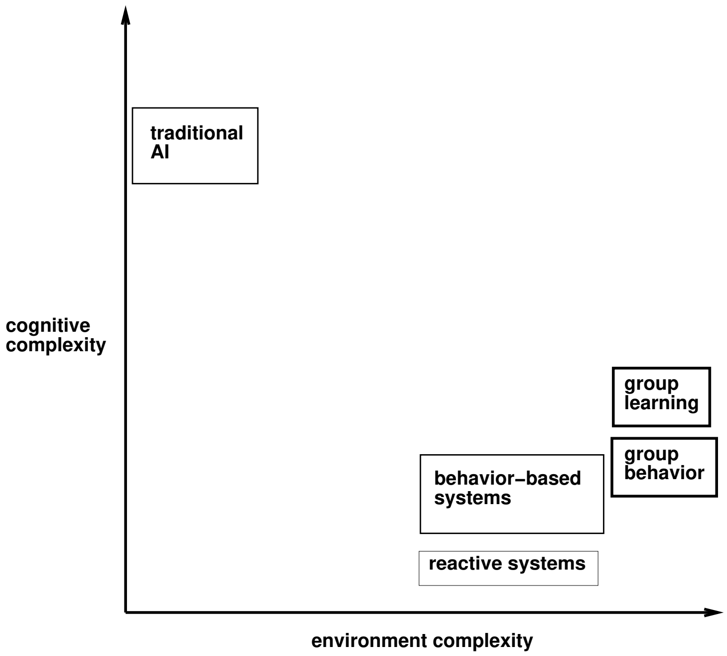
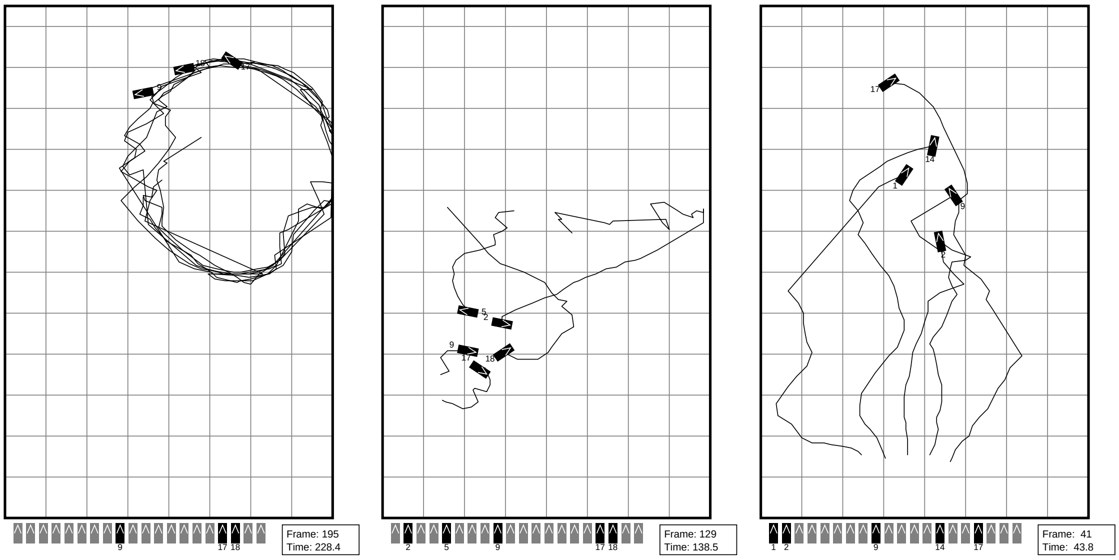


Figure 1-1: Traditional AI has addressed complex agents in simple environments while reactive and behavior–based approaches have dealt with simple agents in noisy and uncertain worlds. This work attempts to scale up along both dimensions simultaneously, by addressing synthesis and learning of complex group behavior.



Following Homing Flocking

Figure 1-2: This figure shows examples of real robot data for three different group behaviors: following, homing, and flocking. The robots, physically 12 inches long, are scaled down and plotted as black rectangles, with white arrows indicating their heading. The dark robots in the row of rectangles at the bottom shows the robots that were used in the experiment. Boxes on the lower right indicate frame numbers and the elapsed time in seconds for each of the runs.

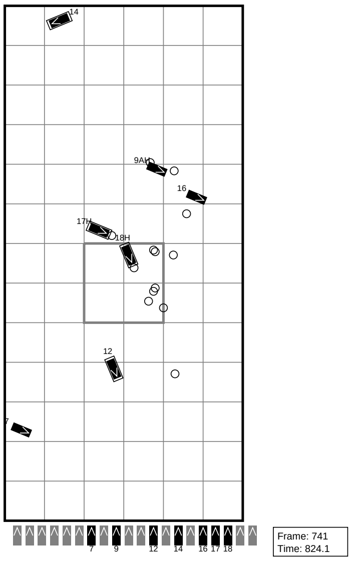


Figure 1-3: An example of the foraging behavior of 7 robots, shown after 13.7 minutes of running. About eight pucks have been delivered to the home region, marked with a grey box. The two robots near home are following each other on the way to the drop-off. Other robots are wandering in search of additional pucks.

(Figure 1-3). Finally, we will validate the proposed approaches with experiments on homogeneous groups of mobile robots.

This chapter gives a brief summary of the novel approaches, of the experimental data, and of the implications of the thesis. The organization of the thesis is outlined at the end of the chapter.

1.1 Synthesis and Analysis of Group Behavior

This thesis is based on the belief that intelligent collective behavior in a decentralized system results from *local interactions* based on simple rules. **Basic behaviors** are proposed as a methodology for structuring those rules through a principled process of synthesis and evaluation. A *behavior* is a control law that clusters a set of constraints in order to achieve and maintain a goal. For example, *safe–wandering* is a behavior that maintains the goal of avoiding collisions while the agent is moving.

We postulate that, for each domain, a set of behaviors can be found that are **basic** in that they are required for generating other behaviors, as well as being a minimal set the agent needs to reach its goal repertoire. The process of choosing the set of basic behaviors for a domain is dually constrained. From the bottom–up, the process is constrained by the dynamics of the agent and the environment. From the top–down, the process is constrained by the agent's goals as specified by the task. The combination of the two types of constraints helps to prune the agent's behavior space.

We will use the example of group interactions between situated, embodied agents to illustrate the process of selecting a basic behavior set. The agents are mobile robots, embodied and endowed with specific mechanical, sensory, and effector constraints. We define the high–level goals of the system as consisting of collectively moving objects (pucks) in the environment in an efficient fashion. In this work, efficiency is defined in terms of minimizing energy by minimizing the amount of time required to complete a task or the number of moves required for each of the agents.

An effective set of basic behaviors in the spatial domain should enable the agents to employ a variety of flexible strategies for puck manipulation, collection, and distribution. The effectiveness of such strategies depends on maximizing synergy between agents: achieving the necessary goals while minimizing inter–agent interference.

We propose the following set of basic behaviors:

- *safe–wandering* – minimizes collisions between agents and environment
- *following* – minimizes interference by structuring movement of any two agents
- *aggregation* – gathers the agents
- *dispersion* – dissipates the agents
- *homing* – enables the agent to proceed to a particular location

According to our definition, the above behavior set is *minimal* or *basic* in that its members are not further reducible to each other. Additionally, we will show that they are sufficient for achieving the set of pre–specified goals. The described basic behaviors are defined with respect to the group. Other utility behaviors, such as *grasping* and *dropping*, can also be a part of an agent's repertoire.

The basic behavior set is evaluated by giving a formal specification of each of the behaviors, and comparing the collection of those specifications to a formal specification of the set of global tasks required for the group.

Once a basic behavior set is established, it can be implemented with a variety of algorithms. The first step in the verification of basic behavior algorithms is a comparison between the formal behavior specification and the formal correctness of the algorithm. We will argue that it is difficult to prove properties of the exact behavior of individual agents within a group, but it is possible to evaluate and predict the behavior of the ensemble as a whole. Using the notion of ensemble behavior, we

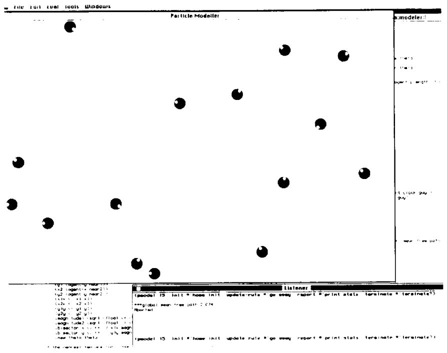


Figure 1-4: The simulator environment called the Interaction Monitor was used to validate the methodologies for synthesizing and analyzing group behavior described in the thesis. The agents are shown as black circles, with white markers indicating their heading. The large rectangle represents the agents' workspace.

will propose group behavior algorithms that utilize a centroid operator that averages the inputs from multiple agents. This operator has statistical properties that allow analyzing and making predictions about the behavior of the group.

This thesis provides detailed specifications and algorithms for each of the basic behaviors. Instead of analytical proofs, it provides empirical evaluations of the performance of each of the algorithms, based on the following criteria:

- repeatability: how consistent is the behavior over different trials?
- stability: does the behavior oscillate under any conditions?
- robustness: how robust is the behavior in the presence of sensor and effector error and noise?
- scalability: how is the behavior effected by increased and decreased group sizes?

The above criteria were applied to the data obtained by performing at least 50 trials of each basic behavior. The experiments were performed on two different multi–agent environments, in order to minimize domain biases. The first environment was a multi–agent simulator (the Interaction Monitor) featuring up to 50 agents with local sensing and distributed, local control (Figure1-4).

The second environment was a collection of 20 physical mobile robots equipped with local sensors and local control (Figure 1-5). Each of the robots is equipped with a suite of infra–red sensors for collision avoidance, puck detection, and stacking, and with micro switches and bump sensors for contact detection. In addition to the

Figure 1-5: Some of the 20 mobile robots used to validate the group behavior methodologies described in the thesis. These robots demonstrated group safe–wandering, following, aggregation, dispersion, flocking, and foraging.

local sensors, the robots are equipped with radios and sonars for triangulating their position relative to two stationary beacons, and for broadcasting that position within a limited radius. The radios are used to detect other robots and gather data for local centroid computations.

The basic behaviors, each consisting of one rule or a small set of simple rules, generated robust group behaviors that met the prespecified evaluation criteria. A small subset of the data is shown here, using the Real Time Viewer[1], a software package for displaying and replaying each of the robots runs, plotting their positions over time, and displaying each frame and the elapsed time for each experiment. The figures show following (Figure 1-6), dispersion (Figure 1-7), and homing (Figure 1-8). More of the data, the algorithms, the specifications, and a detailed evaluation can be found in Chapter 4.

Basic behaviors are intended as building blocks for achieving higher–level goals. The behaviors are embedded in an architecture that allows two types of combination: direct, by summation, and temporal, by switching (see figure 1-9). Both types of combination operators were tested empirically. A simple and robust *flocking* behavior was generated by summing the outputs of *safe–wandering*, *aggregation*, and *homing* (Figure 1-10). A more complex *foraging* behavior that involves finding and collecting pucks, was implemented by switching between *safe–wandering*, *dispersion*, *following*,

[1]Written by Matthew Marjanović.

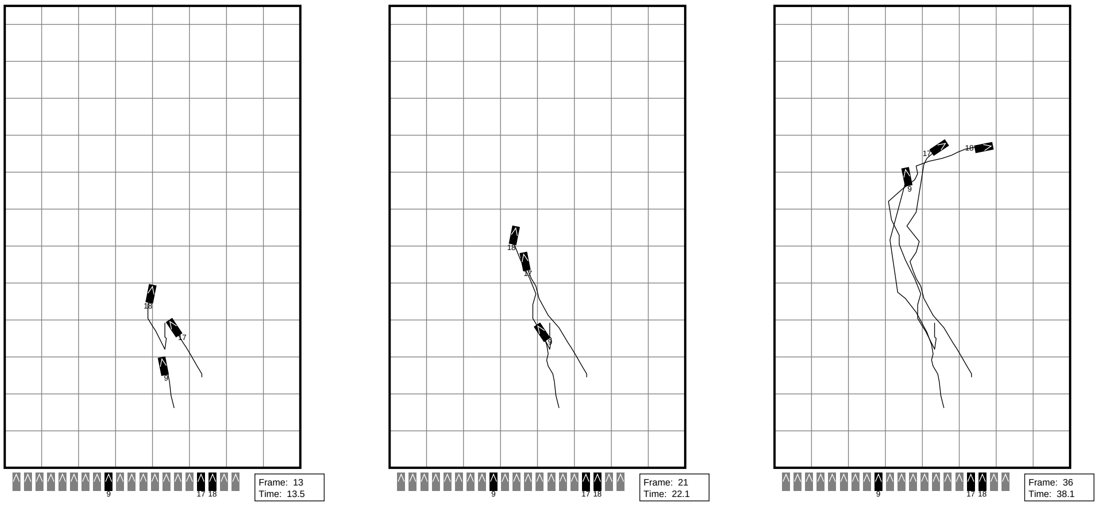


Figure 1-6: Continuous following behavior of 3 robots. The entire time history of the robots' positions is plotted.

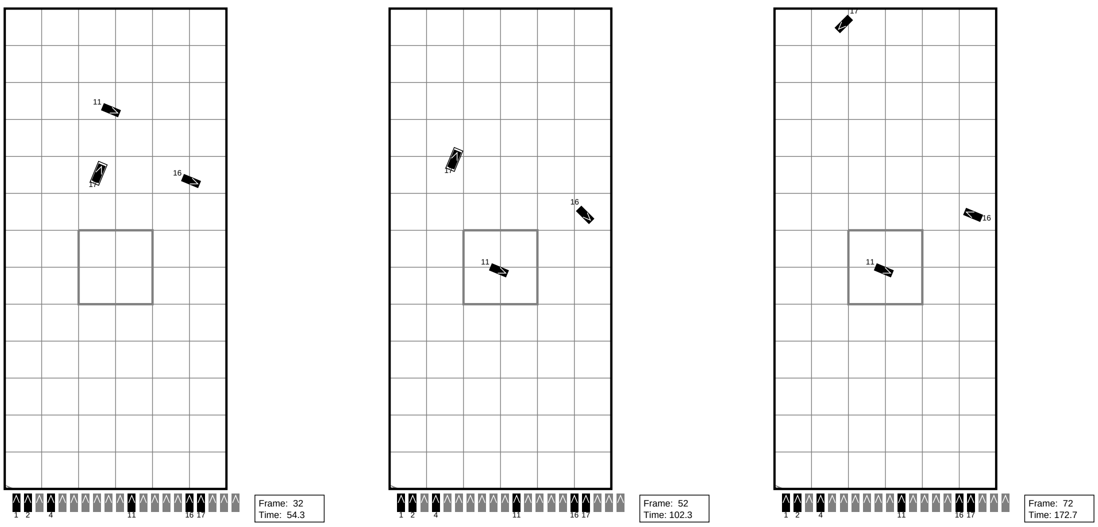


Figure 1-7: Dispersion behaviors of 3 robots.

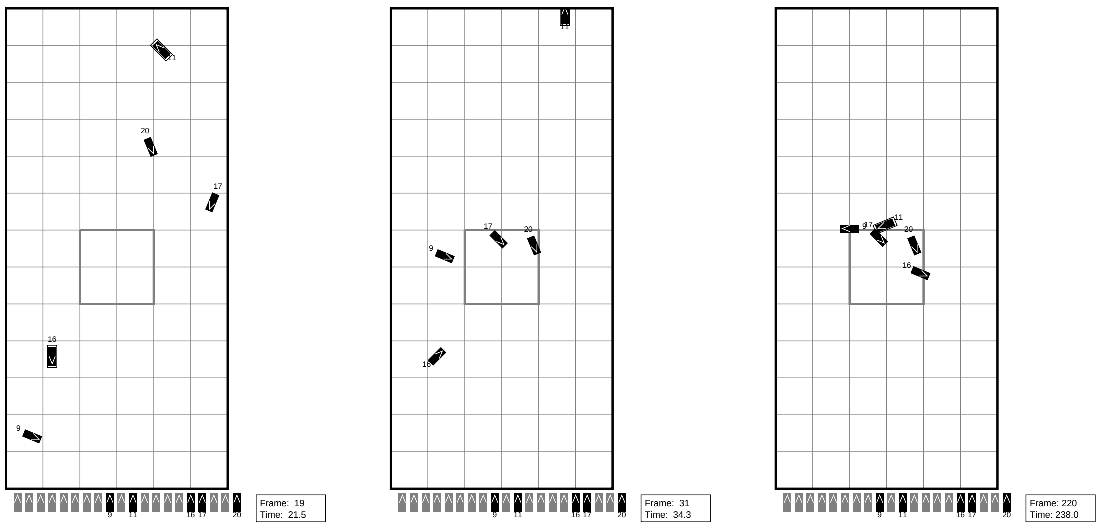


Figure 1-8: Homing behaviors of 5 robots. Four of the five robots reach home quickly and the fifth joints them about 60 second later.

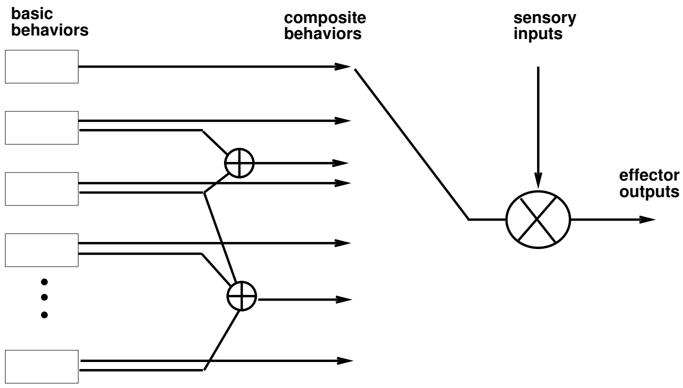


Figure 1-9: The control architecture for generating group behaviors consists of direct and temporal combinations (i.e. sums and switches) of subsets from a fixed basic behavior set. Direct combinations are marked with $\oplus$, temporal combinations with $\otimes$.

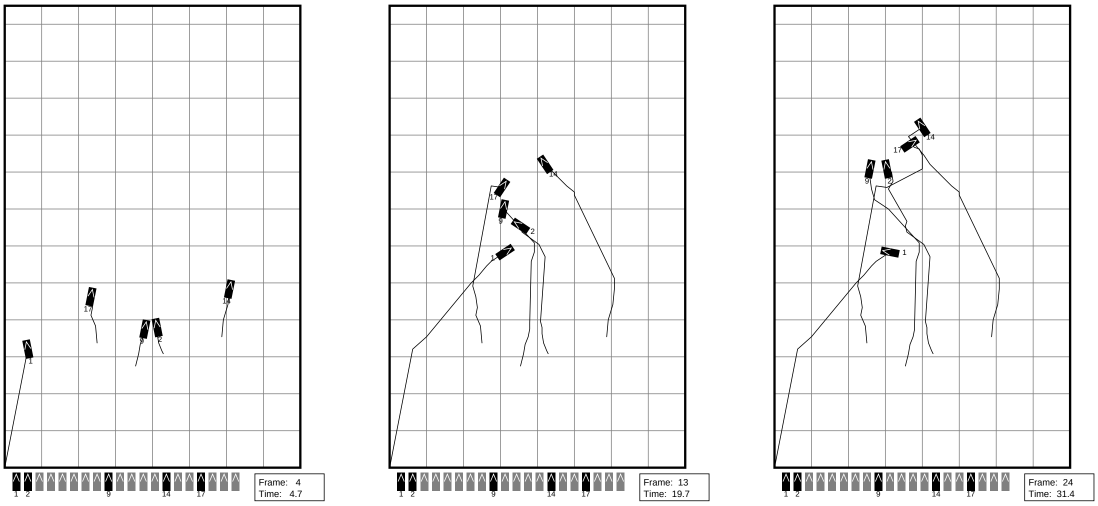


Figure 1-10: Flocking behavior of 5 robots. The robots are started out in a nearly linear dispersed state. They quickly establish a flock and maintain it as the positions of the individual robots within the flock fluctuate over time.

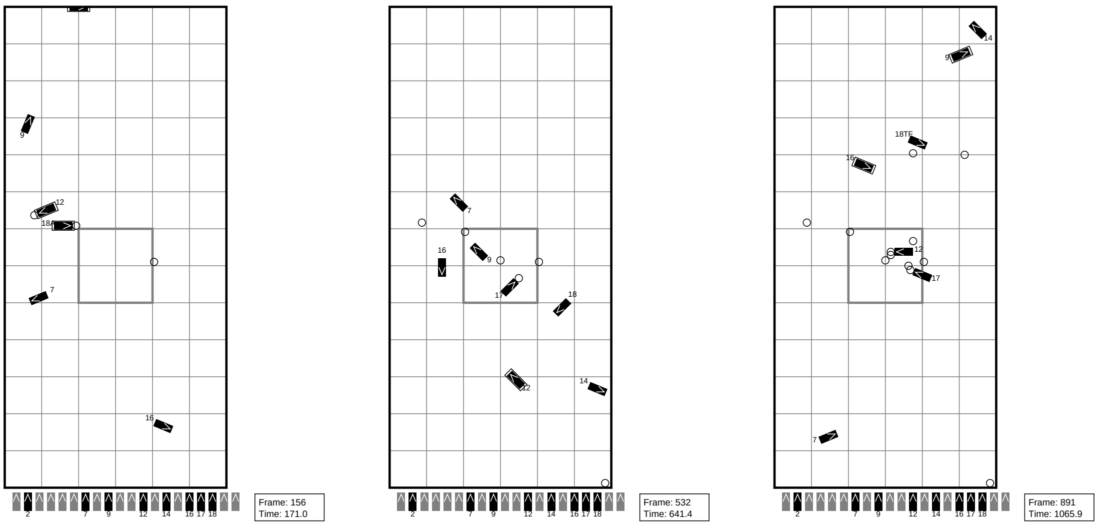


Figure 1-11: An example of the foraging behavior of 6 robots. About eight pucks have been delivered to the home region, marked with a grey box. Two of the robots are dropping off pucks while the others are wandering in search of additional pucks to pick up and deliver home.

and *homing* (Figure 1-11).

In addition to empirical testing of the behaviors and their combinations, the proposed methodology for generating decentralized group behavior was compared to a centralized, "total knowledge" approach. The experimental results showed that the simple, fully distributed strategies, applied to *dispersion* and *aggregation* tasks, converged only a constant factor slower than the centralized approach.

1.2 Learning in Complex Group Environments

The first part of the thesis introduces basic behaviors as a methodology for structuring simple rules into flexible and effective repertoires of group behavior. It also presents combination operators that allow for constructing and achieving higher–level goals. The second part of the thesis, starting with Chapter 6, describes a methodology for automatically combining basic behaviors into higher–level ones, though unsupervised **reinforcement learning** based on the agents' interactions with the environment.

In reinforcement learning (RL) approaches the agent learns from external scalar reward and punishment. RL has been successfully applied to a variety of domains that have largely been modeled as Markovian, where the agent–environment interaction can be described as a Markov Decision Process (MDP). However, the MDP assumption does not directly apply to the noisy and uncertain multi–agent environments addressed in this work. Nonetheless, since external and internal feedback are the most natural sources of information for learning in situated agents, methods for applying RL to such complex domains are needed.

The traditional formulation of RL problems in terms of states, actions, and reinforcement required a reformulation in order to be applied to our domain. The notion of state as a monolithic descriptor of the agent and the environment did not scale up to the multi–agent domain used here, given the continuous and discrete aspects describing the agent (e.g., velocity, IR sensors, radio data), and the existence of many other agents in the environment. Furthermore, the most commonly used notion of actions was inappropriate since atomic actions were too low level and had effect too unpredictable and noisy to be useful to a learning algorithm. Finally, delayed reinforcement and reward discounting were insufficient for learning in our domain.

To make learning possible we propose a reformulation that elevates the level of system description from states and actions to conditions and behaviors. *Behaviors* are control laws that achieve goals but hide low–level control details. Using the notion of *basic behaviors*, a small basis set can be defined as used as a substrate for learning. When actions are replaced with behaviors, states can be replaced with *conditions*, the necessary and sufficient subsets of state required for triggering the behavior set.

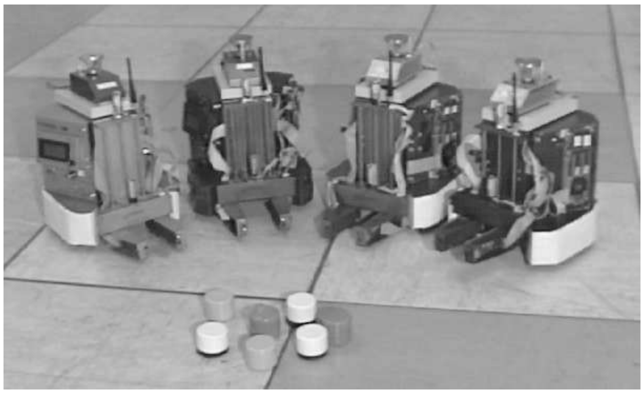


Figure 1-12: The mobile robots used to validate the group behavior and learning methodologies described in this thesis. These robots demonstrated learning to forage by using group safe–wandering, following, and resting behaviors.

Conditions are many fewer than states, thus greatly diminishing the agent's learning space and speeding up any RL algorithm.

In addition to the use of behaviors and conditions, we propose two ways of shaping the reinforcement function in order to aid the learner in a nondeterministic, noisy, and dynamic environment. We introduced *heterogeneous reward functions* that partition the task into subgoals, thus providing more immediate reinforcement. Within a single behavior (i.e., a single goal), we introduced *progress estimators*, functions associated with particular conditions that provided some metric of the learner's performance. Progress estimators, or internal critics, decrease the learner's sensitivity to noise, minimize thrashing, and minimize the effect of fortuitous rewards by correlating some domain knowledge about progress with appropriate behaviors the agent has taken in the past. The details of the reformulation are given in Chapter 7.

The proposed formulation was validated on the task of learning to associate the conditions and behaviors for group foraging with a collection of robots. The behaviors included the foraging subset of basic behaviors: *safe–wandering*, *dispersion*, and *homing*, augmented with *grasping* and *dropping*, as well as with *resting*, a new behavior triggered by an internal "day–time night–time" clock. By clustering, the condition set was reduced to the power set of the following predicates: *have-puck?*, *at-home?*, *night-time?*, and *near-intruder?*.

A smaller group of robots with more reliable hardware was used for the learning experiments. In terms of sensors and effectors, the robots were functionally identical to the first set (Figure 1-12), and the implemented basic behaviors and combinations were directly portable.

Three learning algorithms were implemented and tested on the foraging task. The

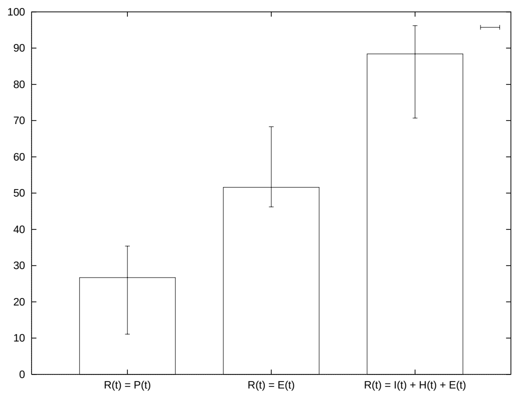


Figure 1-13: The performance of the three reinforcement strategies on learning to forage. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned, averaged over twenty trials.

first was standard Q-learning, while the other two simply summed the reinforcement received over time.

Q-learning received a reward whenever a robot dropped a puck in the home region. The second algorithm was based on the reinforcement received from heterogeneous reward functions based on reaching subgoals including grasping and dropping pucks, and reaching home. The third algorithm used reinforcement both from the heterogeneous reward functions and from two progress estimators: one monitoring progress in getting away from an intruder, and the other monitoring progress toward home. The two progress estimators were found to be sufficient for making the given learning task possible and for consistent and complete learning performance. The absence of either one disabled the robots from learning the complete policy.

The performance of each of the three algorithms was averaged over 20 trials (Figure 1-13). The analysis of the learning performance showed that the parts that were not learned by the first two algorithms relied on the progress estimators and were successfully learned in the third case. Detailed analysis of the results is given in Chapter 8.

1.3 Thesis Outline

The preceding sections briefly summarized the contributions of the thesis. This section outlines the structure of the thesis and summarizes each of the chapters.

Chapters 2 through 5 deal with synthesizing and analyzing group behavior. Chapters 6 through 8 address learning in multi–agent domains. Readers interested in moving directly to the details of the basic behavior approach should skip to Chapter 4. Those interested in going directly to the learning part of the thesis should skip to Chapter 6. All newly introduced, ambiguous, or frequently used terms are defined in Appendix B. The following are summaries of the chapter contents.

Chapter 2 describes the biological, sociological, and pragmatic motivation behind this work. It describes the key issues in individual and multi agent control, and introduces and defines the main concepts of the thesis.

Chapter 3 presents an overview of related work in Robotics, Simulation, Artificial Life, Distributed AI, and analysis of behavior.

Chapter 4 introduces the basic behavior approach, describes the methodology for selecting basic behaviors, and illustrates the process by defining the basic behaviors for a collection of mobile agents interacting in the plane. The chapter describes the experimental environments, basic behavior specifications and algorithms, and the empirical data and the criteria for evaluating the performance of each of the behaviors as well as their efficacy relative to centralized alternatives.

Chapter 5 describes two methodologies for combining basic behaviors into more complex, higher–level behaviors. The methodologies are demonstrated by combining the basic behaviors described in Chapter 4 to generate three different kinds of higher–level behaviors and evaluate their performance. This chapter also discusses methods for minimizing interference between behaviors within an agent.

Chapter 6 motivates learning in situated agents and reviews the existing learning work based on the type of information being acquired by the agent. It then defines the group learning problem discussed in the thesis as an instance of reinforcement learning (RL) and overviews existing RL models and algorithms as applied to the situated agent domain.

Chapter 7 describes a formulation of RL that enables and facilitates learning in our complex situated multi–agent domain. It introduces the use of behaviors and their conditions in place of actions and states, and describes a method for shaping the learning process through the use of heterogeneous reward functions and progress estimators.

Chapter 8 presents the experimental robot environment and the learning task used to validate the methodologies proposed in Chapter 7. It describes the experimental design, the three learning algorithms that were implemented and compared, and discusses the results. In conclusion, the chapter addresses extensions of the presented work including the problem of learning social rules and multiple concurrent tasks.

Chapter 9 summarizes the thesis.

Chapter 2

Motivation and Issues in Agent Control

Why study multiple agents?

The motivation for this work comes from two quite different but complementary directions: the desire to understand and analyze natural systems and the need to design and synthesize artificial ones.

2.1 Biological and Sociological Motivation

Intelligence is a social phenomenon. Most intelligent animals live, obey the rules, and reap the benefits of a society of kin. Societies vary in size and complexity, but have a key common property: they provide and maintain a shared culture (Gould 1982).

Culture is both a result and a cause of intelligent behavior. Intelligent creatures create and refine social rules in order to perpetuate the society. These rules constitute a culture which is communicated and shared by the society, and has important effects on its individual members (Gould 1982, McFarland 1987).

Culture allows for genetic parsimony. Social interaction is used to transfer information across generations, though social learning (McFarland 1985). Thus, less genetic investment is necessary, as fewer abilities need to be innate. Interestingly, as culture adapts, the growing complexity of social rules makes increased demands on individual intelligence, specifically on the ability to absorb and adapt to the culture. Humans are an extreme example of cultural complexity, requiring the longest learning and training developmental period of all animals (Gould 1982).

Culture allows for faster adaptation. As an alternative to evolution, culture allows for testing and adapting social behaviors at a much shorter time scale. Social interactions can be created and destroyed within a single generation. For example, elephants have been shown to learn to avoid humans even if no harm was inflicted for generations, based on a distant cultural memory of past abuse (Gould 1982).

Culture allows for Lamarckian evolution. It enables the direct transfer of learned information to future generations. A single individual's discovery can be adopted by an entire population and passed on. For example, an individual Japanese macaque monkey discovered washing of sweet potatoes. The practice was transmitted culturally through the society and on to later generations (Gould 1982).

Culture makes up for genetic deficiencies. Social interactions can compensate for individual limitations, both in terms of physical and cognitive capabilities. For example, group organizations, such as herds and packs, allow animals to attack larger prey, share information, and increase the chance of mating and survival (McFarland 1985).

In order to be understood, individual intelligence must be observed and analyzed within its social and therefore cultural context. In contrast to traditional AI, which addresses intelligence as an individual phenomenon, this work is based on the belief that intelligent behavior is inextricably tied to its cultural context and cannot be understood in isolation. The emphasis is similar to the principles of ethology, the study of animal behavior. Unlike the behaviorist branch of biology, which studies animals in controlled laboratory settings, ethology observes animals in their natural habitats. This research attempts to study intelligent behavior in its natural habitat: situated within a society.

The complexity of culture results from the interactions among individuals. This research will focus on exploring simple social interactions which result in purposive group behaviors, with the goal of:

1. understanding social and group behavior in nature, and
2. developing a methodology for principled design of group behavior in artificial systems.

The study of social agents and culture as a basis and structure of intelligent behavior is exploratory. Thus, the part of the thesis that addresses that domain is phenomenological, but hopefully also scientific in its attempt to understand natural phenomena and explain them in principled terms.

2.2 Pragmatic Motivation

While nature offers challenges for analysis, engineering demands synthesis. In particular, it strives for efficient, automated, reliable, and repeatable methods of synthesizing useful systems.

Discoveries about systems of multiple interacting agents can be applied to many parallelizable problems. The idea of applying multiple computational (or physical) agents to a variety of distributed domains, from terrain exploration and mapping, to fire fighting, harvesting, and micro surgery, has been around for many years. However, in spite of the potentially numerous applications, the distributed, multi–agent approach is an exception rather than the rule in most domains.

Parallel, decentralized, non–hierarchical computation requires a paradigm shift (Resnick 1992). Regardless of the domain of application, this approach raises a number of difficult issues. The particular few that motivate this research and are addressed in this thesis are:

- What common properties and principles of organization are shared among different domains of application of multi–agent systems?
- How do the interactions of the individuals affect the behavior of the group?
- How does the group get the job done?
- How much does each individual need to know about the group, the task, the environment, and the other agents?
- How much does each individual need to communicate with others in order to get the job done?
- What are the simplest agents and rules we can use to generate complex and useful group behaviors?

This research is aimed at finding common properties across various domains of multi–agent interaction. Identifying these properties allows for classifying group behaviors into common categories and thus simplifies the process of both design and analysis.

The next section defines key terms.

2.3 Key Issues, Terms, and Definitions

2.3.1 Behaviors and Goals

The notion of behavior is the main building block of this work. In the last few years the use of behaviors has been popularized in the AI, control, and learning communities. Approaches labeled "behavior–based AI" and "behavior–based control" are becoming mainstream, but behavior is yet to be cleanly defined and circumscribed.

We define *behavior* to be a control law for reaching and/or maintaining a particular goal. For example, in the robot domain, *following* is a control law that takes inputs from an agent's sensors and uses them to generate actions which will keep the agent moving within a fixed region behind another moving object. In our work, a behavior is based on the sensory input vector only, and does not use internal state. We do not, however, exclude the use of state in the behavior definition, but reserve it for tasks where it is needed.

The above definition of behavior specifies that a behavior is a type of an operator that guarantees a particular goal. In order to serve as general building blocks, basic behaviors must be capable of dealing with both attaining and maintaining goals. *Attainment goals* are terminal states; having reached a goal, the agent is finished. Such goals include reaching a home region and picking up an object. *Maintenance goals* persist in time, and are not always representable with terminal states, but rather with dynamic equilibria that must be maintained. Examples include avoiding obstacles and minimizing interference. Maintenance goals can usually be expressed as sequences of achievement goals but may require fine granularity of description. Situated agents can have multiple concurrent goals, including at least one attainment goal, and one or more maintenance goals.

This thesis will attempt to show that behaviors are a natural, convenient, and efficient abstraction for control, planning, and learning for situated agents. Behaviors hide the low–level details of control that deal with precise control parameters. They allow for specifying robot tasks and goals in terms of higher–level primitives that cut down on the state space and are more intuitive for the user. Finally, they are a good basis for learning in noisy and uncertain situated domains.

Ensemble, collective or *group behavior* is an observer–defined temporal pattern of interactions between multiple agents. Of the innumerably many possible such behaviors for a given domain, only a small subset is relevant and desirable for achieving the agents' goals.

2.3.2 Interaction

Interaction is another foundational concept in this work. Typically, interaction is viewed as any influence that affects an agent's behavior. By this definition, an agent interacts with everything it can sense or be affected by, since all of its external (observable) and internal state can have an impact on its actions.

This work is largely concerned with the interaction that takes place between agents. Thus we propose a stricter definition: *interaction is mutual influence on behavior*. Consequently, objects in the world do not interact with agents, although they may affect their behavior. The presence of an object affects the agent, but the agent does not affect the object since objects, by definition, do not behave, only agents do.

2.3.3 Domain Description

Having defined the key concepts of the thesis, behavior and interaction, we turn to the specification of the domain being addressed.

In order to focus and constrain the study of group behavior, this work focuses on interactions among situated, embodied agents. Some key constraints were imposed on the experimental domain in order to structure the exploration while still providing sufficient variety of behaviors to study. The following are the key constraining properties:

- Agents are homogeneous.
- Agents do not use explicit models of each other.
- Agents do not use directed communication or explicit cooperation.

The reasons for and implications of each of the constraints are described and discussed in the following sections.

Implications of Homogeneity

This work focuses on groups of agents that are *homogeneous* in that they are situated in the same world and have the same goal structure (in our case translating into the same behavior set)[1]. Homogeneous agents will also be referred to as *similar*,

[1] Furthermore, the agents in this work are embodied with similar dynamics. While the dynamics of simulated agents can be made identical, those of physical robots often vary enough to significantly affect their group behavior. The section on hardware limitations explains this in detail. The terms homogeneous and similar will be used interchangeably.

as distinct from *identical*, a property that can be ascribed to SIMD–style agents. Homogeneity has some important implications.

Predictability

The fact that all agents are similar makes their behavior *predictable* to one another in that they do not require internal explicit models of each other. This predictability can be used explicitly, by allowing agents to infer other agents' actions and use that information to make individual decisions, or implicitly, to simplify the control of each individual. This work focuses on the latter approach. For example, identical control laws can take advantage of inherent symmetries in spatial and more abstract domains.

Homogeneity minimizes goal–related conflicts and resulting strategies such as deserting and cheating. Furthermore, homogeneity allows for leaving much of the information about the world implicit. Although the agents in this work do not use explicit expectations about other agents' behavior, their decision process implicitly takes that information into account.

Given their similarity, agents do not need identities and thus do not require abilities for identification. This presents a significant cognitive savings. As homogeneity and similarity greatly reduce individual cognitive requirements, they can be used for simplifying the synthesis and understanding of group behavior.

Finally, homogeneity can result in increased global robustness through redundancy. Failure of any subset of agents should not seriously affect the system, since the agents are similar and thus interchangeable, and no particular agent or group of agents is critical for the accomplishment of the task. To preserve robustness, no specific roles, such as leaders and followers are designated *a priori*. However, temporally (and spatially) local, replaceable leaders may emerge in various situations.

2.3.4 Recognition of Kin

Taking advantage of homogeneity depends on a critical property: the agents must be able to recognize other similar agents. We postulate that the ability to distinguish the agents with whom one is interacting from everything else in the environment is a *necessary condition* for intelligent interaction and group behavior. This ability is innate and ubiquitous in nature, and enables almost all creatures to distinguish others of their own kind, and more specifically to recognize kin from others (McFarland 1985, McFarland 1987).

It is important to note that species and kin recognition need not be explicit, i.e., the agent need not "know" or "be aware" that the other agent being recognized is kin, as long as its response to it is kin–specific. For example, slime mold bases its behavior on the concentration of slime produced by its kin. It cannot be said that it actively "recognizes" kin but it does act in species–specific ways which result in complex group behavior such as the construction of multi–cellular organisms (Kessin & Campagne 1992). Similarly, ants cannot be presumed to "know" that pheromones they sense are produced by their conspecifics. However, the appropriate responses to those pheromones result in the formation of trails and other complex structures (Franks 1989).

Besides being biologically inspired, the ability to recognize kin is pragmatic as it allows even the simplest of rules to produce purposive collective behavior.

2.3.5 Mental Models and Theory of Mind

A dominant school of thought in cognitive psychology and AI is based on the premise that social interaction requires a *theory of mind* (Premack & Woodruff 1978). Namely, in order to engage in social discourse, agents need to have mental models of each other, attribute mental states to each other, understand each other's intentions, and maintain beliefs about each other (Dennett 1987, Cheney & Seyfarth 1990). Indeed, an entire field of theory of the mind rests on the necessity of inferring the internal workings of the mind of the agent(s) with whom one is interacting (Read & Miller 1993).

Maintaining a theory of mind is a complex task and requires a high computational and cognitive overhead (Gasser & Huhns 1989, Rosenschein & Genesereth 1985, Axelrod 1984). Further, controversy surrounds its necessity, as work in both developmental psychology and ethology indicates that theory of mind is not necessary for a large repertoire of complex social interactions (Tomasello, Kruger & Rather 1992, Cheney & Seyfarth 1990, McFarland 1987, Gould 1982, Rosenthal & Zimmerman 1978).

Research in developmental psychology has shown that young children engage in various forms of social interaction even before attaining the sense of self–awareness, a necessary component of constructing a theory of mind. Prior to this stage, occurring around the age of two, children are incapable of separating the internal and external perception of the world (Piaget 1962, Bandura & Walters 1963, Bandura 1971). Even after achieving self–awareness, as determined with the typical dot–and–mirror test (Asendorpf & Baudonniere 1993), around the age of two, children require a number of years before reaching the adult ability to form theories of mind (Bandura 1977, Abravanel & Gingold 1985).

Much research has been aimed at testing whether primates have theories of mind. It has recently been demonstrated that certain species of monkeys, while involved in complex social and cooperative interactions, apparently do not form theories of mind at all (Cheney & Seyfarth 1990, Cheney & Seyfarth 1991). In contrast, chimps appear to have more complex abilities and are indeed able to infer goals of their conspecifics (Cheney & Seyfarth 1990, McFarland 1987). How the internal models are represented and whether they are based on explicit or internal representations, remains open for further study (Gomez 1991).

An Alternative to the Theory of Mind

Exploring the existence and limits of theory of mind in biology is difficult. The type and amount of knowledge and representation that an animal brings to bear in its social interactions is impossible to circumscribe. In an ideal scenario the experimenter would be able to control for the type and amount of this knowledge and test the resulting behavior, in order to determine what is necessary and what is not.

Computational and robot experiments allow us to do just that. The agents being experimented with are much simpler than those in nature, but it is exactly this simplicity that allows us to focus on the specific question of internal social models. In order to study the necessity of theory of mind, this work started from the bottom up, by exploring agents which had none at all.

This work studies group behaviors resulting from the simplest interactions among the simplest of agents. The agents have no explicit models of each other, expectations, or intentions. The goal of this approach is to demonstrate what types of complex interactions can be achieved with such simple basic abilities. The results demonstrate that, particularly in homogeneous groups, significant amount of information about an individual's goals is reflected in the observable external state and behavior, and can be obtained with no direct communication (Cheney & Seyfarth 1990). Consequently, a theory of mind is not necessary for a broad spectrum of behaviors, nor is direct communication. More related issues in communication are discussed next.

2.3.6 Communication and Cooperation

Communication and cooperation have become popular topics in both abstract and applied multi–agent work (for example see Yanco & Stein (1993), Dudek, Jenkin, Milios & Wilkes (1993), Altenburg & Pavicic (1993), and others). Communication is the most common means of interaction among intelligent agents. Any observable behavior and its consequences can be interpreted as a form of communication so for purposes of clarity, we propose some clarifying definitions.

Direct communication is a purely communicative act, one with the sole purpose of transmitting information, such as a speech act, or a transmission of a radio message. Even more specifically, *directed communication* is direct communication aimed at a particular receiver. Directed communication can be one–to–one or one–to–many, but in both cases the receivers are identified.

In contrast, *indirect communication* is based on the observed behavior, not communication, of other agents, and its effects on the environment. This type of communication is referred to as *stigmergic* in biological literature, where it refers to communication based on modifications of the environment rather than direct message passing.

Both direct and indirect communication are practiced by most species in nature. For example, bees use signals, such as the waggle dance, with the sole purpose of transmitting information and recruiting. In contrast, they also use cues, such as the direction of their flight, which transmit hive information as a by–product of their other behaviors (Seeley 1989).

Cooperation is a form of interaction, usually based on communication. Certain types of cooperative behavior depend on directed communication. Specifically, any cooperative behaviors that require negotiation between agents depend on directed communication in order to assign particular tasks to the participants.

Analogously to communication, *explicit cooperation* is defined as a set of interactions which involve exchanging information or performing actions in order to benefit another agent. In contrast, *implicit cooperation* consists of actions that are a part of the agent's own goal–achieving behavior repertoire, but have effects in the world that help other agents achieve their goals.

Having defined precise terminology, the communication and the resulting cooperation constraints imposed on the experimental domain can now be described. In order to study the role of communication in a controlled fashion, and to explore how much communication is needed for the group behaviors described here, a minimalist approach was chosen.

No directed, one–to–one communication between the agents was used in any of the experiments. Indirect communication was based on sensing the external state of neighboring agents, as well as sensing their density, and the effects of their actions. Direct communication was undirected, and limited to local broadcast: agents could transmit messages that could be received by others. However, the agents did not have the ability to choose the receivers of their messages, and thus to engage in directed communication.

The undirected communication constraint affects the kinds of communication that can be implemented or can emerge in a multi–agent system. This work focuses on implicit cooperation without explicit task sharing. For example, instead of addressing the task of moving a large object by many agents, this work deals with distributed solutions to problems like moving numerous small objects, a task that can be solved by a single agent, but can benefit from well–designed multi–agent solutions. For an alternative perspective, see Parker (1994).

2.4 Issues in Agent Control

This section describes and specifies the problem of controlling a multi–agent system by first overviewing approaches to individual agent control, and then discussing their extensions to multiple agents.

Multi–agent research covers a vast array of natural and artificial systems, ranging from the brain to operating systems, and from bird flocks to collection of robots. For the purposes of this work, an *agent* is a process capable of perception, computation, and action within its world[2]. A *multi–agent system* consists of two or more such agents.

The problem of *multi–agent control* can be viewed at the individual agent level and the collective level. The two levels are interdependent and the design of one is, or should be, strongly influenced by the other. However, multi–agent control has grown out of individual agent control, and this history is often reflected in the control strategies at the collective level. The next section describes the main approaches to individual agent control and their extensions and applicability to multi–agent domains.

2.4.1 Individual Agent Control

At one extreme of the agent control spectrum lie traditional top–down *planner–based, deliberative* strategies that use a centralized world model for verifying sensory information and generating actions in the world (Giralt, Chatila & Vaisset 1983, Chatila & Laumond 1985, Moravec & Cho 1989, Laird & Rosenbloom 1990). The information in the world model is used by the planner to produce the most appropriate sequence of actions for the task at hand. These approaches allow for explicitly formulating the task and goals of the system, and estimating the quality of the agent's performance. However, uncertainly in sensing and action and changes in the environment can require frequent replanning the cost of which may be prohibitive for complex systems. Planner–based approaches have been criticized for scaling poorly with the complexity of the problem and consequently not allowing for reaction in real–time (Brooks 1990*b*, Brooks 1991*c*).

[2]The world may or may not be physical.

Various attempts at achieving real–time performance have been proposed. Perhaps the most prominent are *purely reactive* bottom–up approaches which implement the agent's control strategy as a collection of preprogrammed condition–action pairs with minimal state (Brooks & Connell 1986, Agre & Chapman 1987, Connell 1990). These systems maintain no internal models and perform no search, but simply look–up and command the appropriate action for each set of sensor readings. They rely on a direct coupling between sensing and action, and fast feedback from the environment. Purely reactive strategies have proven effective for a variety of problems that can be well defined at design–time, but are inflexible at run–time due to their inability to store information dynamically (Matarić 1992*a*).

The division between reactive and deliberative strategies can be drawn based on the type and amount of computation performed at run–time. Reactive, constant–time run–time strategies can be derived from a planner, by computing all possible plans off–line in advance. For example, situated automata achieve real–time performance by compiling all of the system's goals and the ways of their achievement into a language that compiles into circuits with constant–time computation properties (Rosenschein & Kaelbling 1986). In general, the entire control system of an agent can be precompiled as a decision graph into a collection of reactive rules ("universal plans") (Schoppers 1987). While theoretically appealing, these strategies often scale poorly with the complexity of the environment and the agent's control system.

Hybrid architectures attempt a compromise between purely reactive and deliberative approaches, usually by employing a reactive system for low–level control, and a planner for higher–level decision making. Hybrid systems span a large and diverse body of research. It includes *reactive planning or reactive execution* used in Reactive Action Packages (RAPs), higher–level primitives for planning which hide and take care of the details of execution (Firby 1987), and PRS (Procedural Reasoning System), an architecture for flexible control rule invocation (Georgeff & Lansky 1987), Schemas (Arkin 1989), and several others (Payton 1990, Connell 1991). These systems tend to separate the control system into two or more communicating but otherwise independent parts. In most cases, the low–level reactive process takes care of the immediate safety of the robot, while the higher level uses the planner to select action sequences.

Behavior–based approaches are an extension of reactive systems that also fall between the purely reactive and the planner–based extremes (Brooks 1986, Maes 1989). Although often confused in the literature, behavior–based strategies are strictly more powerful than purely reactive approaches since they have no fundamental limitations on internal state. While behavior–based systems embody some of the properties of reactive systems, and usually contain reactive components, their computation is not limited to look–up. Other than centralized reasoning engine and representation, these systems may use different forms of distributed internal representations and perform distributed computations on them in order to decide what effector action to take (Matarić 1992*a*).

A comparative classification of above methodologies based on domains of applicability has not yet been undertaken.

2.4.2 Multi–Agent Control

Having overviewed single–agent control, this section discusses how the described approaches scale to multi–agent problems.

Extending the planning paradigm[3] from single–agent to multi–agent domains requires expanding the global state space to include the state of each of the agents. Such a global state space is exponential in the number of agents. Specifically, the size of the global state space G is: $|G| = s^a$ where s is the size of the state space of each agent, here assumed to be equal for all agents, or at worst the maximum for all agents, and a is the number of agents. Exponential growth of the state space makes the problem of global on–line planning intractable for all but the smallest group sizes, unless control is synchronized and has SIMD form[4]. Further, since global planning requires communication between the agents and the controller, the bandwidth can grow with the number of agents. Additionally, the uncertainty in perceiving state grows with the increased complexity of the environment. Consequently, global planner–based approaches to control do not appear well suited for problems involving multiple agents acting in real–time based on uncertain sensory information.

At the other end of the control spectrum, extending the reactive and behavior–based approaches to multi–agent domain results in completely distributed systems with no centralized controller. The systems are identical at the local and global levels: at the global level the systems are a collection of reactive agents each executing task–related rules relying only on local sensing and communication. Since all control in such distributed systems is local, it scales well with the number of agents, does not require global communication, and is more robust to sensor and effector errors. However, global consequences of local interactions between agents are difficult to predict.

The following table summarizes the properties of these two approaches to multi–agent control:

[3]The planning paradigm includes includes traditional and hybrid systems. In terms of multi–agent extensions, hybrid systems fit into the planner–based category since their collective behavior is generally a result of a plan produced by a global controller.

[4]All agents perform the same behavior at the same time.

centralized approaches	**distributed approaches**
can optimize global parameters	can only optimize locally
scale poorly	scale well
require global sensing	use local sensing
require global communication	may not require communication
can have a computational bottleneck	no computational bottleneck
impose hierarchical control	use flat control
not usually redundant	are usually redundant

Table 2.1: A comparative summary of typical centralized and distributed approaches.

Centralized approaches have the advantage of potential theoretical analysis. In contrast, parallel distributed systems typically do not lend themselves to traditional analytical procedure.

2.4.3 Analysis of Behavior

This thesis focuses on *fully distributed multi–agent systems*, those in which the behavior of each agent is determined by its own control system rather than by a centralized controller. Such systems are by definition complex, because they are composed of a large number of elements, or because the inter–element interactions are not simple. Multi–agent systems consisting of several situated agents with uncertain sensors and effectors display both types of complexity. This section addresses how these properties affect their behavior and its analysis.

The exact behavior of an agent situated in a nondeterministic world, subject to real error and noise, and using even the simplest of algorithms, is impossible to predict exactly. By induction, the exact behavior of each part of a multi–agent system of such nature is also unpredictable. However, according to Simon (1969), a system is analyzable, and thus well designed, if it is *decomposable* into non–interacting modules. Thus, minimizing inter–module interactions is considered good engineering and principled AI, and most of traditional Artificial Intelligence relies on this style of top–down modularity. In contrast, nature abounds with complex systems whose global behavior results from precisely the type of interactions that current research methodologies try to avoid. These effects can be found at all scales, from the subatomic (Gutzwiller 1992), to the semantic (Minsky 1986), to the social (Deneubourg, Goss, Franks, Sendova-Franks, Detrain & Chretien 1990).

Situated behavior is based on the interaction with, and thus feedback from, the environment and other agents. Both negative and positive feedback are relevant. Negative feedback has a regulatory effect, damping the system's response to external influences, while positive feedback has an amplifying effect, increasing the system's response. In the multi–agent spatial domain, for example, negative feedback controls the local structure among the agents while positive feedback recruits more agents into the structure.

Behaviors based on positive feedback usually require a critical mass to initiate and accelerate with increased group size. All of these behaviors are variations on *recruitment*; the more agents that are engaged in an activity, the more agents that join in. Such behaviors are usually unstable as they are sensitive to the particular conditions and resources required to maintain the recruitment effect. Numerous natural group behaviors are based on positive feedback: lynch mobs, public polls, popularity ratings, traffic jams, ant trails, and worker recruitment in both ants and bees are all instances of positive feedback (Camazine 1993, Deneubourg et al. 1990, Deneubourg, Aron, Goss, Pasteels & Duernick 1986).

A group of interacting agents is a dynamical system. Global behavior of such a complex systems is determined by the local interactions between individuals. These interactions merit careful study in order to understand the global behavior. In natural systems, such interactions result in the evolution of complex and stable behaviors that are difficult to analyze using traditional, top–down approaches. We postulate that in order to reach that level of complexity synthetically, such behaviors must be generated through a similar, interaction–driven, incrementally refined process.

Precise analysis and prediction of the behavior of a single situated agent, specifically, a mobile robot in the physical world, is an unsolved problem in robotics and AI. Previous work has shown that synthesis and analysis of correct plans in the presence of uncertainty can be intractable even in highly constrained domains (Lozano-Pérez, Mason & Taylor 1984, Canny 1988, Erdmann 1989) and even on the simplest of systems (Smithers 1994). Physical environments pose a great challenge as they usually do not contain the structure, determinism, and thus predictability usually required for formal analysis (Brooks 1991*c*, Brooks 1991*b*). Predicting the behavior of a multi–agent system is more complex than the single–agent case. The difficulty in analyzing comes from two properties intrinsic to complex systems:

1. the actions of an agent depend on the states/actions of other agents,

2. the behavior of the system as a whole is determined by the interactions between the agents rather than by individual behavior.

In general, no mathematical tools are available for predicting the behavior of a system with several, but not numerous, relatively complex interacting components, namely a collection of situated agents. In contrast to physical particle systems, which consist of large numbers of simple elements, multi–agent systems in nature and AI are defined by comparatively small groups of much more complex agents. Statistical methods used for analyzing particle systems do not directly apply as they require minimal interactions between the components (Weisbuch 1991, Wiggins 1990).

Instead of attempting to analyze arbitrary complex behaviors, this work focuses on providing a set of behavior primitives that can be used for synthesizing and analyzing a particular type of complex multi–agent systems. The primitives provide a programming language for designing analyzable control programs and resulting group behaviors.

2.4.4 Emergent Behavior

Emergent behavior is a popular topic of research in the field of complex systems (see Forrest (1989), Langton (1989), Langton (1990), and Steels (1994*a*) for overviews). Such behavior is characterized by the following property: it is manifested by global states or time–extended patterns that are not explicitly programmed in but result from local interactions between a system's components. Because emergent phenomena are by definition observed at a global level, they depend on the existence of an observer.

Emergent behavior can be observed in any sufficiently complex system, i.e., a system which contains local interactions with temporal and/or spatial consequences. Perhaps because of their pervasiveness, emergent phenomena have been objects of interest, although perhaps not objects of analytical study, for a long time. The property of observer–dependence make emergent phenomena more difficult to study. Kolen & Pollack (1993) eloquently describe why in general the complexity of a physical system is not an intrinsic property but is dependent on the observer, and further why traditional measures of complexity are insufficient for physical systems. Subjective evaluation is also discussed by Bonabeau (1993).

Emergent phenomena are appealing to some researchers because they appear to provide something for nothing. These types of systems are referred to as "self–organizing" because of their apparent ability to create order. In reality, the dynamics of such self–organizing systems are carefully crafted (usually by eons of evolution) to produce the end–results. Theoretical analysis of multi–agent systems of the type used in this research is difficult, and, as will be argued, exact prediction of the behavior of such systems is not currently within reach. Consequently, work on situated group behavior can benefit from synthesis and experimentation.

Emergent behaviors result from systems that are complex enough to defy our

approach	**level of description**
complex dynamics	microscopic & continuous
<?>	macroscopic & quasi–continuous
state spaces	macroscopic & discrete

Table 2.2: A desirable level of system description for control and analysis lies between the commonly employed ends of the spectrum.

current tools for predictive analysis, and require simulation for prediction (Darley 1994). In order to structure and simplify this process of experimental behavior design, this work will provide a set of basic group behaviors and methods for synthesizing them from local rules. These basic behaviors and their combinations are emergent in that they result from the local interactions, but are predictable and well understood.

2.4.5 Limits of Analysis

The difficulty in analyzing complex multi–agent systems lies in the level of system description. Descriptions used for control are usually low level, detailed, and continuous. In contrast, planning and analysis are usually done at a high level, often using an abstract, discrete model. A more desirable and manageable level may lie in between those two, as depicted in Table 2.2.

In general, this work is concerned with predicting the global behavior of the system rather than the precise behavior of any of its components. At the high level of precision requiring a detailed level of description, most interactions are chaotic and unpredictable (Kolen & Pollack 1993). The goal of analysis is to gain predictive power by modeling the system at the right level. In the case of artificial complex systems, however, it is not possible to determine that level without generating and testing the system itself.

For the case of a fully deterministic agent and world, it is possible, but usually not realistic, to enumerate all trajectories the agent can take in its action or behavior space. This is equivalent to elaborating the agent's phase space. Early AI methods for proving correctness consisted of showing that, for a given set of possible initial conditions, usually expressed as discrete states, the agent would, through a series of actions, reach the desired terminal state, often designed to be the goal. Search–based methods for plan or action generation are particularly amenable to this type of analysis (Fikes & Nilsson 1971). However, besides the scaling problem, this approach to behavior analysis fails in more realistic worlds in which both the agent and the environment are not deterministic.

State transitions in nondeterministic worlds can be modeled probabilistically (e.g., Doyle & Sacks (1989)) but obtaining appropriate values for the probabilities is in general very difficult since it requires a complete and accurate model of the world. Even small inaccuracies in the values can accrue and result in artifactual dynamics at the global level. Consequently, most probabilistic models fail to capture the stochastic dynamics of the kinds of complex behavior this work is concerned with.

The crux of the problem, as before, is determining the appropriate level of system description. Quantitative analysis is extremely difficult for any but the simplest of deterministic systems. This may not appear to be a problem, as most researchers would be satisfied with knowing the system's global, qualitative behavior. Global behavior, however, is generally defined in quantitative terms from which qualitative descriptions are derived, whether it be on the microscopic scale of particle interactions (Abraham & Shaw 1992) or on the macroscopic scale of building maps (Chatila & Laumond 1985) of the environment.

The path to a qualitative description of a system is indirect, requiring abstracting away the details or through clustering analytical, quantitative information. A qualitative description is a collection of non–analytic symbols (i.e., words instead of numbers) with complicated associated semantics. When these semantics are defined, they are either stated in terms of other symbols or eventually grounded in numerical terms.

Given the difficulty of the problem, most analytical approaches to date have been limited to constrained special–case scenarios. This is not surprising since any general method for analyzing complex systems with interacting components is unlikely to be powerful enough to provide useful predictions.

Since prediction of group behavior is too difficult from the individual perspective, approaches that focus on describing and analyzing ensemble properties appear better suited for the domains addressed in this work. The next section describes an approach to assessing and qualitatively predicting global behavior by measuring interference, a local property that has collective consequences.

2.4.6 Interference and Conflict

Interference is any influence that opposes or blocks an agents' goal–driven behavior. In societies consisting of agents with identical goals, interference manifests itself as competition for shared resources. In diverse societies, where agents' goals differ, more complex conflicts can arise, including goal clobbering[5], deadlocks, and oscillations.

[5]The term is used in the same sense as in Sussman & McDermott (1972) and Chapman (1987).

Two functionally distinct types of interference are relevant to this work: interference caused by multiplicity called *resource competition*, and interference caused by goal–related conflict called *goal competition*.

Resource competition includes any interference resulting from multiple agents competing for common resources, such as space, information, or objects. As the size of the group grows, this type of interference increases, causing the decline in global performance, and presenting an impetus for social rules.

Resource competition manifests itself in homogeneous and heterogeneous groups of coexisting agents. In contrast, goal competition arises between agents with different goals. Such agents may have identical high–level goals (such as, for example, a family has), but individuals can pursue different and potentially interfering subgoals at any particular instance, i.e., they can be "functionally heterogeneous." Such heterogeneity does not arise in SIMD–style groups of functionally identical agents in which all are executing exactly the same program at each point in time.

Goal competition is studied primarily by the Distributed AI community (Gasser & Huhns 1989). It usually involves predicting other agents' goals and intentions, thus requiring agents to maintain models of each other (e.g., Huber & Durfee (1993) and Miceli & Cesta (1993)). Such prediction abilities require computational resources that do not scale well with increased group sizes[6]. In contrast, in the work discussed here, goal competition, and thus the need for agents to model each other, is minimized by agent homogeneity, and we focus largely on issues of direct resource competition.

2.4.7 Individual vs. Group Benefit

Social rules attempt to eliminate or at least minimize both resource and goal competition. In particular, their purpose is to direct behavior away from individual greediness and toward global efficiency[7]. In certain groups and tasks, agents must give up individual optimality in favor of collective efficiency. In those cases, greedy individualistic strategies perform poorly in group situations because resource competition grows with the size of the group. The agents described here fall into this category.

Since social rules are designed for optimizing global resources, it is in the interest of each of the individuals to obey them. However, since the connection between individual and collective benefit is rarely direct, societies can harbor deserters who disobey social rules in favor of individual benefit. Game theory offers elaborate studies of the effects of deserters on individual optimality (Axelrod 1984), but domains

[6]The problem of maintaining internal models or so called theories of mind is discussed in detail in section 2.3.5.

[7]In cultural contexts global efficiency is sometimes elevated to "the common good."

treated in game theory are much more cleanly constrained than those treated here. In particular, game theory deals with rational agents capable of evaluating the utility of their actions and strategies. In contrast, our work is concerned with situated agent domains where the agents cannot be assumed to be rational due to incomplete or nonexistent world models and models of other agents, inconsistent reinforcement, and noise and uncertainty.

Furthermore, the goal of this work is not to devise optimal strategies for a specific group behavior but to provide methodologies for finding efficient approaches to a variety of related problems. Optimality criteria for agents situated in physical worlds and maintaining long–term achievement and maintenance goals are difficult to characterize and even more difficult to achieve. While in game theory interference is a part of a competing agent's predictable strategy, in the embodied multi–agent domain interference is largely a result of direct resource competition, which can be moderated with relatively simple social rules. For example, complex traffic jams can be alleviated through the appropriate use of yielding.

2.4.8 Estimating Interference

Understanding interference is an integral part of synthesizing and analyzing group behavior. In synthesis, the task must be distributed over multiple agents in a way that minimizes interference, or the benefits of concurrent execution are lost. In analysis, interference must be taken into account in order to characterize the realistic behavior of a distributed system as well as motivate the existence of social rules and protocols.

Attempting to precisely predict inter–agent interference is equivalent to trying to predict the system's exact behavior. As has been argued about analysis in general, this level of prediction is impossible to reach. This section proposes a qualitative alternative that can be applied to obtain useful estimates.

Agent density is a key parameter for estimating interference since it measures likelihood of interaction. The higher the density the higher the probability that any two agents will encounter each other and interact. Even without evaluating the outcome of interaction, being able to predict its estimated frequency is a useful part of describing the dynamics of a group. For example, the probability of interaction based on density determines how "collectively–conscious" an agent must be, or how much greedy behavior it can get away with.

Density estimation is straight–forward. We define *group density* to be the ratio of the sum of the agents' *footprints* and the size of the available interaction space. An agent's *footprint* is the sphere of its influence. In the spatial domain, an agent's footprint is based on its geometry, its motion constraints, and its sensor range and configuration. The size of the interaction space is the area of the physical space the agents can inhabit. The same idea applies in more abstract domains as well. In many such domains the interaction space is time, and the agent's footprint is the duration of information exchange. For instance, in a telecommunications domain density can be estimated from the duration of all calls within a unit of time. Highway traffic is another example in which the relevant space of interactions is time. The agent density can be represented by the ratio of the sum of the agents' footprints and the total surface area of the road.

The density metric allows for computing how much interaction space is necessary for a group to perform any task, and whether a specific amount of interaction space is sufficient. In the spatial domain, for example, using the number and size of the agents is enough to compute the *mean free path* of an agent and use it to estimate how many collisions are expected between agents executing random walks. Similarly, for the telecommunications domain the average uninterrupted call duration relative to the average number of calls per unit time can be computed, which gives an estimate of how much "phone interaction space" is available for the given parameters. Finally, for the highway domain the same computation yields the average length of "free" speeding[8].

Such an approximate measure of density can then be used to estimate how much interaction space, on average, is required for the system, even before the specifics of the task are considered. By bringing the constraints of the task into the computation, the expected interference over the duration of the task can also be estimated. For most tasks, interference will vary depending on the fluctuations of the density over the lifetime of the task. This temporal density distribution demonstrates which parts of the task require social rules. Although the exact computation of relevant density is dependent on the particular domain and task, a rough approximation provides useful metrics for estimating the dynamics of the group and the evolution of behavior of the system as a whole.

2.5 Summary

Among other things, this chapter has described the constraints that were imposed on the agents in order to structure and focus our study of group behavior. This work in the thesis is focused on homogeneous agents using no explicit world models, undirected communication, and implicit cooperation. All of these constraints were chosen in order to approach the group behavior problem bottom–up and incremen-

[8]This model does not include stationary police cars.

tally. This work is concerned with testing the limits of minimal internal modeling and communication in order to find when such simple abilities are sufficient and when more complex representation and communication abilities are necessary.

Chapter 3

Related Work

3.1 Robotics and Behavior Control

This thesis focuses on the problems involved in synthesizing and analyzing intelligent group behavior. In particular, the work described here applies to agents that are embodied and situated in physically constrained worlds, inhabited by other agents of the same kind, and dealing with multiple goals ranging from basic survival to accomplishing one or more tasks. The experimental environments in which the work was validated used mobile robots and multi–agent simulations.

Consequently, this work is related to a number of lines of research within and outside of AI, including mobile robotics, intelligent control, simulations of multi–agent systems, distributed artificial intelligence, artificial life, machine learning, ethology, and cognitive science. This section presents an overview of the work in these related fields, with the exception of machine learning, which is covered in the second part of the thesis.

3.1.1 Control of Multiple Physical Robots

The last decade has witnessed a shift in the emphasis of robotics in general and mobile robotics in particular toward physical implementations. Most of the work in robotics so far has focused on control of a single agent. The following is the majority of projects that have dealt with control of multiple physical robots. Fukuda, Nadagawa, Kawauchi & Buss (1989) and subsequent work describe an approach to coordinating multiple homogeneous and heterogeneous mobile robotic units, and demonstrate it on a docking task. Caloud, Choi, Latombe, LePape & Yim (1990), Noreils (1992) and Noreils (1993) remain faithful to the state–based framework, and apply a traditional planner–based control architecture to a box–moving task implemented with two robots in a master–slave configuration. Kube (1992) and Kube & Zhang (1992) describe a series of simulations of robots performing a collection of simple behaviors that are being incrementally transferred to physical robots. Barman, Kingdon, Mackworth, Pai, Sahota, Wilkinson & Zhang (1993) report on a preliminary testbed for studying control of multiple robots in a soccer–playing task. Parker (1993*b*) and Parker (1994) describes a behavior–based task–sharing architecture for controlling groups of heterogeneous robots, and demonstrates it on a set of physical robots performing toxic waste cleanup and box pushing. Donald, Jennings & Rus (1993) report on the theoretical grounding for implementing a cooperative manipulation task with a pair of mobile robots. Perhaps closest in philosophy as well as the choice of task is work by Altenburg (1994) and Beckers, Holland & Deneubourg (1994). Altenburg (1994) describes a variant of the foraging task using a group of LEGO robots controlled in reactive, distributed style. Beckers et al. (1994) demonstrate a group of four robots clustering initially randomly distributed pucks into a single cluster through purely stigmergic communication.

In terms of cooperation and communication, most of the above work has fallen along the two ends of the spectrum: it either uses extensive explicit communication and cooperation, or almost none at all. In systems that are cooperative by design, the two or more robots are aware of each other's existence, and can sense and recognize each other directly or through communication. This type of research explores explicit cooperation, usually through the use of directed communication and is represented by Caloud et al. (1990), Noreils (1992), and Parker (1993*a*).

The other category includes work on implicit cooperation, in which the robots usually do not recognize each other but merely coexist and indirectly cooperate by having identical or at least compatible goals. Such work includes Dallas (1990) and Kube (1992). The work described in this thesis falls nearer this end of the spectrum, but is focused on agents that can discriminate each other from the rest of the world, and use this ability as a basis for social behavior.

3.1.2 Simulations of Multiple Agents

With the exception of the work described above, the problem of multi–agent control has been treated mostly in simulation and under two major categories: simulations of situated systems and simulations of abstract agents.

Simulations of situated systems involve some degree of faithfulness to the physical world, at least to the extent of employing simple models of sensors, effectors, and physical laws. A number of simulations of behavior–style controlled systems have been implemented. For instance, Steels (1989) describes a simulation of simple robots using the principles of self–organization to perform a gathering task. Brooks, Maes, Matarić & Moore (1990) report on a set of simulations in a similar task domain, with a fully decentralized collection of non–communicating robots. Arkin (1992) describes a schema–based approach to designing simple navigation behaviors, used for programming multiple agents working in a simulated environment with future extensions to physical agents; Arkin, Balch & Nitz (1993) apply the approach to a multi–agent retrieval task. Brock, Montana & Ceranowicz (1992) describe SIMNET simulations of large numbers of tank–like robots performing avoidance and formation following. Kube, Zhang & Wang (1993) propose a behavior–arbitration scheme that will be tested on physical robots. Simulations tend to simplify both sensing and actuation. Physically–based simulations, however, using realistic physics models of the agent, allow for generating and testing more realistic behavior. For example, Hodgins & Brogan (1994) describe experiments with fully physically–based simulations of groups of hopping robots.

In contrast to simulations of multiple robots, "swarm intelligence" refers to simulations of abstract agents dealing with more theoretical problems of communication protocols, the design of social rules, and strategies for avoiding conflict and deadlock often in societies with with large numbers of simple agents. Representative work includes Fukuda, Sekiyama, Ueyama & Arai (1993), Dario & Rucci (1993), Dudek et al. (1993), Huang & Beni (1993), Sandini, Lucarini & Varoli (1993), Kurosu, Furuya & Soeda (1993), Beni & Hackwood (1992), Dario, Ribechini, Genovese & Sandini (1991), and many others. This work is also related to DAI (see below) but in contrast to DAI it deals with agents of comparatively low cognitive complexity.

3.2 Artificial Life

The field of Artificial Life (Alife) focuses on bottom–up modeling of various complex systems. Alife work relevant to this thesis features simulations of colonies of ant–like agents, as described by Corbara, Drogoul, Fresneau & Lalande (1993), Colorni, Dorigo & Maniezzo (1992), Drogous, Ferber, Corbara & Fresneau (1992), Travers (1988), and many others. Deneubourg et al. (1990), Deneubourg & Goss (1989), and Deneubourg, Goss, Pasteels, Fresneau & Lachaud (1987) have experimented with real and simulated ant colonies and examined the role of simple control rules and limited communication in producing trail formation and task sharing. Deneubourg, Theraulax & Beckers (1992) define some key terms in swarm intelligence and discuss issues of relating local and global behavior of a distributed system. Assad & Packard (1992), Hogeweg & Hesper (1985) and other related work also report on a variety of simulations of simple organisms producing complex behaviors emerging from simple interactions. Schmieder (1993) reports on an experiment in which the amount of "knowledge" agents have about each other is increased and decreased based on local encounters. Werner & Dyer (1990) and MacLennan (1990) describe systems that evolve simple communication strategies. On the more theoretical end, Keshet (1993) describes a model of trail formation that fits biological data.

Work in Artificial Life is related to the work in this thesis in that both are concerned with exploiting the dynamics of local interactions between agents and the world in order to create complex global behaviors. However, work in Alife does not usually concern itself with agents situated in physically realistic worlds. Additionally, it usually deals with much larger populations sizes that the work presented here. Finally, it most commonly employs genetic techniques for evolving the agents' comparatively simple control systems.

3.3 Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) is another field that deals with multi–agent interactions (see Gasser & Huhns (1989) for an overview). DAI focuses on negotiation and coordination of multi–agent environments in which agents can vary from knowledge–based systems to sorting algorithms, and approaches can vary from heuristic search to decision theory. In general, DAI deals with cognitively complex agents compared to those considered by the research areas described so far. However, the types of environments it deals with are relatively simple and low complexity in that they feature no noise or uncertainty and can be accurately characterized.

DAI can be divided into two subfields: Distributed Problem Solving (DPS) and Multi–Agent Systems (MAS) (Rosenschein 1993). DPS deals with centrally designed systems solving global problems and using built–in cooperation strategies. In contrast, MAS work deals with heterogeneous, not necessarily centrally designed agents faced with the goal of utility–maximizing coexistence.

Decker & Lesser (1993*a*) is a good example of DPS work. It addresses the task of fast coordination and reorganization of agents on a distributed sensor network with the goal of increasing system performance and decreasing performance variance. Hogg & Williams (1993) is another good example showing how parallel search performs better with distributed cooperative agents than with independent agents.

Examples of MAS work include Ephrati (1992), which describes a master–slave scenario between two agents with essentially the same goals. Miceli & Cesta (1993) describe an approach to using an estimate of the usefulness of social interactions at the individual agent level in order for agents to select what other agents to interact with. This decision is based on an estimate of possible future payoff in terms of help given the agents' attitudes and skills. Unfortunately, the estimation of dependence relations scales poorly with the size of the group, and as is the case of most DAI work, is best suited for a small number of highly deliberative, non–situated knowledge–based agents. Along similar lines, Kraus (1993) describes negotiations and contracts between selfish agents. Durfee, Lee & Gmytrasiewicz (1993) discuss game–theoretic and AI approaches to deals among rational agents. The paper describes the advantages of introducing meta–level information.

Certain aspects of DAI work are purely theoretical and deal with the difficulty of multi–agent planning and control in abstract environments. For example, Shoham & Tennenholtz (1992) discuss the complexity of automatically deriving social laws for agent groups. They show that the problem is NP–complete but can, under a number of restrictions, be made polynomial.

Some DAI work draws heavily from mathematical results in the field of parallel distributed systems. In particular, Huberman (1990) describes the effects of information exchange on the performance of a collection of agents applied to a class of search problems. He also addresses the ubiquity of log–normal distributions of performance found across different domains, and hypothesizes a universal law of distribution for all large systems of interdependent agents using resources allocated based on perceived progress. Clearwater, Huberman & Hogg (1991) present related work on cooperative strategies for solving constraint satisfaction problems.

DAI and Alife merge in the experimental mathematics field that studies computational ecosystems, simulations of populations of agents with well defined interactions. The research is focused on global effects and the changes in the system as a whole over time. This process of global changes is usually referred to "co–evolution" (Kephart, Hogg & Huberman 1990). Often the systems studied have some similarities to the global effects found in biological ecosystems, but the complex details of biological systems cannot be reasonably addressed. Co–evolution experiments are used to find improved search-based optimization techniques. For example, Hillis (1990) demonstrates how co–evolution can be used to overcome local maxima in evolving optimal sorting algorithms.

3.4 Behavior Analysis

Previous section have described related work in synthesis and control of group behavior. This section reviews related work in analysis of group behavior.

As described earlier, Distributed Artificial Intelligence (DAI) deals with multi–agent negotiations and coordination in a variety of abstract environments. Decker &

Lesser (1993*b*) is an example of a DAI approach to modeling a distributed system. It depends on the ability to specify the agents' beliefs, intentions, and their quality and duration. These types of models do not scale well with the group size. Further, in order to apply at all they need to abstract away the low–level properties of the system, such as the exact noise and errors, which have been shown to critically effect the high–level behavior (Weisbuch 1991, Wiggins 1990).

Similarly, Kosoresow (1993) describes a probabilistic method for agent coordination based on Markov processes. This method relies on specifying agents inference mechanisms (as chains), and having agents with compatible and specifiable goals and preferences. This type of approach applies to domains where the problem of resource allocation can be clearly specified. However, the ability to predict agents' behavior in order to assess the resource allocation problem is extremely difficulty in physical system with noise and uncertainty. If it were not, a number of mathematical and game–theoretic paradigms would apply.

The classical robotics field of motion–planning has dealt with the problem of planning for multiple objects. For example, Erdmann & Lozano-Pérez (1987) describe theoretical results on the motion–planning problem for multiple polygonal moving objects. The presented solution searches the two–dimensional representation of space–time slices to find a safe path. These results depend on having only one object move at a time, a constraint that cannot be easily enforced in situated systems. Furthermore, the proposed strategy is too computationally intensive to be applied for real–time control.

Donald et al. (1993) discuss motion–planning algorithms for coordinated manipulation with different numbers of agents and different amounts of *a priori* knowledge about the object to be moved. The theoretical aspect of the work focuses on computing the information requirements for performing particular robot tasks. The work is directly applicable to manipulation tasks, such as box–pushing, that can be addressed with one or more robots as cooperating "force–appliers." In contrast, our work does not focus on algorithms for explicit cooperation on tasks such as object manipulation, but instead on distributed solutions to problems that do not necessitate cooperation but can benefit from it.

Strategies for proving distributed algorithm correctness are tangentially related to analyzing multi–agent behavior. Lynch & Tuttle (1987), for example, describe such methods for distributed systems with hierarchical components. More closely related is work by Lynch (1993) that uses a simulation method for reasoning about real–time systems modeled as general automata. The work is targeted at proving properties of message–passing protocols, most of which are more constrained and less uncertain than communication among distributed physical agents.

Work on stochastic analysis of qualitative dynamics, such as that by Doyle & Sacks (1989), is appealing for its qualitative nature. However, the proofs depend on the ability to represent the system as a series of transitions in a graph and the system's dynamics as a Markov chain over that graph. The difficulty lies in establishing such a model for a multi–agent system. It is in general difficult to obtain the values for the transition probabilities that capture the complex dynamics of such systems. Simpler models can be constructed but fail to contain enough detail to conserve the dynamics.

Related work on analysis of group behavior has been conducted in branches of biology. For example, Belić, Skarka, Deneubourg & Lax (1986) present a model for honeycomb constructions based on partial differential equations describing the bee density distribution in the hive and their wax distribution behavior. Less structured group behavior, such as exploration and foraging, has also been addressed. For instance, Benhamou & Bovet (1990) describe a probabilistic model for foraging. The work closest to the domains addressed in this thesis is done by Deneubourg et al. (1986), Deneubourg et al. (1987), Calenbuhr & Deneubourg (1992), etc. The authors propose strategies for describing and analyzing various collective behaviors in ants. Their work is closest in nature to the kind of analysis we propose as viable for describing group behavior of situated, embodied agents. In both cases the analysis is performed at the level of the collective rather than the individual.

Similarly, Miramontes, Sole & Goodwin (1993) present a framework for describing ant behavior as individually chaotic but collectively stable and periodic. Spatial distributions of activity display similar symmetries. Brown & McBurnett (1993) describe a model of a simple political voting system which displays a large array of group behaviors based on simple local feedback (i.e., recruitment or persuasion) mechanisms. The system has two stable states: a homogeneous distribution and a collection of invariant blocks. Intuitively, this is an analogy of an equal power distribution, in which any imbalance results in a transient instability. Camazine (1993) shows an analogous pattern for honey-comb population, nectar foraging, and brood sorting while DeAngelis, Post & Travis (1986) demonstrate how most aggregation–type behaviors can be shown to fit this pattern.

Another form of common feedback–based behavior involves the synchronization of rhythmic patters of activity. For example, Meier-Koll & Bohl (1993) describe the synchronization of circadian rhythms of in human and animal subjects and models them as a collection of coupled oscillators. Analogous effects are commonly observed in hormonal cycles (Vander, Sherman & Luciano 1980). In such systems, the synchronized state is a stable behavior, as is the evenly dispersed equal–power state, while all other states are transient. Sismondo (1990) reports on similar synchronization behavior in insect rhythmic signaling and proposes a similar model of the behavior.

3.5 Summary

The work in this thesis shares motivations and goals with a number of related fields, including AI, robotics, DAI, Alife, and ethology. This chapter reviewed the most related lines of research from each of these fields in preparation for the next chapter which describes, in detail, the proposed approach.

Chapter 4

The Basic Behavior Approach

One of the hardest problems in AI is finding the right level of system description for effective control, learning, modeling, and analysis. This thesis proposes a particular description level, instantiated in so-called **basic behaviors**, building blocks for synthesizing and analyzing complex group behavior in multi–agent systems.

Biology provides evidence in support of basic behavior units at a variety of levels. A particularly clean and compelling case can be found in motor control. Controlling a multi–joint manipulator such as a frog leg or a human arm is a complex task, especially if performed at a low level. In order to cut down the complexity, nature imposes an abstraction. Mussa-Ivaldi & Giszter (1992) show that a relatively small set of basis vector fields, found in the frog's spine, generates the frog's entire motor behavior repertoire by applying appropriate combinations of the basis vectors. Bizzi, Mussa-Ivaldi & Giszter (1991) and Bizzi & Mussa-Ivaldi (1990) discuss control of the human arm with a similar approach. The described motor basic behaviors are a result of the types of constraints: the dynamics of the manipulator and the dynamics of the motor tasks. In the case of motor control, the behaviors are designed for specific optimizations, such as minimizing effort by minimizing jerk, executing straight line trajectories, and using bell–shaped velocity profiles (Atkeson 1989).

Taking the idea from motor control, we define *behaviors* as control laws that encapsulate sets of constraints so as to achieve particular goals. *Basic behaviors* are defined as a minimal set of such behaviors, with appropriate compositional properties, that takes advantage of the dynamics of the given system to effectively accomplish its repertoire of tasks.

Basic behaviors are intended as a tool for describing, specifying, and predicting group behavior. By properly selecting such behaviors one can generate repeatable and predictable group behavior. Furthermore, one can apply simple compositional

Problem	Synthesis and analysis of intelligent group behavior in order to understand the phenomenon (science) and apply it (engineering).
Assertion	Complex group behavior results from local interactions based on simple rules.
Approach	Propose basic behaviors for structuring such simple rules.
Validation	Implement robot group behaviors using a basic behavior set and combinations.

Table 4.1: A summary of the group behavior problem being addressed in the thesis, and the structure of the proposed solution.

operators to generate a large repertoire of higher–level group behaviors from the basic set.

The idea behind basic behaviors is general, but particular sets of such behaviors are domain–specific. In order to demonstrate the methodology, basic behaviors for group interaction in the spatial domain will be derived, combined, analyzed theoretically, and tested empirically. Table 4.1 summarizes the research goals, the approach, and the experimental methodology.

4.1 Selecting and Evaluating Basic Behaviors

This chapter describes how **basic behaviors** are selected, specified, implemented, and evaluated. The idea of basic behaviors is general: they are the intended as primitives for structuring, synthesizing, and analyzing system behavior, as building blocks for control, planning, and learning. Basic behaviors are related to dynamic attractors, equilibrium states, and various other terms used to describe stable, repeatable, and primitive behaviors of any system. This work is concerned with finding ways of identifying such behaviors for a specific system, and using them to structure the rest of the system's behavioral repertoire. The power of basic behaviors lies in their individual reliability and in their compositional properties.

This work focuses on basic behaviors for generating intelligent group interactions in multi–agent systems. It is based on the belief that global behavior of such systems results from local interactions, and furthermore, that those interactions are largely governed by simple rules. Basic behaviors present a mechanism for structuring the space of possible local rules into a small basis set.

This chapter will illustrate the process of selecting basic behaviors on concrete examples of behaviors for a group of agents interacting in physical space. The process of identifying the basic behaviors, formally specifying them, implementing them, testing their properties both theoretically and empirically, and finally combining them, will be carried out. The criteria for selecting basic behaviors for the domain of spatially interacting agents are described first.

4.1.1 Criteria for Selection

We propose that, for a given domain, a small set of basis or *basic behaviors* can be selected, from which other complex relevant and desirable group behaviors can be generated. Basic behavior sets should meet the following criteria:

Necessity: A behavior within a basic behavior set is necessary if it achieves a goal required for the agent's accomplishment of its task(s), and that goal cannot be achieved with any of the other basic behaviors or their combinations. Thus, a basic behavior cannot be implemented in terms of other behaviors and cannot be reduced to them.

Sufficiency: A basic behavior set is sufficient for accomplishing a set of tasks in a given domain if no other behaviors are necessary. The basic behavior set should, under the combination operators, generate all of the desirable higher–level group behaviors.

If such behaviors are designed by hand, as opposed to being observed in an existing system, they should, in addition to the above criteria, also have the following properties:

1. *Simplicity:* the behavior should be implemented as simply as possible,

2. *Locality:* within our framework, the behavior should be generated by local rules, utilizing locally available sensory information,

3. *Correctness:* within the model in which it is tested, the behavior should provably attain (and in some cases maintain) the goal for which it was intended within the set of conditions for which it is designed,

4. *Stability:* the behavior should not be sensitive to perturbations in external conditions for which it is designed,

5. *Repeatability:* the behavior should perform according to specification in each trial under reasonable conditions and error margins,

6. *Robustness:* the performance of the behavior should not degrade significantly in the presence of specified bounds of sensory and effector error and noise,

7. *Scalability:* the behavior should scale well with increased and decreased group size.

It is difficult to imagine any fixed metric for selecting an "optimal" set of behaviors, since the choice of the basic behavior set depends on the task(s) it will be applied to. This work makes no attempt to devise optimality criteria in any formal sense. Furthermore, this work does not provide theoretical proofs of correctness of the algorithms for the presented behaviors. While such proofs may be computable for a simple model of the agents and the environment, they become prohibitively difficult for increasingly more realistic models that include sensors, effectors, and dynamics. As an alternative to simplified modeled environments, the behaviors were tested in the fully complex worlds with all of the error, noise, and uncertainly. In order to make the evaluation more complete, various initial conditions and group sizes were tested, and a large amount of data were obtained for analysis. Behavior evaluation is described in detail in section 4.5.

The next section illustrates the process of selecting basic behaviors for the domain of planar mobile agents.

4.1.2 Basic Behaviors for Movement in the Plane

The experimental work in this thesis is focused on interactions among mobile agents in two–dimensional space. This domain has the desired complexity properties: the number of possible collective behaviors is unbounded. Fortunately, the unbounded space of possible spatial and temporal patterns can be classified into classes, and thus effectively viewed from a lower level of resolution. The classification is based on task and domain–specific criteria which allow for selecting out the (comparatively) few relevant behavior classes to focus on. The proposed basic behaviors impose such classes; they define observable group behaviors without specifying particular rules for implementing them.

Group behaviors in the spatial domain can be viewed as spatio–temporal patterns of agents' activity. Certain purely spatial fixed organizations of agents are relevant, as are certain spatio–temporal patterns. Purely spatial fixed organizations of agents correspond to goals of attainment while spatio–temporal patterns correspond to goals of maintenance.

Safe–Wandering	the ability of a group of agents to move around while avoiding collisions with each other and other obstacles. Here, the homogeneous nature of the agents can be used for inter–agent collision avoidance. Thus, two distinct strategies can be devised; one for avoiding collisions with other agents of the same kind, and another for avoiding collisions everything else.
Following	the ability of two or more agents to move while staying one behind the other.
Dispersion	the ability of a group of agents to spread out over an area in order to establish and maintain some predetermined minimum separation.
Aggregation	the ability of a group of agents to gather in order to establish and maintain some predetermined maximum separation.
Homing	the ability to reach a goal region or location.

Table 4.2: A basic behavior set for the spatial domain, intended to cover a variety of spatial interactions and tasks for a group of mobile agents.

In the process of selecting basic behaviors, the designer attempts to decide what behavior set will suffice for a large repertoire of goals. While the dynamical properties of the system provide bottom–up constraints, the goals provide top–down structure. Both of these influences guide the behavior selection process. Energy minimization is a universal goal of powered physical systems. In the planar motion domain this goal translates into minimization of non–goal–driven motion. Such motion is either generated by poor behavior design, or by interference between agents. Thus, minimizing interference means maximizing goal–driven behavior and minimizing unnecessary motion.

Minimizing interference translates directly into the achievement goal of immediate avoidance and the maintenance goal of moving about without collisions. Avoidance in groups can be achieved by dispersion, a behavior that reduces interference locally. It can also serve to minimize interference in classes of tasks that require even space coverage, such as those involving searching and exploration.

In contrast to various goals that minimize interaction by decreasing physical proximity, many goals involve the exchange of resources through physical proximity. Consequently, aggregation is a useful primitive. Moving in a group requires some form of coordinated motion in order to minimize interference. Following and flocking are examples of such structured group motion.

Table 4.2 shows a list of behaviors that constitutes a basic set for a flexible repertoire of spatial group interactions. Biology offers numerous justifications for these behaviors. Avoidance and wandering are survival instincts so ubiquitous it obviates discussion. Following, often innate, is seen in numerous species (McFarland 1985). Dispersion is commonplace as well. DeScnutter & Nuyts (1993) show elegant evidence of gulls aggregating by dynamically rearranging their positions in a field to maintain a fixed distance from each other. Camazine (1993) demonstrates similar gull behavior on a ledge. People maintain similar arrangements in enclosed spaces (Gleitman 1981). Similarly, Floreano (1993) demonstrates that simulated evolved ants use dispersion consistently. Aggregation, as a protective and resource–pooling and sharing behavior, is found in species ranging from the slime mold (Kessin & Campagne 1992) to social animals (McFarland 1987). The combination of *dispersion* and *aggregation* is an effective tool for regulating density. Density regulation is a ubiquitous and generically useful behavior. For instance, army ants regulate the temperature of their bivouac by aggregating and dispersing according to the local temperature gradient (Franks 1989). Temperature regulation is just one of the many side–effects of density regulation. Finally, homing is a basis of all navigation and is manifested by all mobile species (for biological data on pigeons, bees, rats, ants, salmon, and many others see Gould (1987), Muller & Wehner (1988), Waterman (1989), Foster, Castro & McNaughton (1989), and Matarić (1990*b*)).

Besides the described behavior set, numerous other useful group behaviors exist. For example, biology also suggest surrounding and herding as frequent patterns of group movement, related to a higher level achievement goal, such as capture or migration (McFarland 1987). These and other behaviors can be generated by combining the basic primitives, as will be described and demonstrated in the next chapter.

4.2 Basic Behavior Experiments

The remainder of this chapter describes the experimental environments, presents the algorithms for implementing the proposed basic behaviors, and evaluates their performance based on a battery of tests and a collection of criteria.

4.2.1 Experimental Environments

Behavior observation is one of the primary methods for validating theories in synthetic AI projects like the one described in this thesis. In order to have conclusive results, it is necessary to try to separate the effects caused by the particular experimental environment from those intrinsic to the theory being tested. In order to get to the heart of group behavior issues rather than the specific dynamics of the test environment, two different environments were used, and the results from the two were compared. The two environments are the Interaction Modeler, and a collection of physical robots.

Another motivation for using both a physical and a modeled environment is the attempt to isolate any observable inconsistencies in the performance of the same behaviors in the two different environments. In general, it is difficult to determine what features of the real world must be retained in a simulation and what can be abstracted away. By testing systems in the physical world some of the effects that arise as artifacts of simulation can be identified (Brooks 1991*a*). This is the motivation behind using data from physical robots. By the same token, the current state of the art of physical robot environments imposes many constraints and biases on the types of experiments that can be conducted. Consequently, results from any physical environment must also be validated in an alternative setup. Two different robot types were used, in order to eliminate system–specific biases.

Since this work is concerned with basic principles of interaction and group behavior rather than a specific domain, it is especially concerned with effects that are common to both the modeled and the physical worlds.

4.2.2 The Agent Interaction Modeler

The Interaction Modeler (IM) is a simulator which allows for modeling a simplified version of the physics of the world and the agent sensors and dynamics (Figure 4-1). The Modeler and the control software for the agents are written in Lisp. However, for purposes of realism, the modeler is divided into three distinct components: the simulator, the physics modeler, and the agent specification. The simulator executes the agent specifications and moves the agents according to their control algorithms and their sensory readings. The simulator implements the physics of the sensors, but not the physics of the world. The latter are implemented by the physics modeler that checks the positions and motions computed by the simulator against simplified physical laws, and applies corrections. The IM loops between the simulator and the physics modeler.

The main purpose of the Interaction Modeler is to observe and compare phenomena to those obtained on physical robots. However, the Modeler is also useful for preliminary testing of group behaviors which are then implemented on physical robots. Although it is difficult to directly transfer control strategies from simulations

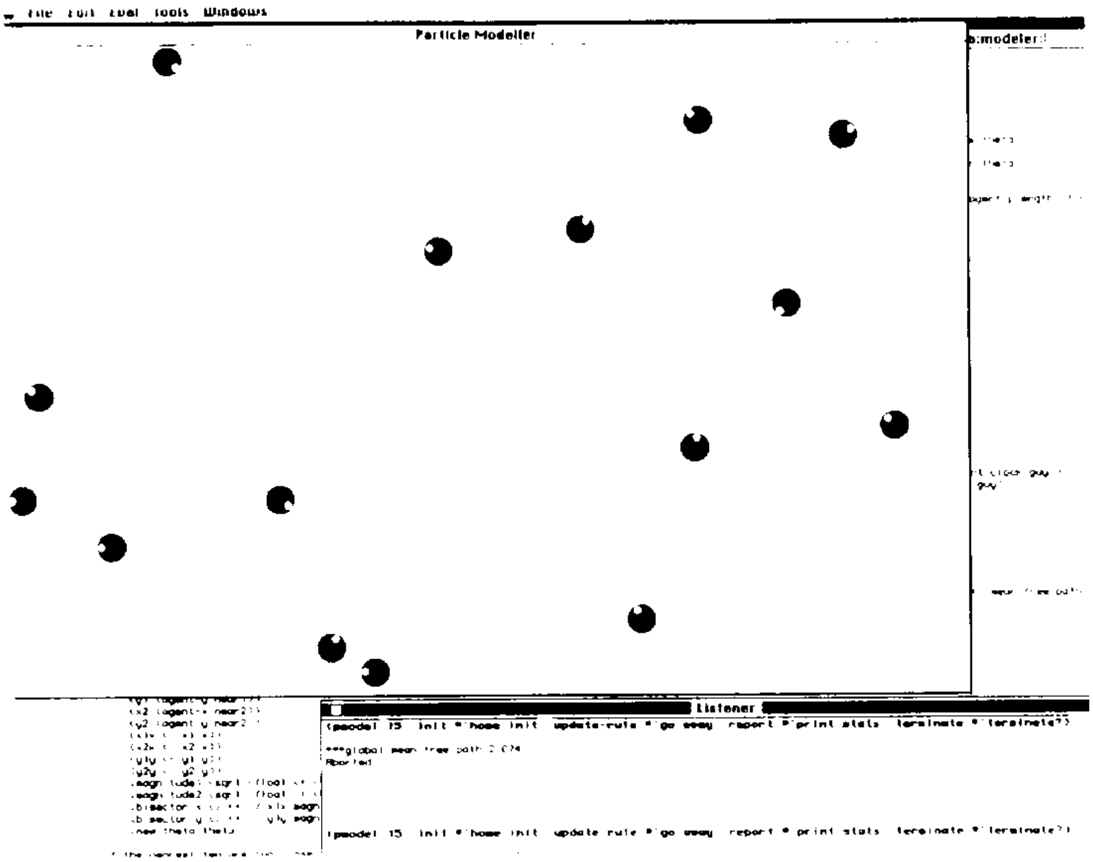


Figure 4-1: The interaction modeler environment. The agents are shown as black circles with white markers indicating their heading. The large rectangle indicates the boundaries of their workspace. The agents are equipped with local sensors and simplified dynamics.

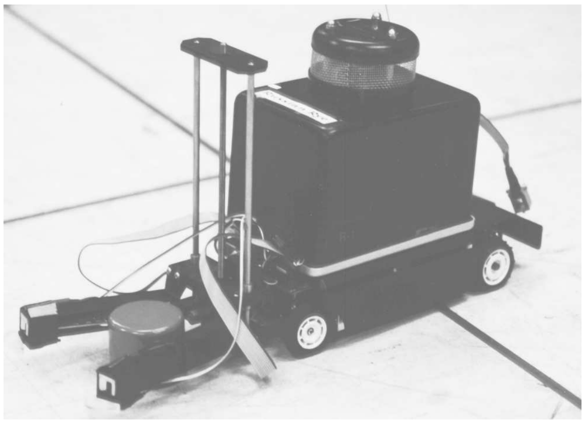


Figure 4-2: Each of the Nerd Herd robots is a 12"–long four–wheeled base equipped with a two–pronged forklift for picking up, carrying, and stacking pucks, and with a radio transmitter and receiver for inter–robot communication and data collection.

to the physical world, the modeler is useful for eliminating infeasible control strategies at an early stage, as well as for testing vastly larger numbers of agents, performing many more experiments, and varying parameter values.

4.2.3 The Mobile Robot Herd

Group behavior experiments are implemented and tested on a collection of 20 physically identical mobile robots affectionately dubbed "The Nerd Herd." Each robot is a 12"–long four–wheeled vehicle, equipped with one piezo–electric bump sensor on each side and two on the rear of the chassis. Each robot has a two–pronged forklift for picking up, carrying, and stacking pucks (Figure 4-2). The forklift contains two contact switches, one on each tip of the fork, six infra–red sensors: two pointing forward and used for detecting objects and aligning onto pucks, two break–beam sensors for detecting a puck within the "jaw" and "throat" of the forklift, and two down–pointing sensors for aligning the fork over a stack of pucks and stacking (Figure 4-3). The pucks are special–purpose light ferrous metal foam-filled disks, 1.5 inches diameter and between 1.5 and 2.0 inches in height. They are sized to fit into the unactuated fork and be held by the fork magnet.

The robots are equipped with radio transceivers for broadcasting up to one byte of data per robot per second. The system uses two radio base stations to triangulate

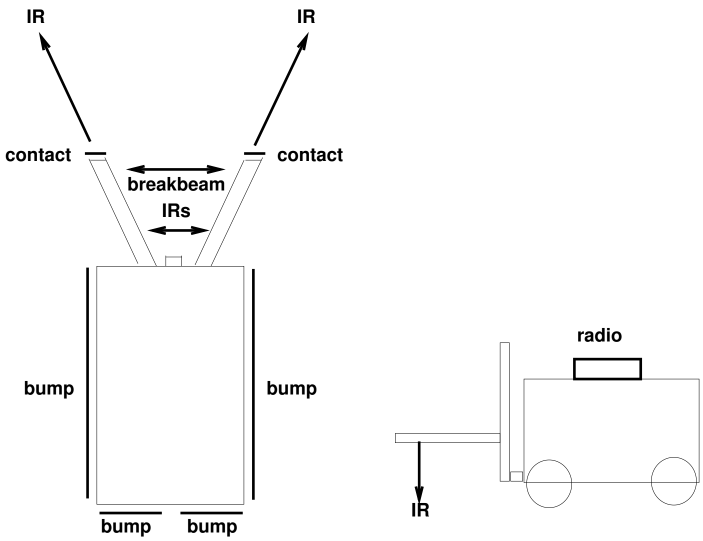


Figure 4-3: Each of the Nerd Herd robots is equipped with contact sensors at the ends of the fork, piezo–electric bump sensors on each side and two on the rear of the chassis, and six infra–red sensors on the fork. Two forward–pointing IRs are located at the ends of the forks, two break–beam IRs in the jaw and throat of the fork, and two down–pointing IR for stacking pucks in the middle of each of the fork arms.

the robots' positions. The radio system is used for data gathering and for simulating additional sensors. In particular, radios are used to distinguish robots from other objects in the environment, an ability that cannot be implemented with the on–board IR sensors[1].

The mechanical, communication, and sensory capabilities of the robots allow for exploration of the environment, robot detection, and finding, picking up, and carrying pucks. These basic abilities are used to construct various experiments in which the robots are run autonomously, with all of the processing and power on board. The processing is performed by a collection of four Motorola 68HC11 microprocessors. Two of the processors are dedicated to handling radio communication, one is used by the operating system, and one is used as the "brain" of the robot, for executing the down–loaded control system used in the experiments. The control systems are programmed in the Behavior Language, a parallel programming language based on the Subsumption Architecture (Brooks 1990*a*).

[1]The IRs are all the same frequency and mechanically positioned for obstacle detection rather than communication.

4.2.4 Hardware Limitations

Properties of physical hardware impose restrictions not only on the control strategies that can be applied, but alson on the types of tasks and experiments that can be implemented. Robot hardware is constrainted by various sensory, mechanical, and computational limitations. This section describes some relevant properties of the hardware we used and their effect.

The robots' mechanical steering system is inaccurate to within 30 rotational degrees. Furthermore, the position triangulation system works sufficiently well when the robots are within the predetermined range of the base stations. However, the exchange of information between the robots, which nominally ought to take place at 1Hz, suffers from extensive loss of data. Consequently, as much as half of the transmitted data were lost or incorrect. The combined effect of steering and positioning uncertainty demanded that the robots move slowly in order to minimize error. Thus, the limiting factor on the robot speed was imposed by sensing and actuation, not by the controller.

The infra–red sensors have a relatively long range (12 inches), and vary in sensitivity. Consequently, not only do different robots have different sensing ranges that cannot be tuned due to hardware restrictions, but the sensitivity between the two sides of the fork on a single robot varies as well. Consequently, the amount of time and effort required for detecting, picking up, or avoiding objects varied across robots and over time. Thus, the control system could not be dependent on uniformity of the group.

This uncertainty and variability, although frustrating, is beneficial to experimental validity. For instance, hardware variability between robots is reflected in their group behavior. Even when programmed with identical software, the robots behave differently due to their varied sensory and actuator properties. Small differences among individuals become amplified as many robots interact over extended time. As in nature, individual variability creates a demand for more robust and adaptive behavior. The variance in mechanics and the resulting behavior provides a stringent test for all of the experimental behaviors.

4.2.5 Experimental Procedure

All robot and modeler programs were archived and all basic behaviors were tested in both domains. All robot implementations of basic and composite behaviors were tested in at least 20 trials[2]. In the case of the Modeler, all behaviors were tested in at least 20 trials, with both identical and random initial conditions. Different strategies for the same group behaviors were tested and compared across the two domains.

Modeler data were gathered by keeping a record of relevant state (such as position, orientation, and gripper state) over time. The same data were gathered in robot experiments through the use of the radio system. The system allowed for recording the robots' position and a few bytes of state over time. For each robot experiment, the robots' IDs and initial positions were recorded. Some of the experiments were conducted with random initial conditions (i.e., random robot positions), while in others identical initial positions were used in order to measure the repeatability of the behaviors. All robot data were also recorded on video tape, for validation and cross referencing.

Throughout this chapter, the Interaction Modeler data are shown in the form of discrete snapshots of the global state of the system at relevant times, including initial state and converged state. The robot data are plotted with the Real Time Viewer (RTV), a special purpose software package designed for recording and analyzing robot data[3]. RTV uses the transmitted radio data to plot, in real–time, the positions of the robots and a time–history of their movements, i.e. a trail, the positions of the previously manipulated pucks, and the position of home. It also allows for replaying the data and thus recreating the robot runs.

The robots are shown as black rectangles aligned in the direction of their heading, with their ID numbers in the back, and white arrows indicating the front. In some experiments robot state is also indicated with a symbol or a bounding box. In all shown data plots, the size of the rectangles representing the robots is scaled so as to maintain the correct ratio of the robot/environment surface area, in order to demonstrate the relative proximity of all active robots. The bottom of each plot shows which of the twenty robots are being run. The corner display shows elapsed time, in seconds, for each snapshot of the experiment. Figure 4-4 shows a typical data plot.

4.3 Basic Behavior Specifications

This section gives formal specifications for each behavior in terms of the goal it achieves and maintains.

Basic behaviors in 2D space are specified in terms of positions p, distances d, and

[2]In the case of foraging, most data were obtained with another set of robots, described in section 8.1.

[3]RTV was implemented and maintained by Matthew Marjanović.

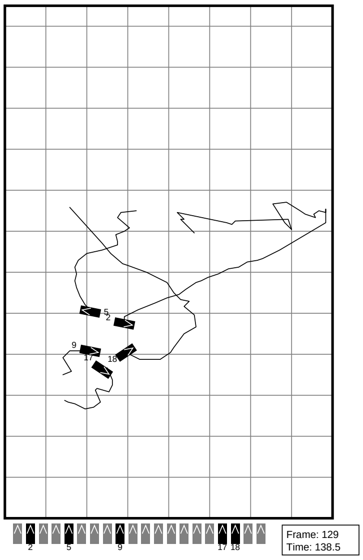


Figure 4-4: An example of a robot data plot: the robots are shown as scaled black rectangles aligned in the direction of their heading, with their ID numbers in the back, and white arrows indicating the front. The bottom of the plot shows which of the twenty robots are being run, and the corner display shows elapsed time in seconds.

distance thresholds δ_{avoid}, $\delta_{disperse}$, and $\delta_{aggregate}$.

$\mathcal{R}$ is the set of robots: $\mathcal{R} = \{R_i\}, \ \ 1 \le i \le n$

$$p_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \qquad p_{home} = \begin{pmatrix} x_{home} \\ y_{home} \end{pmatrix}$$

$$d_{home,i} = \sqrt{(x_{home} - x_i)^2 + (y_{home} - y_i)^2}$$

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Using this notation, the following are specifications for the basic behavior goals.

Safe–Wandering:

The goal of safe–wandering is to keep moving while maintaining a minimum distance δ_{avoid} between agents:

$$\frac{dp_j}{dt} \neq 0 \ \ and \ \ \forall(i) \ \ d_{i,j} > \delta_{avoid}$$

Following:

The goal of following is to achieve and maintain a minimum angle θ between the position of the leader i relative to the follower j:

$$i = leader, \ \ j = follower$$

$$0 \ \ \leq \ \ \frac{dp_j}{dt} \cdot (p_i - p_j) \ \ \leq \ \| \frac{dp_j}{dt} \|\| (p_i - p_j) \| \cos\theta$$

$$\theta = 0 \ \ \Rightarrow \ \ \cos\theta = 0 \ \ \Rightarrow$$

$$0 \ \ \leq \ \ \frac{dp_j}{dt} \cdot (p_i - p_j) \ \ \leq \ \| \frac{dp_j}{dt} \|\| (p_i - p_j) \|$$

Dispersion:

The goal of dispersion is to achieve and maintain a minimum distance $\delta_{disperse}$ between agents:

$$\forall(j) \ \ d_{i,j} > \delta_{disperse} \ \ and \ \ \delta_{disperse} > \delta_{avoid}$$

Aggregation:

The goal of aggregation is to achieve and maintain a maximum distance $\delta_{aggregate}$ between agents:

$$\forall(j) \ \ d_{i,j} < \delta_{aggregate}$$

Homing:

The goal of homing is to decrease the distance between the agent and a goal location called "home":

$$\forall j \;\; \frac{dp_j}{dt} \cdot (p_j - p_{home}) < 0$$

4.4 Basic Behavior Algorithms

This section presents the algorithms used to implement each of the proposed basic behaviors in the Interaction Modeler and on the robots. The algorithms are given in formal notation and in algorithmic pseudo code. All algorithms are formally expressed as velocity commands of the form:

$$command\,(v)$$

Two operators, $\mathcal{N}$ and $\mathcal{C}$, are used for computing most of the algorithms. $\mathcal{N}$ is the **neighborhood operator** which, given a robot R and a distance threshold δ, returns all other robots within that neighborhood:

$$\mathcal{N}(i,\delta) = \{j \in i,..n \;\mid\; d_{i,j} \leq \delta\}$$

C is the **centroid operator** which, given a robot i and a distance threshold δ, returns the local centroid:

$$C(i,\delta) = \frac{\sum_{j \in \mathcal{N}(i,\delta)} p_j}{|\mathcal{N}(i,\delta)|}$$

C_g is the global centroid operator:

$$C_g = \frac{\sum_{j \in \mathcal{R}} p_j}{|n|}$$

4.4.1 Safe–Wandering

Strategies for moving while avoiding collisions are perhaps the most studied topic in mobile robotics. The work in this thesis was concerned with finding avoidance strategies that perform well in group situations and scale well with increased group

```
Avoid-Other-Agents:
If an agent is within d_avoid
   If the nearest agent is on the left
      turn right
      otherwise turn left.
```

Algorithm 4.1:

```
Avoid-Everything-Else:
If an obstacle is within d_avoid
   If an obstacle is on the right only, turn left.

   If an obstacle is on the left only, turn right.
   After 3 consecutive identical turns, backup and turn.

   If an obstacle is on both sides, stop and wait.
   If an obstacle persists on both sides,
      turn randomly and back up.
```

Algorithm 4.2:

sizes. Finding a guaranteed general–purpose collision avoidance strategy for an agent situated in a dynamic world is difficult. In a multi–agent world the problem can become intractable.

Inspired by biological evidence which indicates that insects and animals do not have precise avoidance routines (Wehner 1987), we used the following general avoidance behavior:

$$command\ (\ v \begin{pmatrix} cos(\theta + u) \\ sin(\theta + u) \end{pmatrix}\ )$$

where θ is R's orientation and u is the incremental turning angle away from the obstacle. A simple `Avoid-Other-Agents` rule was devised, as shown in Algorithm 4.1.

The `Avoid-Other-Agents` behavior takes advantage of group homogeneity. Since all agents execute the same strategy, the behavior can rely on and take advantage of the resulting spatial symmetry. If an agent fails to recognize another with its other–agent sensors (in this case radios), it will subsequently detect it with its collision–avoidance sensors (in this case IRs), and treat it as a generic obstacle, using the

```
Safe--Wander:
If an agent is within d_avoid
   If the nearest agent is on the left
      turn right
      otherwise turn left.

If an obstacle is within d_avoid
   If an obstacle is on the right only, turn left.

   If an obstacle is on the left only, turn right.
   After 3 consecutive identical turns, backup and turn.

   If an obstacle is on both sides, stop and wait.
   If an obstacle persists on both sides,
      turn randomly and back up.

Otherwise move forward by d_forward, turn randomly.
```

Algorithm 4.3:

`Avoid-Everything-Else` behavior, as shown in Algorithm 4.2.

A provably correct avoidance strategy for arbitrary configurations of multiple agents is difficult to devise. In order to increase robustness and minimize oscillations, our strategies take advantage of the unavoidable noise and errors in sensing and actuation, which result in naturally stochastic behavior. This stochastic component guarantees that the an avoiding agent will not get stuck in infinite cycles and oscillations. In addition to the implicit stochastic nature of the robots' behavior, `Avoid-Everything-Else` also utilizes an explicit probabilistic strategy by employing a randomized move.

Variations of the above avoidance algorithm were experimented with and compared based on the amount of time the agent spent avoiding relative to the amount of time spent it moving about freely. This ratio is an indirect measure of the quality of the avoiding strategy in that the more time the agents spend avoiding the worse the strategy is. Avoiding time is dependent on the agent density, so it was used as a controlled variable in the experiments. The ratio used to evaluate avoidance is an indirect metric; a direct measure of being stuck would be more useful, but the robots did not have the appropriate sensors for determining this state. No significant performance differences were found among the similar strategies that were tested.

The strategy for *safe–wandering* is the combination of the two avoidance strategies with a default rule for moving with occassional changes of heading, as shown in Algorithm 4.3.

4.4.2 Following

```
Follow:
If an agent is within d_follow
   If an agent is on the right only, turn right.

   If an agent is on the left only, turn left.
```

Algorithm 4.4:

Following is implemented with respect to the follower agent. It is achieved with a simple rule that steers the follower to the position of the leader:

$$command\left(\frac{v_0}{\| p_{leader} - p_{follower} \|} (p_{leader} - p_{follower}) \right)$$

Following can be implemented as a complement of the `Avoid-Everything-Else` behavior, as shown in Algorithm 4.4.

Figure 4-5 illustrates following on three robots. Additional data on following will be presented and analyzed in the next section.

This approach to *following* models tropotaxic behavior in biology, in which two sensory organs are stimulated and the difference between the stimuli determines the motion of the insect (McFarland 1987). Ant osmotropotaxis is based on the differential in pheromone intensity perceived by the left and right antennae (Calenbuhr & Deneubourg 1992), while the agents described here use the binary state of the two directional IR sensors.

Under conditions of sufficient density, safe–wandering and following can produce more complex global behaviors. For instance, osmotropotaxic behavior of ants exhibits emergence of unidirectional lanes, i.e., regions in which all ants move in the same direction. The same lane–forming effect could be demonstrated with robots executing following and avoiding behaviors. However, more complex sensors must be used in order to determine which direction to follow. If using only IRs, the agents cannot distinguish between other agents heading toward and away from them, and are thus unable to select whom to follow.

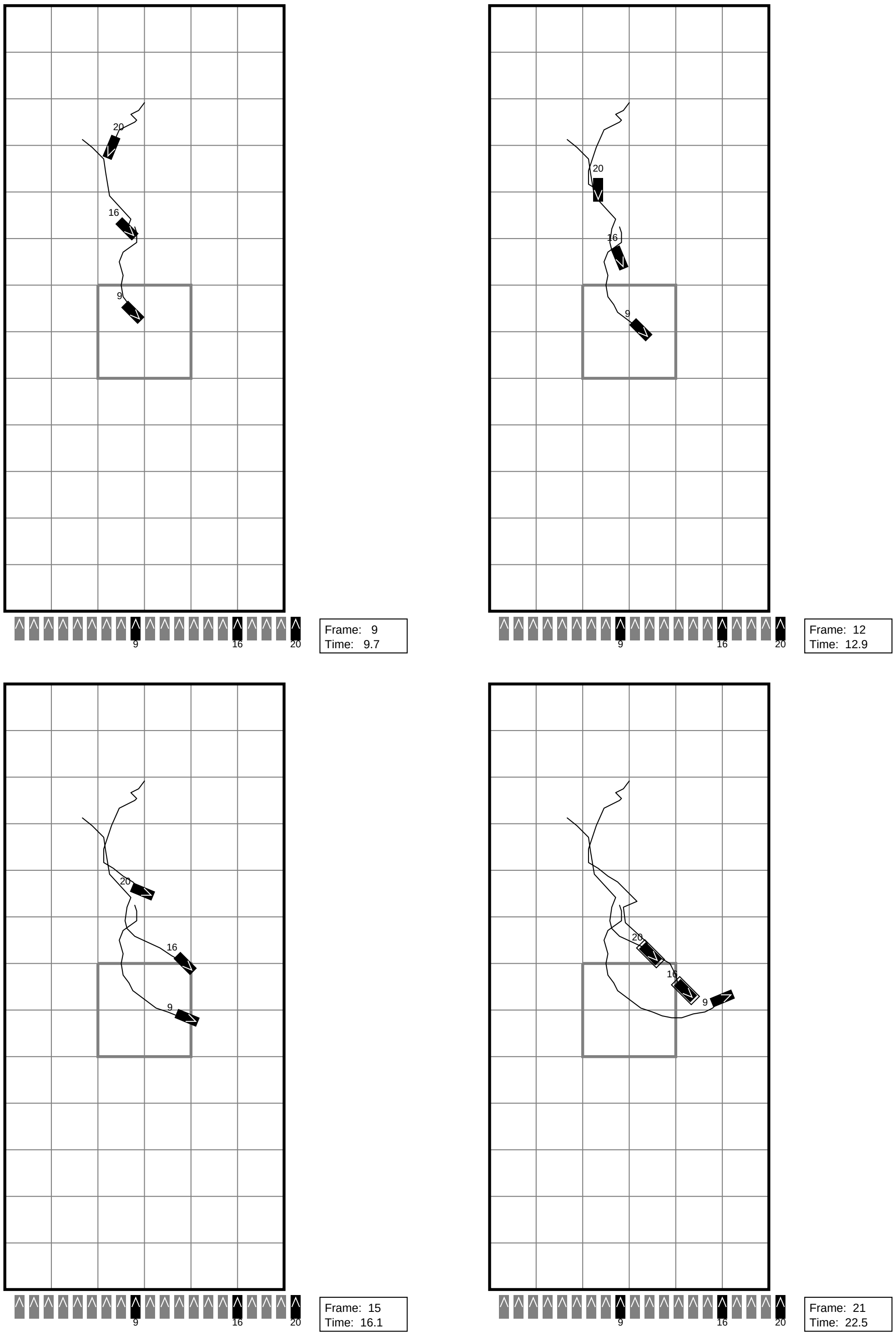


Figure 4-5: An example of following with three robots. Continuous time trails are shown. In spite of deviations in individual paths, the queue is conserved.

```
Centroid-Disperse:
 If one or more agents are within d_disperse
    move away from Centroid_disperse.
```

Algorithm 4.5:

```
Neighbor-Disperse:
Find 2 nearest neighbors within d_disperse
Compute the angle between them,
Compute the negative of the bisector,
align in that direction and go forward.
```

Algorithm 4.6:

4.4.3 Dispersion

A robust *dispersion* behavior can be designed as an extension of the existing safe–wandering. While avoidance in safe–wandering reacts to the presence of a single agent, *dispersion* uses the local distribution of all of the nearby agents (i.e., the locations of other agents within the range of the robot's sensors) in order to decide in which direction to move. The algorithm, shown in Algorithm 4.4, computes the local centroid to determine the local density distribution of nearby agents, and moves away from the highest density:

$$command\ (\ \frac{-v_0}{\parallel \mathcal{C}(i, \delta_{disperse}) - p_i \parallel}\ (\ \mathcal{C}(i, \delta_{disperse}) - p_i)\ )$$

Under conditions of high density, the system can take a long time to achieve a dispersed state since local interactions propagate far and the motion of an individual can disturb the state of many others. Thus, *dispersion* is best viewed as an ongoing process which maintains a desired distance between the agents while they are performing other tasks.

A number of dispersion algorithms were tested in the modeled environment as well. As in the robot implementation, all of the approaches were based on detecting the position of the nearest agents. However, the modeler allowed for using more precise information, such as the exact distance and direction of the nearest neighbors. The dispersion algorithm shown in Algorithm 4.6 was most successful in terms of efficiency and reliability.

Figure 4-6 shows the initial state and the final state of a dispersion experiment

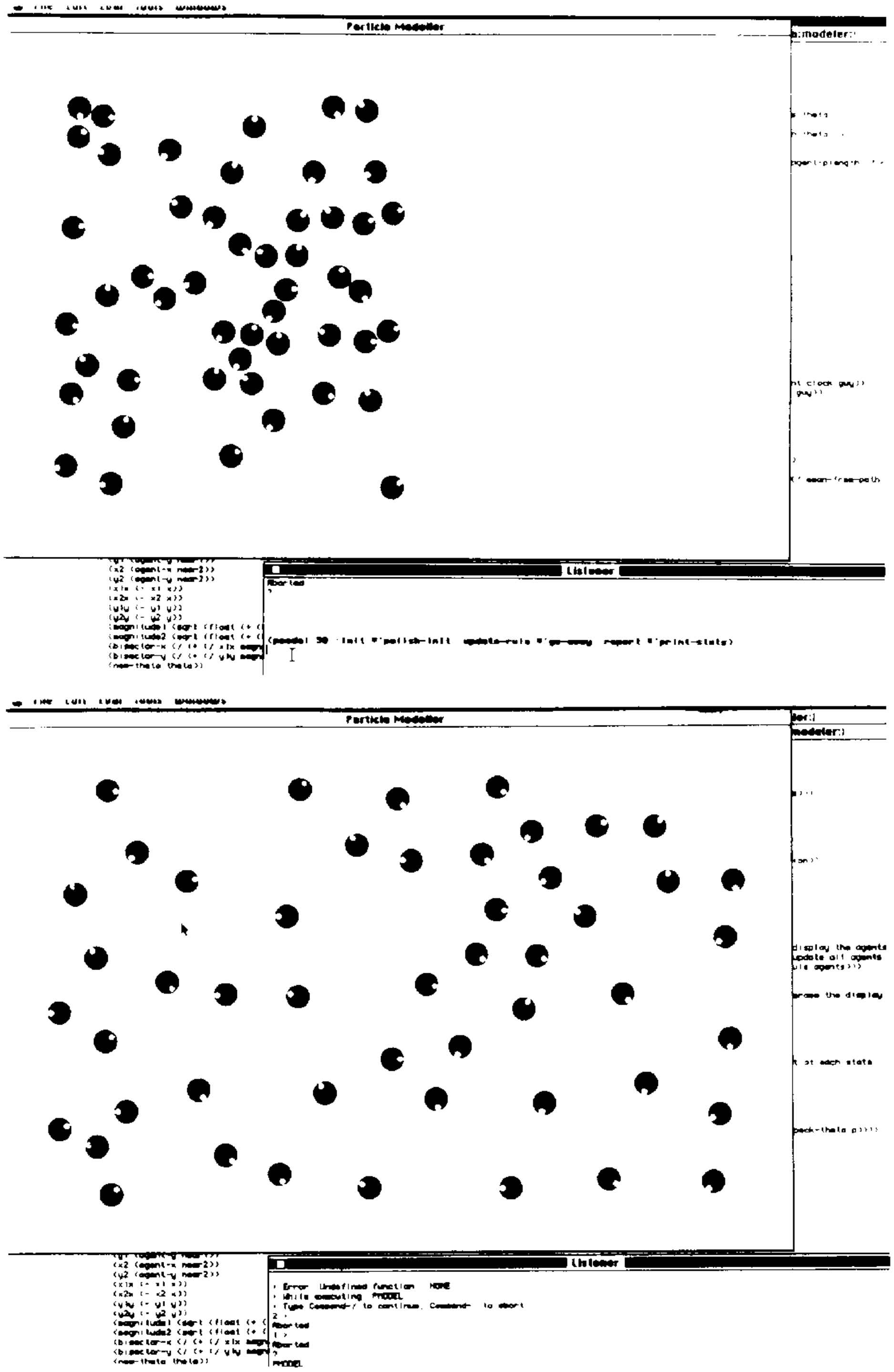


Figure 4-6: An example of *dispersion*. Fifty agents are initially packed in one half of the workspace. `d_dispersion` is set to two times the agent's diameter. After approximately 20 time steps, the equilibrium is reached and all agents stop moving.

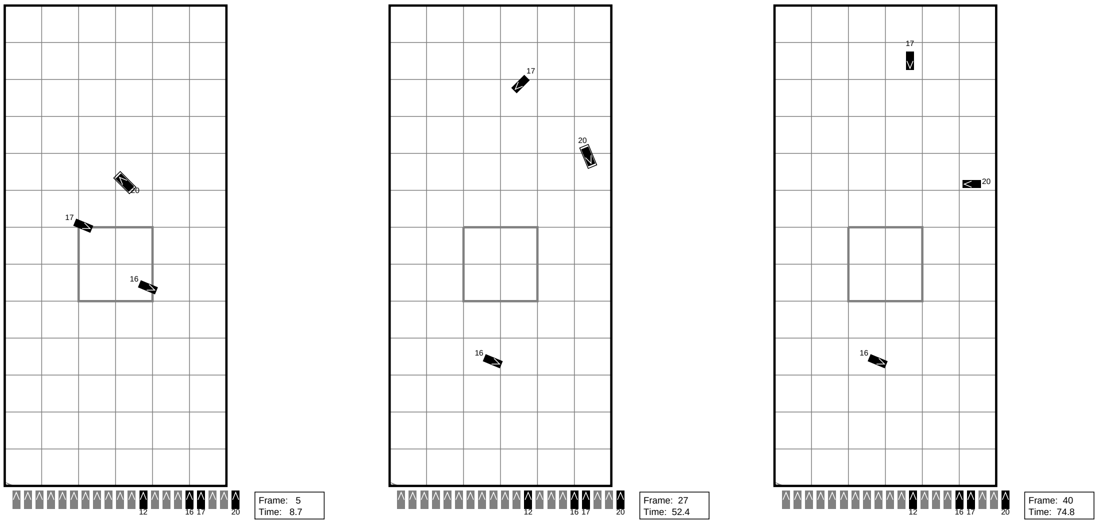


Figure 4-7: *Dispersion* with three robots, initiated close to each other. The robots found a static dispersed equilibrium state after 74 seconds.

using the above centroid–based dispersion rule tested in the Interaction Modeler. Initially crowded in one part of the available free space, the agents apply the simple dispersion rule in order to establish `d_disperse` or the maximum available inter–agent distance. Figure 4-7 shows the same dispersion algorithm applied to four robots.

Dispersion was also evaluated based on time to convergence. Algorithms using the local centroid, and the nearest two agents, were compared to each other and to a potential field summation approach, in which the scalar distance from each nearby agent was proportional to the magnitude of a repulsive vector associated with it. The vectors of all nearby agents were summed and the agent moved in the direction of the resultant. The performance of the three algorithms was compared using two different initial conditions, random and densely packed. Both were tested in order to normalize for different density distributions through the lifespan of the task. As expected, the random initial position results in faster convergence times than a packed initial condition for all three algorithms. No statistically significant difference was found between the algorithms.

4.4.4 Aggregation

Aggregation is the inverse of dispersion:

$$command\ (\ \frac{+v_0}{\parallel \mathcal{C}(i, \delta_{aggregate}) - p_i \parallel}\ (\ \mathcal{C}(i, \delta_{aggregate}) - p_i)\ )$$

```
Aggregate:
If nearest agent is outside d_aggregate
   turn toward the local Centroid_aggregate, go.

Otherwise, stop.
```

Algorithm 4.7:

```
Home:
If at home
  stop.
  otherwise turn toward home, go.
```

Algorithm 4.8:

and can be implemented using the centroid operator as well, as shown in Algorithm 4.7.

Aggregation was evaluated using the same criteria used in evaluating dispersion, as well as the same experiments. Analogous algorithms were implemented, using the local centroid, two nearest neighbors, and potential fields. Instead of varying initial conditions, aggregation algorithms were evaluated using two different terminating conditions. The more difficult terminating condition required that all agents form a single aggregate, whereas the easier of the two conditions required only that they form one or more groups in which all agents are within a fixed distance from their neighbors. As expected, the former terminating condition required more time to be achieved. Aside from that effect, no statistically significant difference was found between the algorithms.

4.4.5 Homing

The simplest homing strategy is a greedy local one:

$$command\left(\frac{v_0}{\| p_{home} - p_i \|} (p_{home} - p_i) \right)$$

and implemented as a simple pursuit, as shown in Algorithm 4.8.

Figure 4-8 illustrates the homing behavior of five robots using this strategy. The data illustrate that the actual trajectories are far from optimal, due to mechanical and sensory limitations, in particular due to the error in the sensed position. The same

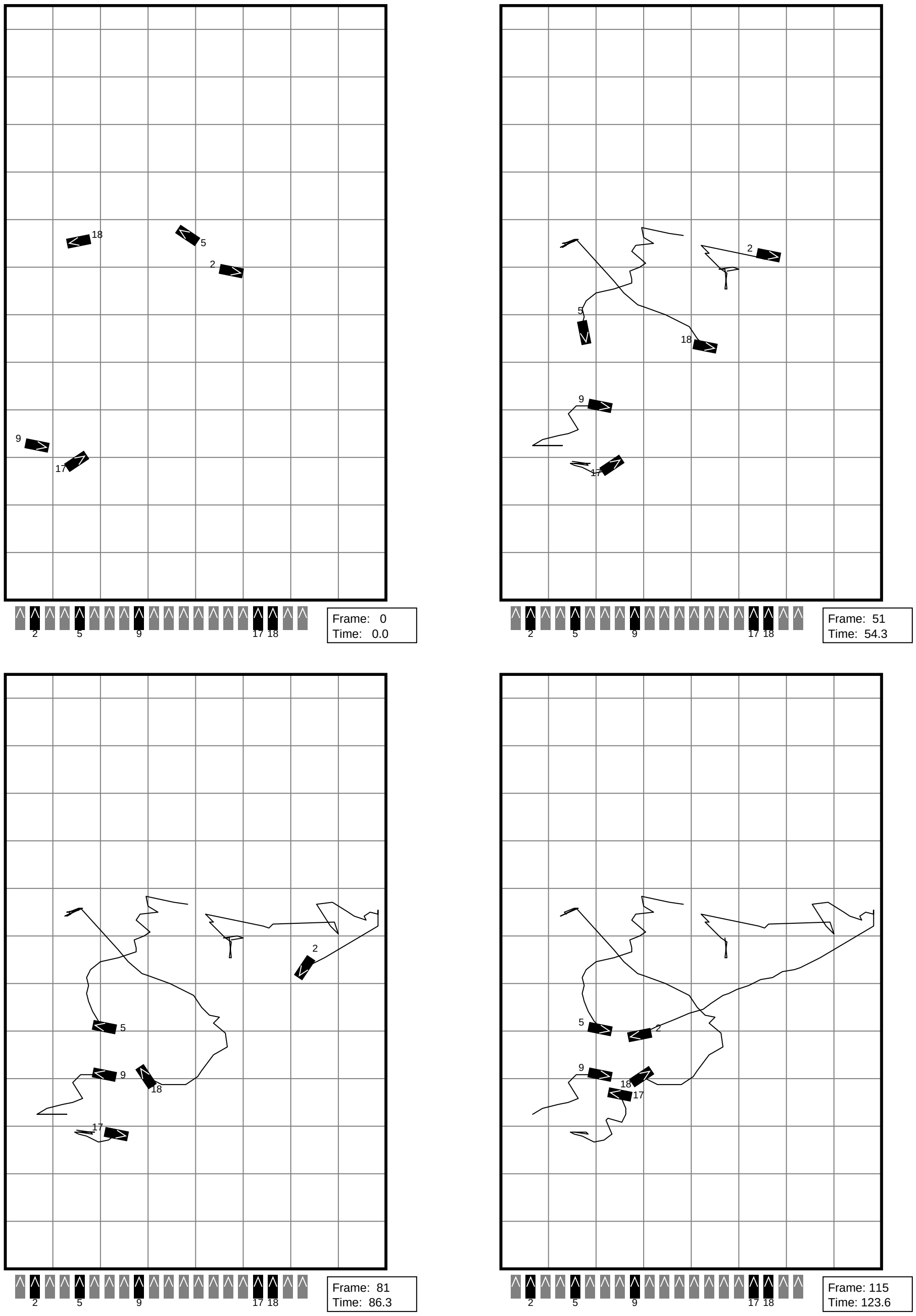


Figure 4-8: *Homing* behavior of five robots. Started in an arbitrary initial configuration, four of the robots reached the home region within 100 seconds, and the fifth joined them 30 seconds later. The trails reflect errors in position sensing, as well as interference between the robots as approach the home region.

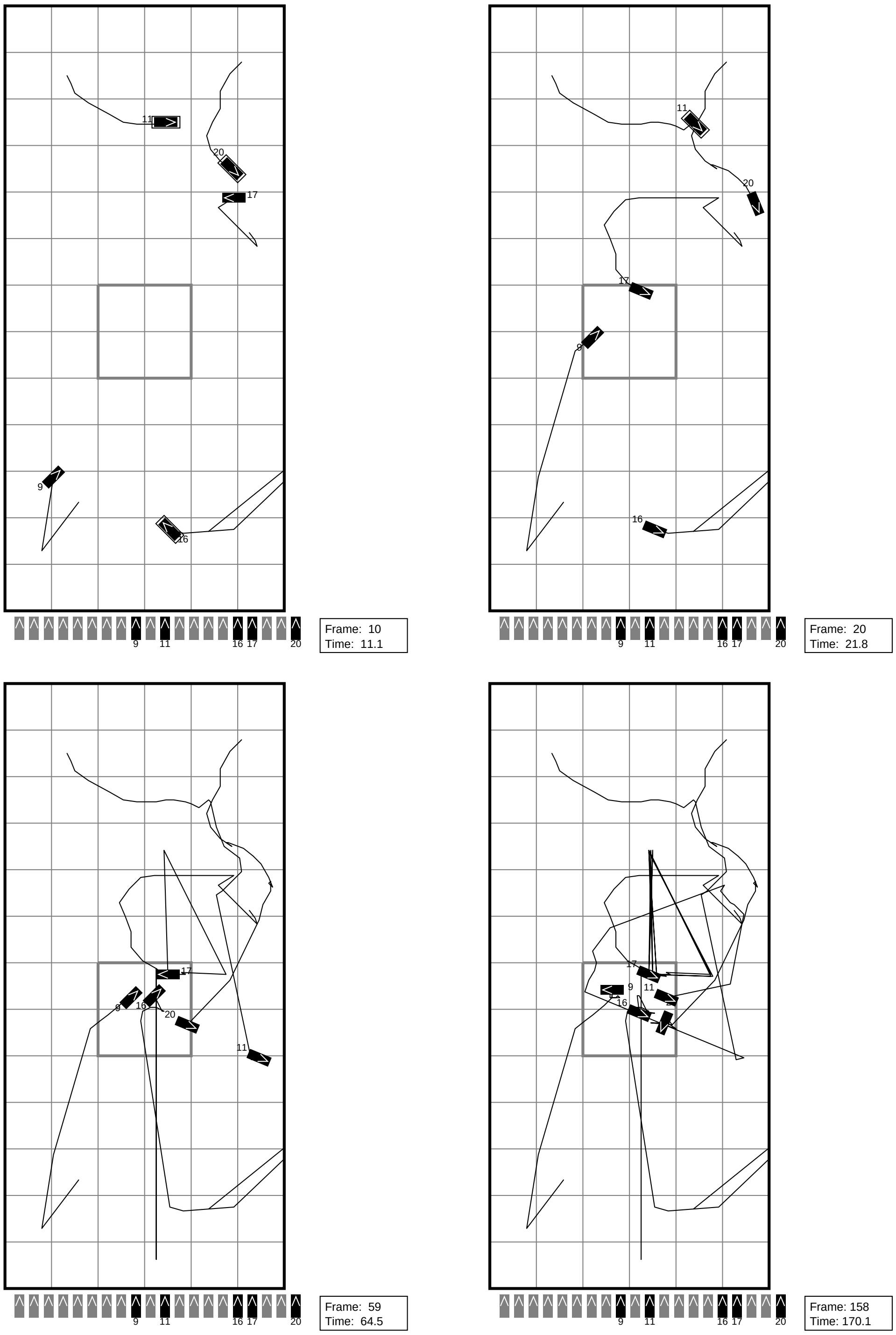


Figure 4-9: Another example of *homing* behavior of five robots, started in arbitrary initial positions. Trail histories demonstrate drastic errors in positioning, indicated by large jumps in consecutive robot location. In particular, the triangular path shown for robot #17 is due to repetitive position errors. In spite of the errors, all of the robots successfully reached home.

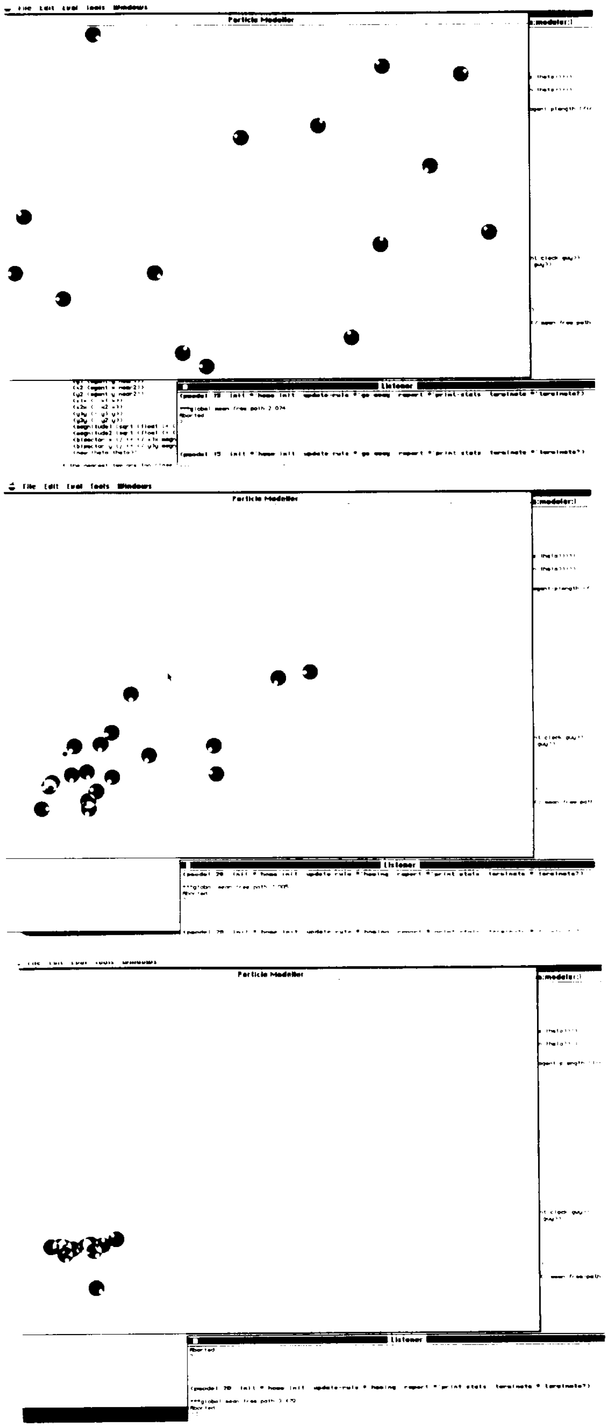


Figure 4-10: *Homing* behavior of a large group of simulated agents. Increased interference and competition for space is obvious around the goal region.

algorithm, when tested on the Interaction Modeler, produces more direct homing trajectories. Figure 4-9 shows another robot run of homing with five robots. In this run the entire time history of the robots' positions are shown, and the positioning errors can be easily seen. Nonetheless, all robots reach home. Figure 4-10 illustrates homing in simulation.

Individual homing is effective as long as the density of agents is low. If enough agents are homing within confined space, they interfere with each other. In the case of our non-holonomic robots, interference had even more enduring effects on the group. Figure 4-11 shows the growing interference between robots as they approach the goal region. Entire time–trails are shown to demonstrate how much group interference slows down individual performance.

Simulation and robot experiments described in this work show that interference increases if the agents have non–zero turning radii, unequal velocities, or are subject to sensor and control errors. All of the above conditions are common in situated agents, suggesting the need for some form of group or structured navigation, such as flocking, which will be introduced in an upcoming section.

4.5 Basic Behavior Evaluation

4.5.1 Empirical Evaluation of Basic Behaviors

Evaluation is one of the most difficult components of research, and it is somewhat new to the field of AI and Experimental Robotics. By nature and by design, the two fields are based on building artificial computational and physical systems. However, results from such synthetic endeavors do not fall cleanly into the well defined set of evaluation criteria designed for natural sciences. Analyzing something one has designed is intrinsically different from analyzing something externally imposed.

As a young and diverse field, AI still lacks standardized evaluation criteria. Consequently, it is left to each researcher to establish criteria that are both specific to the project and generally acceptable. The ideas proposed in this thesis are evaluated in two ways. The first addresses the merit of the general approach and its applicability to various domains. This evaluation is performed in the summary of the thesis, after the entire work has been presented. The second type of evaluation addresses the specific instantiation of the ideas in the spatial domain. This chapter presents the evaluation criteria applied to the implementations and performance of spatial basic behaviors and their composites.

AI and Robotics research in general is exploratory and often prone to phenomeno-

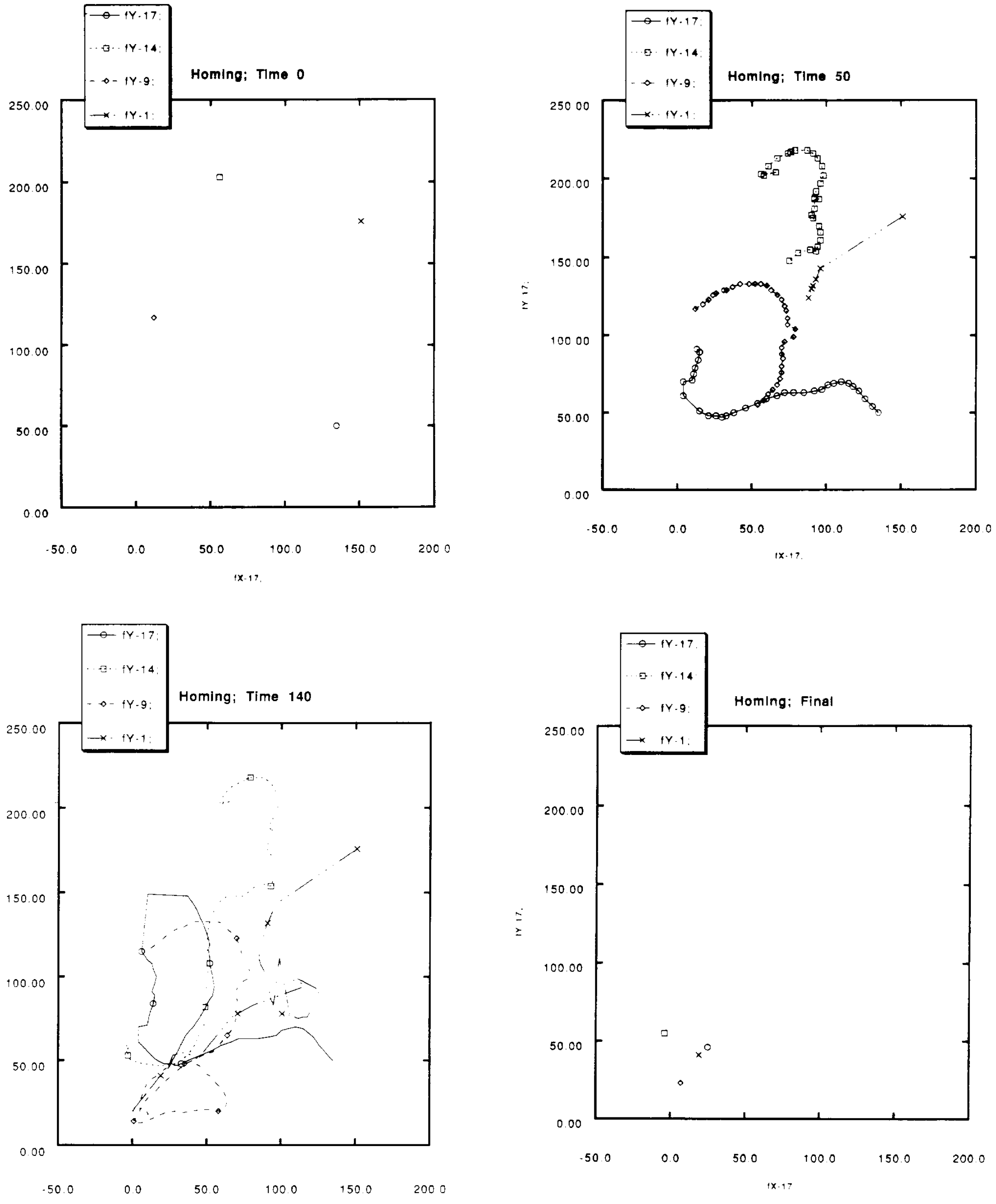


Figure 4-11: *Homing* behavior of four robots. Home is located in the region $(x, y) = (0..50, 0..50)$. Trails are marked with different patterns in order to demonstrate the increase in interference with proximity, resulting in circuitous paths.

logical evaluation. To prevent this, all of the evaluation criteria for the experimental part of the work were established prior to testing and were applied to the performance of each of the behaviors as well as to their combinations. An earlier section on basic behavior selection elaborated the criteria for choosing the basic behavior set and hinted at some evaluation procedures. This section gives a detailed illustration of empirical basic behavior evaluation on the example of *following*.

According to our pre-specified definition, a robot was said to be *following* when it maintained a minimal angle θ between itself and the leader. Repeatability and robustness of *following* were evaluated based on its manifested average uninterrupted duration, i.e. average time to failure. This duration was almost completely dependent on how reliably the front–pointing sensors could detect the "leader". Figure 4-12 illustrates continuous *following* behavior of 3 robots over a four minute period. The robot at the "front" of the queue is moving forward with its wheels slightly turned, thus tracing out a circular path. The other two robots follow their local "leader" according to the presented algorithm. The path of the first robot is smooth, while the followers oscillate in order to keep the robot ahead of them within IR range. One of the robots separated after two minutes, while the other two stayed together for the duration of the shown 243.3 second run. Figure 4-13 also illustrates the robustness of *following*; the robot in the lead moves about randomly and the follower keeps up throughout the duration of the run.

The range of the IR sensors used was directed and short, requiring the agents to stay close together within the queue. Consequently, errors in steering could cause a follower to lose sight of the leader if it failed to turn sufficiently in order to maintain the leader in sight. If the two continued to move in the same direction, as they would during a higher–level task, the follower could catch up with the leader again. If not, they would separate.

The narrow IR range explains why long queues and trains of agents were physically difficult to maintain. However, queues were stable and insensitive to dynamic obstacles and sensory and mechanical irregularities in the form of sensor noise, errors in steering, and perturbations in velocity. Figure 4-14 illustrates *following* on three robots in the presence of sensory or effector error. The middle robot stalls due to some error, and the robot behind it stops as well, then turns and follows the leader, as it senses the first robot in its range. The middle robot activates again, senses the second robot within its range, follows it, and the queue is maintained. Figure 4-15 demonstrates *following* in the presence of static constraints in the environment, such as walls and corners. The robots are able to avoid the walls and maintain the queue.

Following was also evaluated based on scalability in order to test its performance as agents are added and removed. The data above demonstrate the behavior that

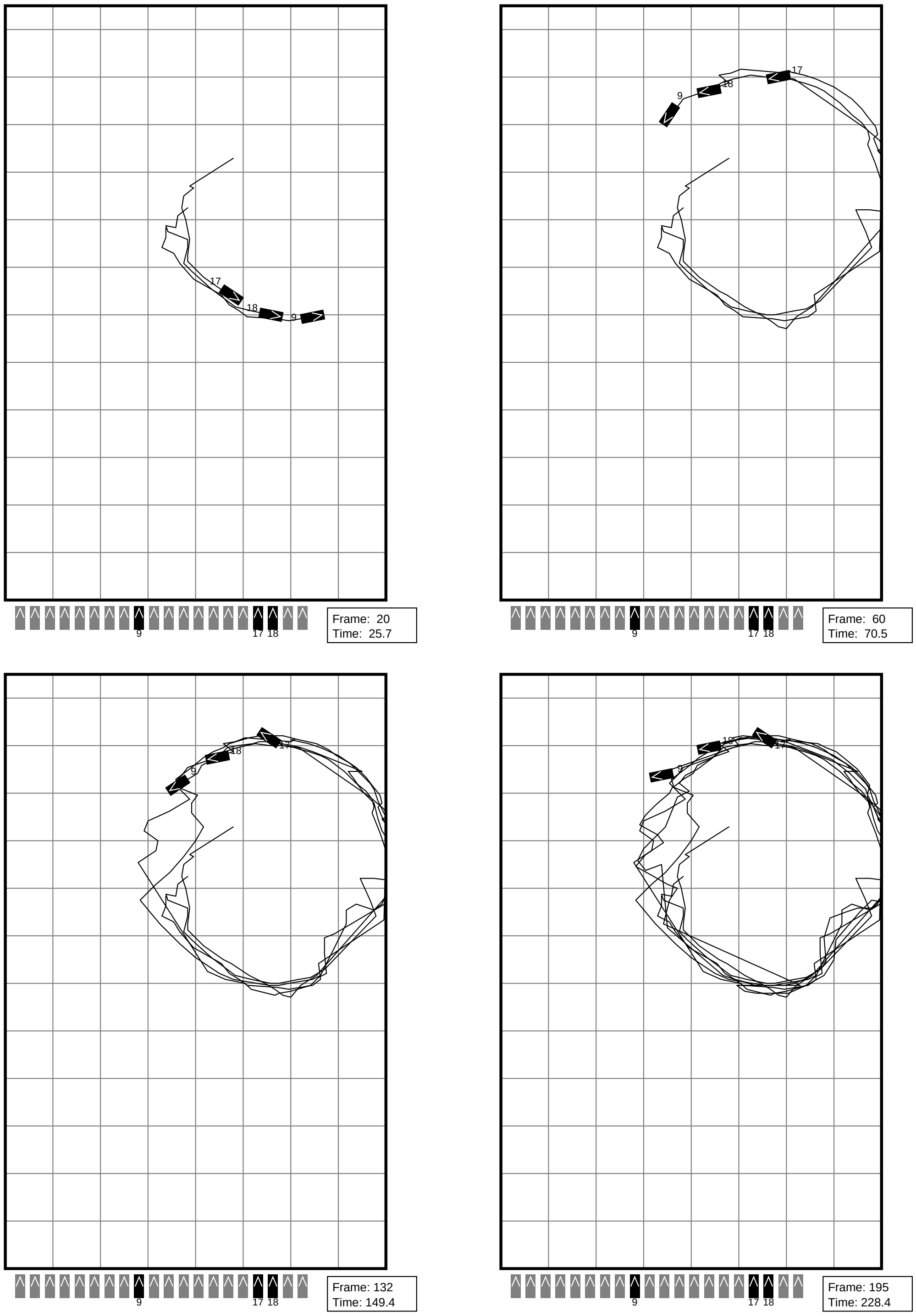


Figure 4-12: Continuous *following* behavior of 3 robots over 4.8 minutes. In the initial conditions, the wheels of the front robot are turned sideways, resulting in a circular trajectory. The robots reliably maintain a stable queue in spite of individual local variations in control.

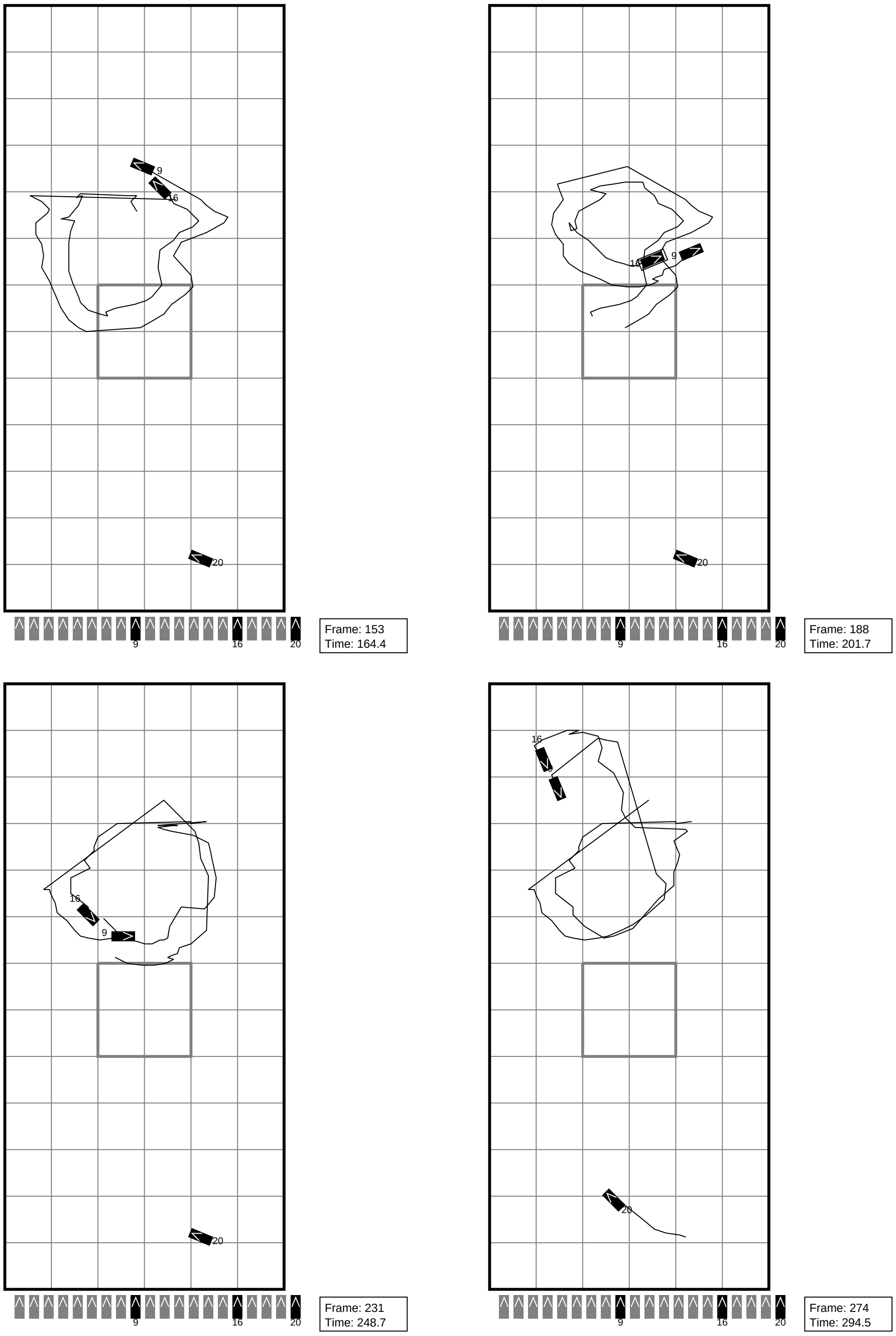


Figure 4-13: Continuous *following* performance of two robots over 4.9 minutes. The third robot (#20) is out of range so it does not join the others. The robot in the front moves about randomly, while the follower stays close behind.

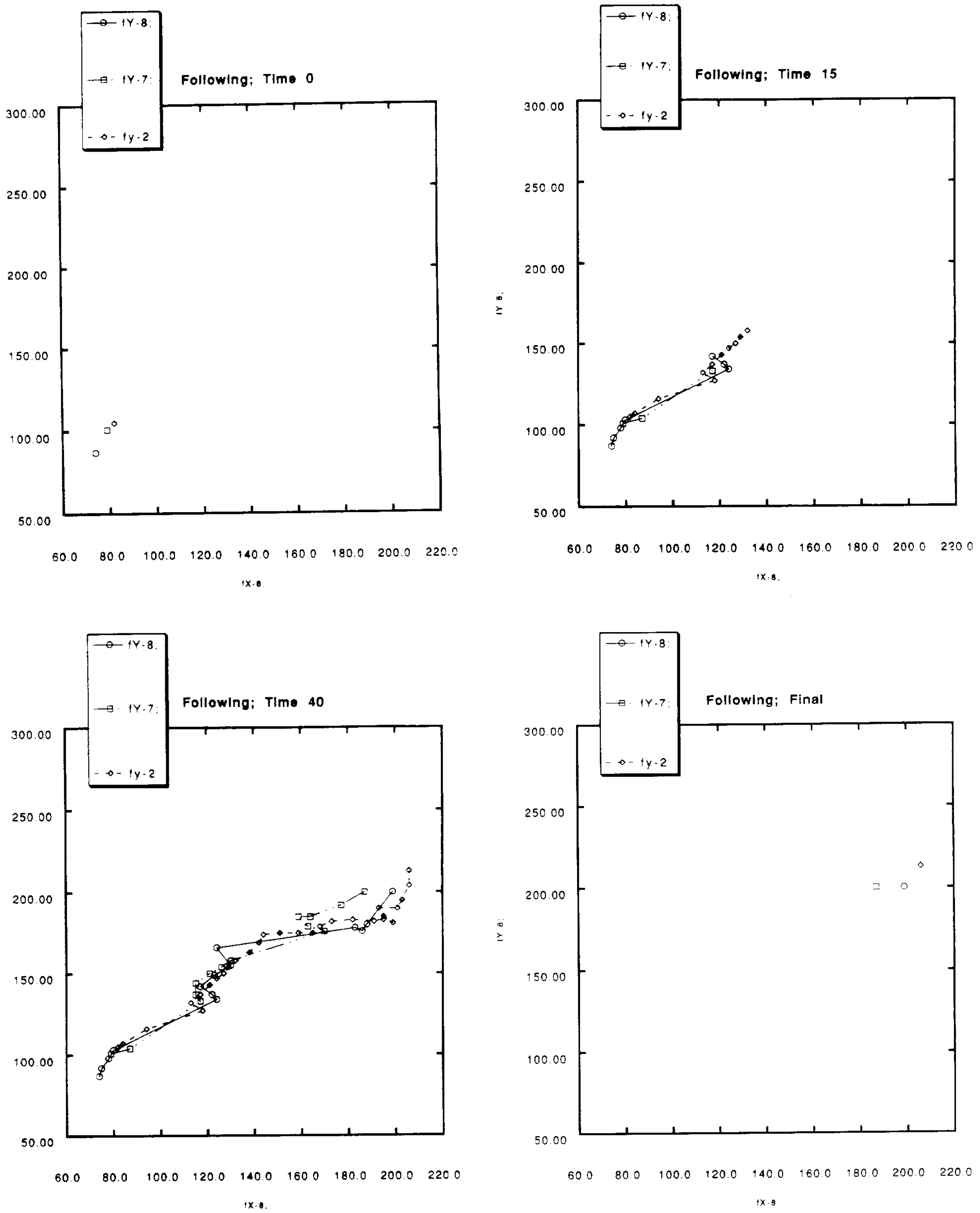


Figure 4-14: Performance of *following* with three robots in the presence of obstacles, sensory, and steering errors that cause the middle robot stall. The third robot passes it and maintains the first robot in its range. The middle robot senses the now–second robot within its range, follows it, and the queue is maintained.

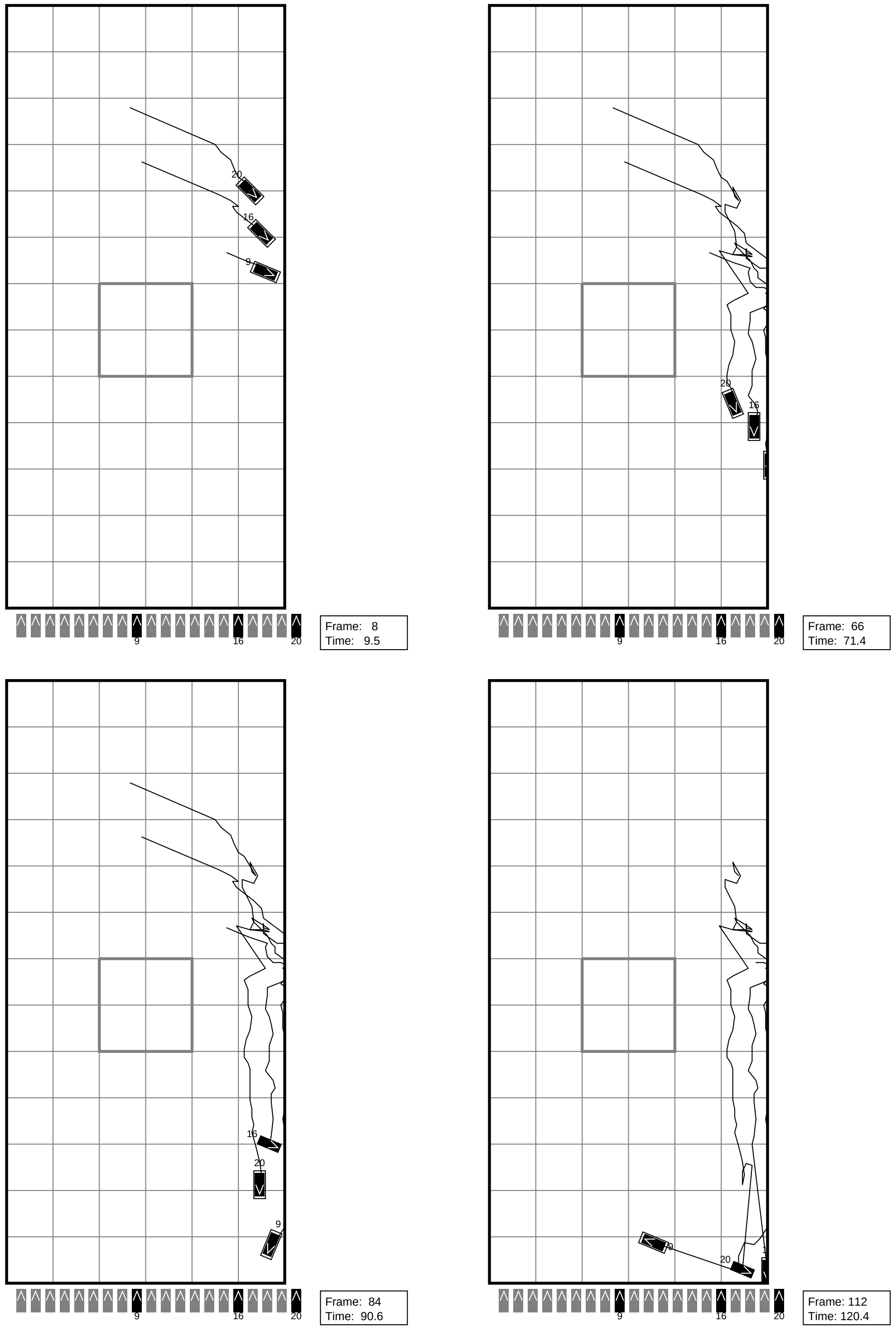


Figure 4-15: Performance of the *following* behavior of three robots in the presence of external obstacles and constraints. The robots maintain a queue while avoiding the wall and going around a corner.

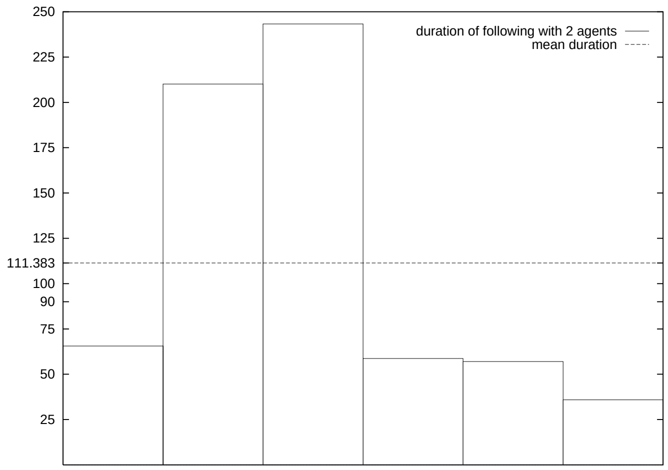


Figure 4-16: *Following* behavior of 2 robots. The x-axis plots individual trials, the y-axis plots the duration of uninterrupted following. The mean duration is indicated with the dashed line.

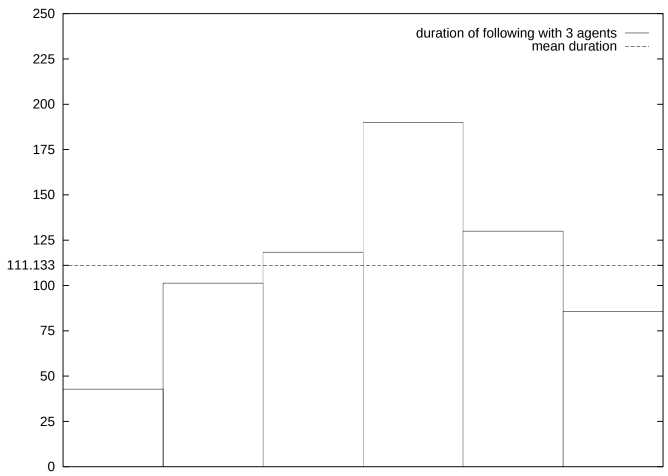


Figure 4-17: *Following* behavior with 3 robots. The x-axis plots individual trials, the y-axis plots the duration of uninterrupted *following*, in seconds. The mean duration is indicated with the dashed line.

results if an agent stalls, or is removed from the middle of the queue. The next set of data deals with the performance as new agents are added to the queue, the situation that is expected to happen more commonly, since *following* is, at a global level, a recruiting behavior.

Figure 4-16 demonstrates average following time for two robots in multiple runs. Figure 4-17 plots *following* data for three robots. The mean following time for two agents is nearly identical as that for three. This is exactly as expected, since *following* is a completely local behavior between two agents. The failure of any pair is as likely as the failure of any other, and the pairs are mutually independent, soq agents can be dynamically added and removed from the ends of the queue without affecting the rest.

This section has illustrated the criteria we used to evaluate the proposed basic behaviors. The evaluation process was illustrated on the example of *following*. The described criteria were systematically applied to all of the other basic behaviors as well.

4.5.2 Evaluation of Heterogeneous Groups

An obvious alternative for a fully distributed system of identical agents is a hierarchical distributed system. In order to evaluate the performance of the homogeneous basic behaviors, they were compared to particular hierarchical implementations. This section describes the performance of a hierarchical group of agents on two basic behaviors: aggregation and dispersion. These two behaviors were chosen because they can be stated in terms of achievement goals and, given sufficient space, can reach a static state. The algorithms were evaluated based on the time or the number of steps required to reach that well–defined state.

A version of hierarchical agents was implemented by classifying the agents into a total order, based on a randomly assigned unique ID number, thus simulating an established pecking order in the group (Chase, Bartolomeo & Dugatkin 1994, Chase 1993, Chase 1982, Chase & Rohwer 1987). While in homogeneous algorithms all agents moved simultaneously according to identical local rules, in the hierarchical case the ID number determined which agents were allowed to move while others waited. In all cases, a simple precedence order, a spatially–local hierarchy, was established such that within a small radius the agent with the highest ID got to move. Multiple types of dispersion and aggregation algorithms were tested with such hierarchical agents.

Using the Interaction Monitor, 20 experiments were conducted with each group size (3, 5, 10, 15, and 20 agents) and each of the algorithms. Additionally, the algorithms were tested on two different degrees of task difficulty. Aggregation was tested

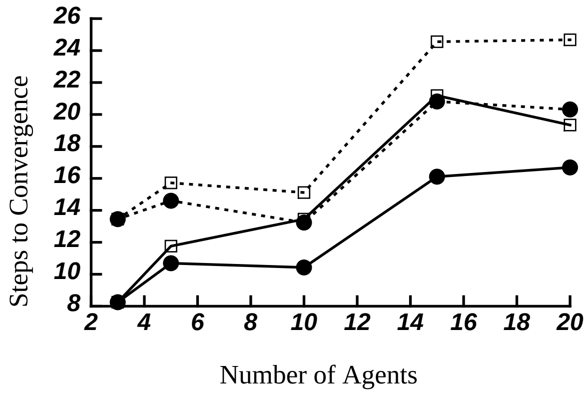


Figure 4-18: The performance of two different *aggregation* algorithms based on time required to reach static aggregated state. Two termination conditions were tested: a single group (data points shown with boxes) and a few stable groups (data points shown with dots). The performance of hierarchical algorithms is interpolated with solid lines while the homogeneous ones are interpolated with dotted lines.

on two terminating conditions: a single aggregate containing all of the agents, and a small number of stable aggregates. The former terminating condition is more difficult. Similarly, dispersion was tested on two initial conditions: a random distribution of initial positions, and a packed distribution in which all of the agents start out in one half of the available space. The latter condition is more difficult.

It was found that, in the case of aggregation, hierarchical strategies performed somewhat better than our homogeneous approaches. Figure 4-18 plots the average number of moves an agent takes in the aggregation task against the different group sizes and the two different terminating conditions: a single aggregate and a few stable groups. Both hierarchical and homogeneous algorithms behaved as expected, performing better on the simpler of the two terminating conditions. Their performance declined consistently with the growing group size.

Unlike aggregation, in the case of dispersion, homogeneous strategies outperformed hierarchical ones. Figure 4-19 plots the average number of moves an agent makes in the dispersion task for the different group sizes on two different initial conditions: a random distribution, and a packed initial state. Again, both hierarchical and homogeneous algorithms improved with the easier initial conditions.

Although the performance difference between the homogeneous and hierarchical algorithms was repeatable and consistent, it was small, and its magnitude barely surpassed the standard deviation among individual trials for each of the algorithms and group sizes. The standard deviation was particularly significant in the case of small (3 and 5) group sizes. Thus, no statistically significant difference was found in

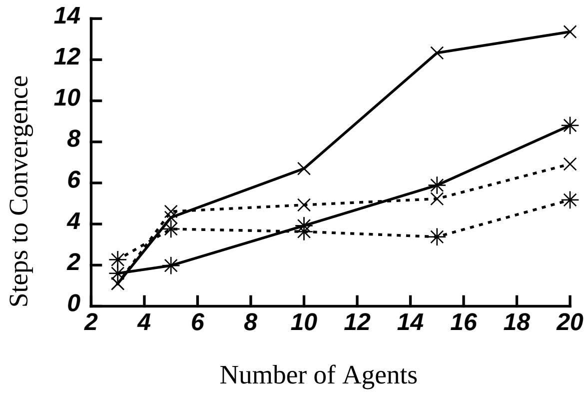


Figure 4-19: The performance of two different *dispersion* algorithms based on the time required to reach static dispersed state. Two initial states were tested: a random distribution (data points shown with stars) and a packed distribution (data points shown with crosses). The performance of the hierarchical algorithms is interpolated with solid lines while the homogeneous ones are interpolated with dotted lines.

global performance of hierarchical and flat algorithms for aggregation and dispersion. Furthermore, the slight differences that were detected between the two strategies would mostly likely be negligible on physical agents, due to sensor uncertainty and effector errors.

We believe that the similarity in performance between the homogeneous and simple heterogeneous algorithms is caused by the following:

- **Functionally homogeneous agents**: In spite of the linear priority ordering, the agents are fundamentally homogeneous since they are functionally indistinguishable. Thus, the hierarchical relationships between agents are spatially and temporally independent, since the agents keep no history of their past encounters with each other.

- **Simplicity of behavior**: The only behavior being observed is spatial, in the domain where the consequences of actions of identical agents have no time–extended consequences.

- **Large group sizes:** In sufficiently large groups of functionally identical agents, temporary effects are averaged out as fluctuations and noise. This property is crucial for producing reliable global behavior in the presence of local perturbations, and is observable in the shown data: the general trends in global performance are consistent even although the standard deviation among trials is quite large.

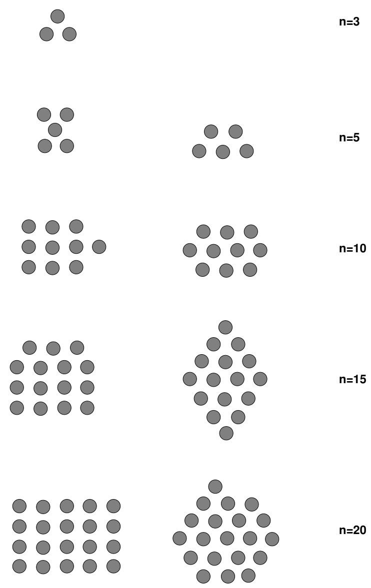


Figure 4-20: The initial conditions used for comparing *dispersion* algorithms. Maximally packed states for five different group sizes (3, 5, 10, 15, and 20) were tested.

The experiments comparing simple hierarchical and homogeneous algorithms demonstrate that, in the described domain, simple hierarchical strategies do not affect the global performance because their impact on the global behavior is negligible. More complex hierarchical strategies could be devised, in order to assure their influence on the global behavior, but would require an increased perceptual and cognitive overhead, such as perhaps keeping a history of past encounters and models of previously encountered agents. This data permit us to hypothesize the following: for simple spatial domains 1) simple homogeneous solutions can work quite well, and 2) more complex strategies requiring individual agents to perform recognition, classification, and representation may be are required to significantly improve group performance.

4.5.3 Evaluating Distributed v. Centralized Algorithms

The beginning of the thesis compared centralized and distributed approaches, and argued that centralized approaches do not scale for the types of systems this thesis has dealt with. For the purposes of comparison, however, a set of special case scenarios was constructed, for which optimal centralized solutions could be computed for the dispersion task. While computing the optimal dispersion strategy for an arbitrary configuration of agents is difficult and, for large group sizes, intractable, the strategy can be computed for special classes of initial positions.

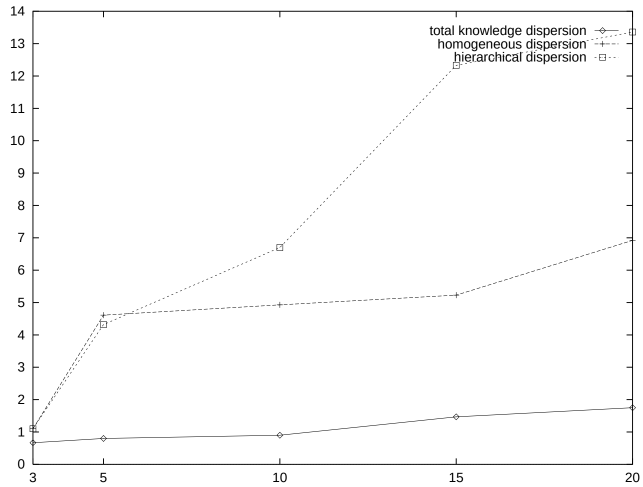


Figure 4-21: The performance of the optimal global "total knowledge" algorithm for *dispersion* (data points shown with diamonds) compared with the hierarchical and homogeneous dispersion strategies (data points shown with boxes and crosses, respectively).

Packed configurations of agents were designed for five group sizes: 3, 5, 10, 15, and 20, as shown in Figure 4-20. These configurations were chosen for two reasons: 1) they presented challenging initial conditions for dispersion, and 2) optimal dispersion solutions could be computed by taking advantage of the symmetry of configurations. Optimal solutions employed the general strategy of moving the outer agents first until enough space is cleared for the next layer to move, and so on. The average number of moves per agent for obtaining a dispersed state was computed for each of the group sizes.

The "total knowledge" algorithm was tested along with the existing hierarchical and homogeneous algorithms on the Interaction Modeler. The data for the distributed algorithms were averaged over 20 trials for each group size. Figure 4-21 plots the performance of the three algorithms.

Not surprisingly, the total knowledge algorithm performs the best. However, it is important to note that although its performance declines slower than that of the distributed algorithm, the two are only offset by a constant factor. Given that the performance of the total knowledge algorithm is not practically attainable in real–time, the distributed alternative with minimum computational and sensing overhead presents a useful alternative.

4.6 Summary

This chapter has introduced the methodology for selecting basic behaviors and demonstrated it on the spatial domain. A basic behavior set consisting of *safe–wandering, following, dispersion, aggregation,* and *homing* was proposed, implemented in two different experimental environments, and tested in simulation and on physical robots. Experimental data were evaluated using a collection of criteria we specified *a priori.* The performance of the basic behaviors was also tested compared against hierarchical and total knowledge approaches.

The next chapter introduces ways in which the described basic behaviors can combined in order to be achieve a variety of higher–level goals and tasks.

Chapter 5

Combining Basic Behaviors

5.1 Two Types of Behavior Combination

Basic behaviors are designed to be a substrate for a variety of more complex compound group behaviors for a given domain (Figure 5-1). Generating compound behaviors requires applying some kind of a combination operator whose properties are well understood and which produces the desired output composite behavior. This is is considered to be one of the challenges of behavior–based control, i.e., *arbitration*, the problem of coordinating the activity of multiple input behaviors in order to produce desired output behavior.

Depending on the complexity of the system, arbitration can and usually must be performed at multiple points. One level of arbitration can be achieved by designing mutually exclusive behavior conditions (Matarić 1992*c*). Creating a unique one–to–one mapping between conditions and behaviors guarantees a mutually exclusive set of condition–action couplings. In contrast, if the mapping is one–to–many, so that a condition can result in more than one possible behavior, then there is a possibility that two or more behaviors may be in conflict.

Mutually exclusive behavior conditions are sufficiently powerful for arbitrating in a system that performs only one behavior at a time. However, in more complex systems, multiple behaviors can contribute to the output (Parker 1994, Payton, Keirsey, Kimble, Krozel & Rosenblatt 1992, Ferrell 1993). Consequently, most practical systems use mutually exclusive behavior conditions within a coherent layer or submodule of the system dealing with a particular coherent set of tasks. Between modules and layers another level of arbitration is necessary which either implements a type of a sum of the inputs or a switch. The general form of a behavior–based system involves such combination operators at one or more levels.

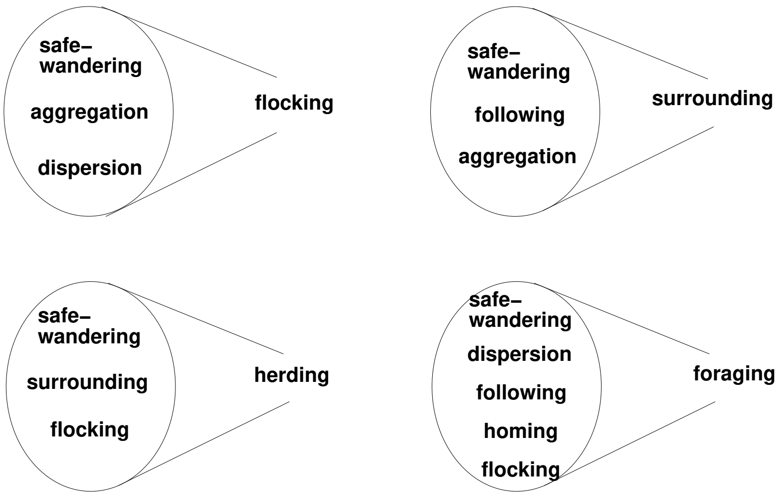


Figure 5-1: Basic behaviors can be combined to generate a variety of more complex behaviors.

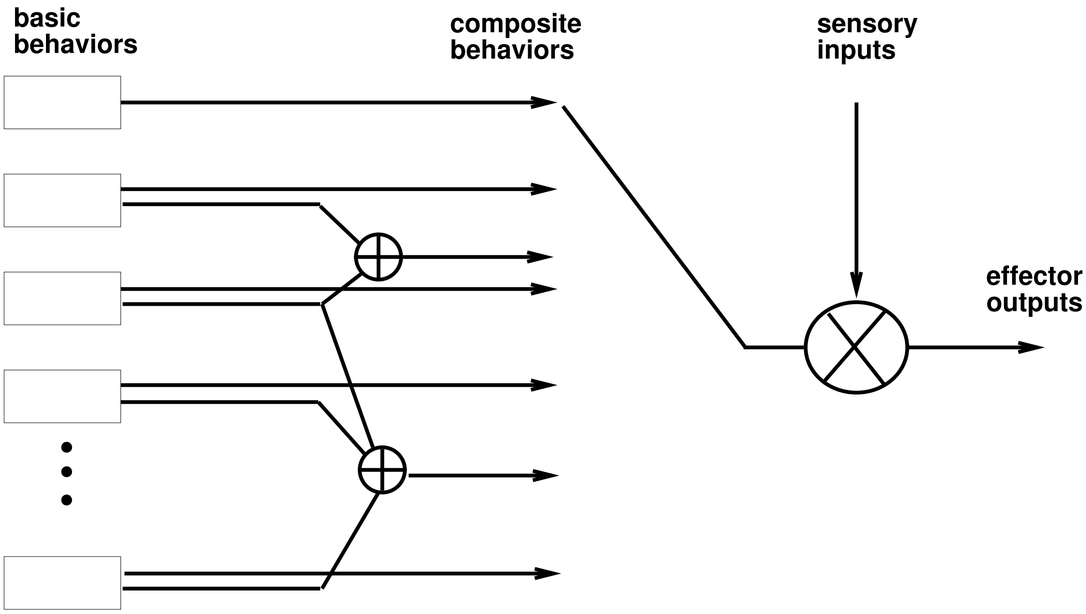


Figure 5-2: The control architecture for generating group behaviors consists of direct and temporal combinations of subsets from a fixed basic behavior set. Direct combinations are marked with $\oplus$, temporal combinations with $\otimes$.

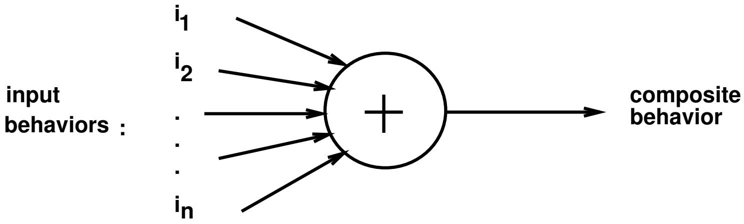


Figure 5-3: The general form of direct behavior combinations. Outputs from two or more behaviors are summed.

The architecture proposed here for combining basic behaviors has the described general form. In order to take advantage of the expressive combinatorial power of the basic behaviors, the architecture uses both combination operators: behaviors can be combined directly, by executing multiple behaviors at once, and temporally, by sequencing the behaviors one at a time. Direct combinations allow for multiple concurrently active behaviors to contribute to outputs. Temporal combinations assure a coherent sequence of the outputs. The two types of combination operators, applied to the fixed set of basic behaviors, can generate an unbounded repertoire of collective behaviors, because temporal combinations can extend arbitrarily in time (Figure 5-2). The following sections describe the operators and demonstrate them with implemented compound behaviors.

5.1.1 Direct Combinations of Basic Behaviors

A direct combination of behaviors is some function of the outputs of a subset of the basic behaviors, as illustrated in Figure 5-3. In the spatial domain, the outputs of all of the basic behaviors are in the form of direction and velocity vectors, so appropriately weighted sums of such vectors directly produce coherent higher–level behaviors. This method is illustrated by using direct combination to implement flocking.

Flocking is defined as collective motion that satisfies the following constraints: all of the agents within sensing range of each other must stay within a flocking range from their neighbors as they move. Unlike *aggregation*, *flocking* not only requires the agents to stay together, but also to move toward a goal, generically referred to as *home*. Formally:

$$\forall(i,j)\ \ d_{i,j} < \delta_{flock}\ \ and\ \ \frac{dp\mathcal{C}_g}{dt}\cdot(\mathcal{C}_g - p_{home}) < 0$$

Flocking can be implemented by combining the outputs of *safe–wandering*, *aggregation*, *dispersion*, and *homing*, such that the specified constraints are satisfied, as

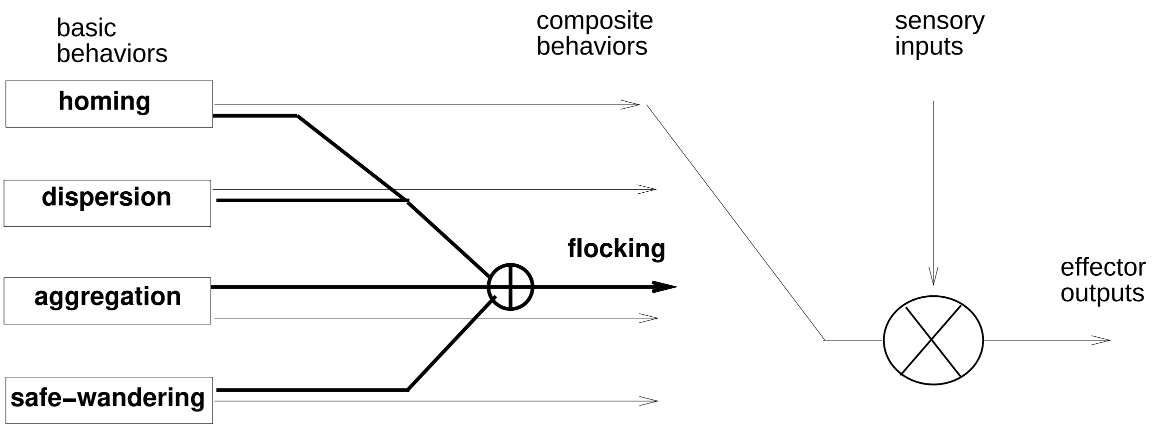


Figure 5-4: The implementation of flocking as a combination of *safe–wandering*, *dispersion*, *aggregation*, and *homing*. *Safe–wandering*, *aggregation*, and *dispersion* produce robust *flocking*, and *homing* gives the flock a goal location and direction to move in.

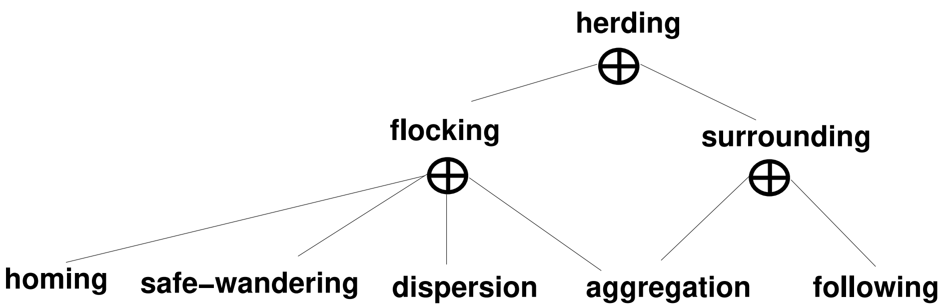


Figure 5-5: An example of direct basic behavior combination within a higher–level task.

shown in Figure 5-4. Intuitively, *aggregation* keeps the robots from getting too far from each other, *dispersion* keeps them from getting too close, and *safe–wandering* prevents each agent individually, and thus the flock as a whole, from colliding with any non–agent obstacles, and *homing* moves the flock toward some goal. Flocking can be further reduced to a combination of just *safe–wandering*, *aggregation*, and *homing* for a range of values of δ_{flock}, such that $\delta_{flock} < \delta_{aggregate}$, so that *safe–wandering* also has a dispersing effect.

The given set of basic behaviors allows for many other direct composites, such as *surrounding*, a combination of *aggregation* and *following*, and *herding*, a combination of *surrounding* and *flocking*, as shown in Figure 5-5.

For any given high–level goal, the structure of direct behavior combination is a

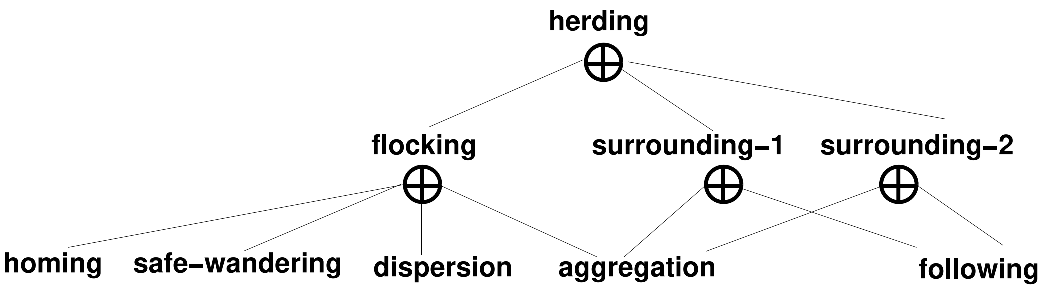


Figure 5-6: Direct behavior combinations use continuous summing functions. Consequently, the same basic behaviors can be reused and recombined repeatedly within a common higher–level goal. In this example, two types of *surrounding* are used, depending on the sensory conditions.

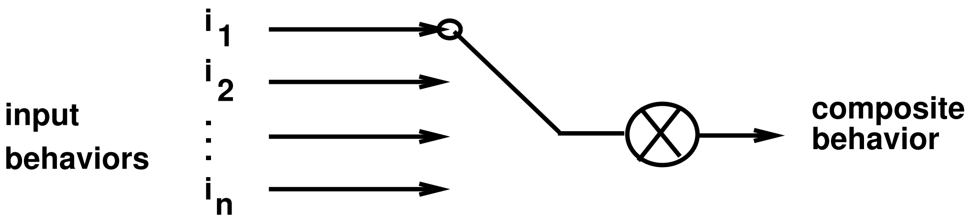


Figure 5-7: The general form of temporal behavior combinations switches between mutually exclusive behaviors. Only one behavior is active at a time, resulting in a behavior sequence triggered by different sensory conditions.

directed acyclic graph (DAG) with behaviors as nodes and inheritance relations as arcs. The semantics of the arcs are identical to the semantics of the $\oplus$ and $\otimes$ combination operators. Basic behaviors are the originator nodes of the graph. Except for the final high–level behavior node, all other nodes are combinations of originator and other intermediate nodes in the graph. Figure 5-5 illustrates an example of a graph in which *aggregation* is shared by two intermediate nodes: *flocking* and *surrounding*. Since behavior combinations are based on continuous function (weighted sums) of the input parameters, the same nodes can be used in multiple combinations. For example, figure 5-6 illustrates the use of the same basic behaviors (*aggregation* and *following*) to construct two different types of *surrounding* behaviors, and then combining both in *herding*.

5.1.2 Temporal Combinations of Basic Behaviors

Basic behaviors and their direct combinations achieve and maintain single goals.

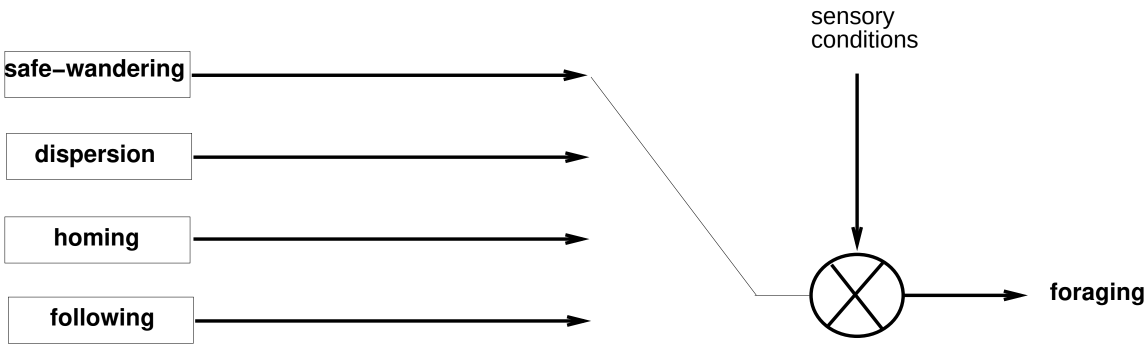


Figure 5-8: The implementation of foraging using a temporal combination of *safe–wandering*, *dispersion*, *homing*, and *following*. Each triggered by different sensory conditions, the behaviors collectively result in foraging.

For example, *dispersion* achieves the goal of establishing a minimum distance between all of the agents while *following* maintains the goal of preserving a queue of moving agents each of which is within a given distance and direction from its neighbors. In order to achieve higher–level tasks defined by multiple sequential goals, basic behaviors must be properly temporally combined.

Such combinations are temporal sequences of basic behaviors, each of which is triggered by appropriate conditions in the environment, as shown in Figure 5-7. Combining interactions temporally relies on the agents' ability to perceive the state that triggers a behavior change. Given this ability, simple finite state machine controllers can be designed to generate a variety of multi–goal behaviors. This method is illustrated on an implementation of *foraging*, a group task of gathering objects ("food") from the environment (Figure 5-8).

In foraging, the high–level achievement goal of the group is to collect objects from the environment and deliver them home. This complex behavior is a prototype for a variety of tasks including harvesting, garbage collection, and clearing toxic spills and mine–fields. For the foraging task, in addition to having the basic behavior repertoire, individual agents are also able to search for pucks, pick them up, and drop them. Furthermore, foraging uses a restricted notion of kinship defined by the agents' "puck state:" any two robots without pucks are "kin", as are any two that are carrying pucks. Since the robots cannot directly sense each other's external state, puck state is is broadcast by each of the robots within a limited range via the radios.

Foraging is initiated by *dispersion*[1], and then *safe–wandering*. Finding an object triggers *homing*. Encountering another agent with a different immediate goal, as

[1] Floreano (1993) shows that evolved systems of ants favor dispersion as the first step in foraging.

Condition					Behavior
at-home?	have-puck?	crowded?	behind-kin?	sense-puck?	
0	0	0	0	0	safe–wandering
0	0	0	1	0	following
0	0	1	0	0	dispersion
0	0	1	1	0	dispersion
0	1	0	0	0	homing
0	1	0	1	0	following
0	1	1	0	0	dispersion
0	1	1	1	0	dispersion
1	0	0	0	0	safe–wandering
1	0	0	1	0	following
1	0	1	0	0	dispersion
1	0	1	1	0	dispersion
1	1	0	0	0	drop-puck
1	1	0	1	0	drop-puck
1	1	1	0	0	drop-puck
1	1	1	1	0	drop-puck
0	0	0	0	1	pickup-puck
0	0	0	1	1	pickup-puck
0	0	1	0	1	pickup-puck
0	0	1	1	1	pickup-puck
0	1	0	0	1	homing
0	1	0	1	1	following
0	1	1	0	1	dispersion
0	1	1	1	1	dispersion
1	0	0	0	1	safe–wandering
1	0	0	1	1	following
1	0	1	0	1	dispersion
1	0	1	1	1	dispersion
1	1	0	0	1	drop-puck
1	1	0	1	1	drop-puck
1	1	1	0	1	drop-puck
1	1	1	1	1	drop-puck

Table 5.1: The controller for *foraging*. For brevity, conditions for avoidance are left out. Whenever one is sensed, the agent executes the avoidance rules of *safe–wandering*.

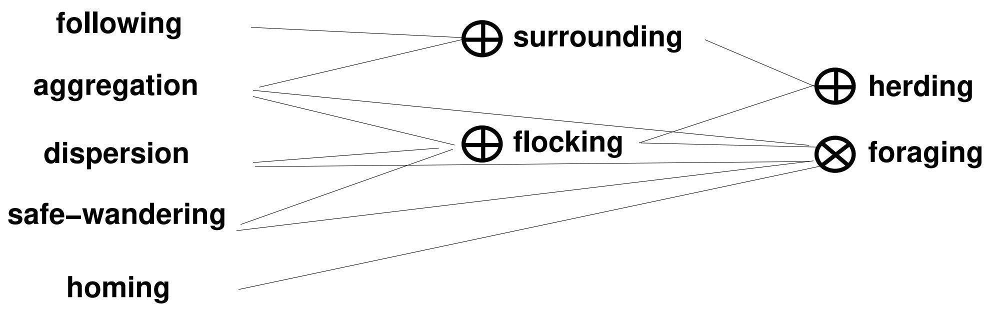


Figure 5-9: An example of applying both direct and temporal combinations to the same basic behaviors to generate various higher–level behaviors. In this case, *safe–wandering* is used to generate *flocking*, and it is used in foraging. Similarly, *aggregation* is used in *foraging* and in *surrounding*.

manifested by its external state, e.g., not carrying a puck induces *safe–wandering* away from the object. Conversely, encountering kin triggers *flocking*. Reaching home and depositing the object triggers *dispersion* if multiple robots are at home, or *safe–wandering* if the robot is alone. Figure 5.1 shows the controller for the task.

Foraging demonstrates how basic behaviors can be temporally combined into a higher–level compound behavior. The combination is simple in that conflicts between two or more interacting agents, each potentially executing a different behavior, are resolved uniformly due to agent homogeneity. Since all of the agents share the same goal structure, they will all respond consistently to environmental conditions. For example, if a group of agents is following toward home and it encounters a few agents dispersing, the difference in the agents' external state will either induce following agents of the same kind or avoiding agents of any other type, thus dividing or "specializing" the group again.

Foraging is just one example of a variety of spatial and object manipulation tasks that can be implemented with the described architecture and the given basic behaviors. Other tasks include sorting objects, building structures, surveying and mapping an unknown territory, and many others.

Figure 5-9 illustrates how the same basic behaviors, in this case *dispersion* and *safe–wandering*, can be used in a direct combination, *flocking*, and also in a temporal combination, *foraging*.

The next section demonstrates robot implementations of compound behaviors.

```
Flock:
Sum outputs from Safe--Wander, Disperse, Aggregate, and Home.
```

Algorithm 5.1:

5.2 Implementations of Compound Behaviors

5.2.1 Flocking

As described earlier, *flocking* is a form of structured group movement that serves to minimize interference, protect individuals, and enable efficient information exchange. Flocking was implemented with a simple algorithm shown in Algorithm 5.1.

The choice of weights on the different behavior outputs was determined by the dynamics and mechanics of the agents, the ranges of the sensors, the agents' turning radii, and their velocity. In the robot implementation, flocking consisted of a combination of safe–wandering and aggregation only, by using the appropriate combination of δ_{avoid} and $\delta_{aggregate}$ thresholds.

Like following, flocking is a coordinated–motion behavior which is best evaluated by testing its duration, repeatability and robustness. As expected, performance of flocking was dependent on the size of the flock. Small flocks, consisting of four or fewer agents, were less stable[2], while larger flocks remained stable even if any of the agents failed due to mechanical problems. Figure 5-10 demonstrates just such a case, in which one of the agents' position sensors failed and it quickly diverged from the flock.

The utility of flocking can easily be seen through its interference–minimizing properties. For instance, it is much more efficient than individualistic homing as the number of homing agents increases. Although flocking involves a compromise between individual and group goals, which may make an individual agent's path locally suboptimal, the collective behavior is more efficient in that all of the agents get to the destination faster than they do, on the average, using individualist greedy homing strategies[3].

Typical flocking behavior is shown in figures 5-11, 5-12, and 5-13. Flocking was also tested in more challenging environments. For example, a barrier roughly the size of two robots was presented in front of the flock as the flock was moving. As expected,

[2]According to the definition of stability given in Chapter 4.1

[3]Traffic laws are human forms of following and flocking. They impose structure on the collective motion so as to minimize average interference.

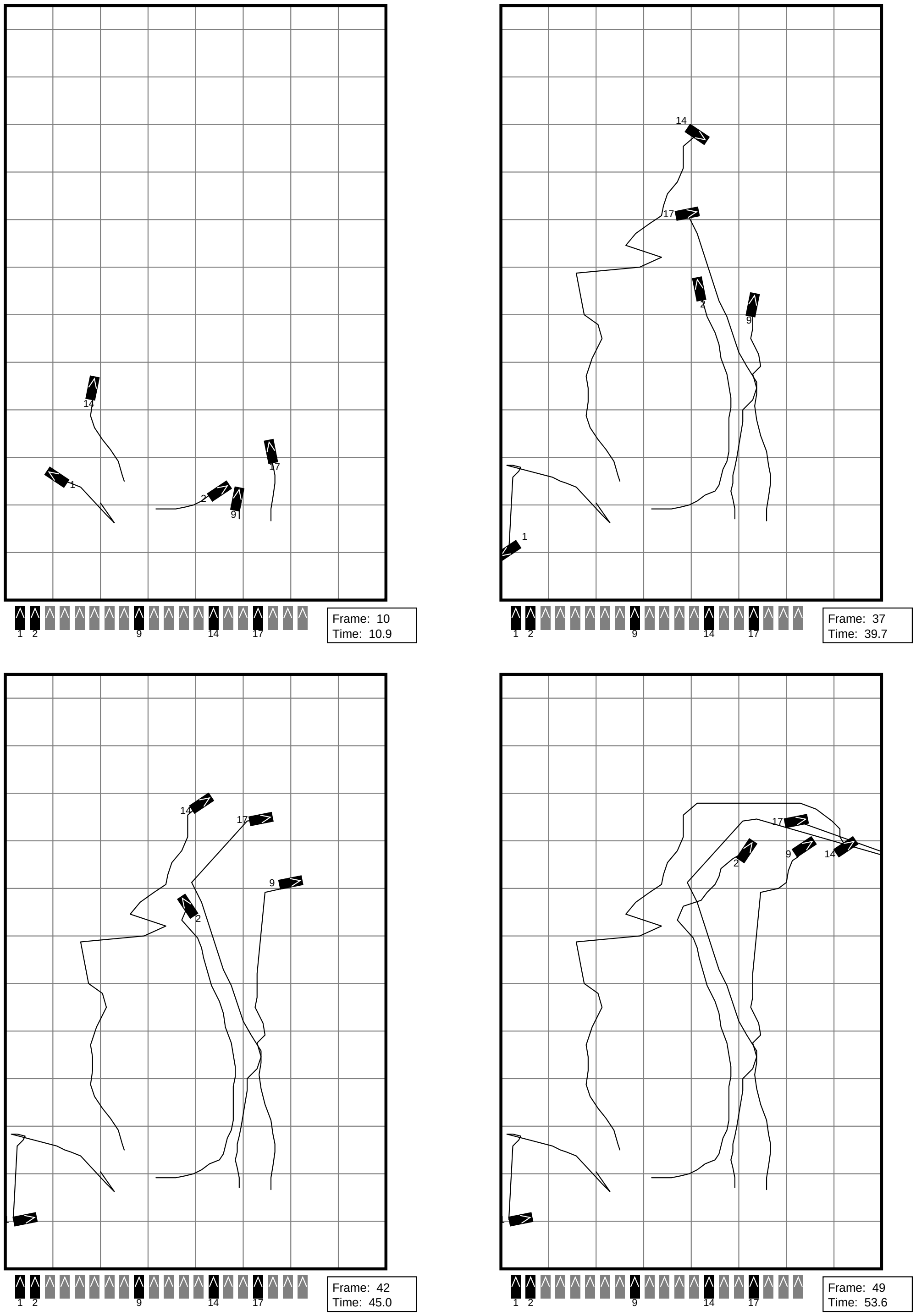


Figure 5-10: *Flocking* behavior of five robots. One of the robots separates, without affecting the behavior of the others. Due to a failure of the position sensors, the robot falls behind the group and cannot rejoin them. The rest of the robots reorganize and maintain the global structure.

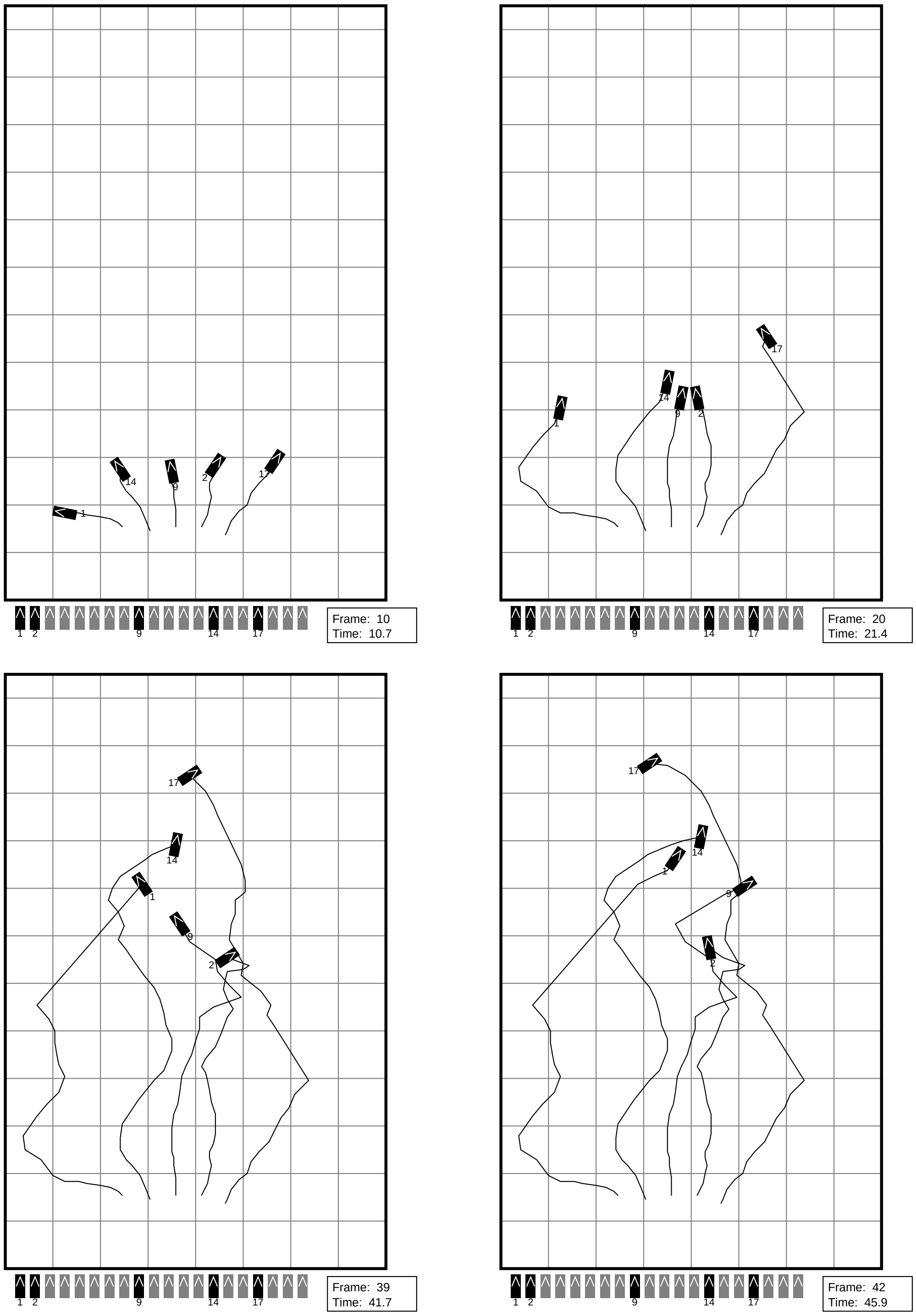


Figure 5-11: *Flocking* behavior of the same five robots in another trial. The robots maintain a coherent flock, in spite of the often large position errors sensed by individuals. These errors are manifested in the variability in the spacing between the robots as the flock moves.

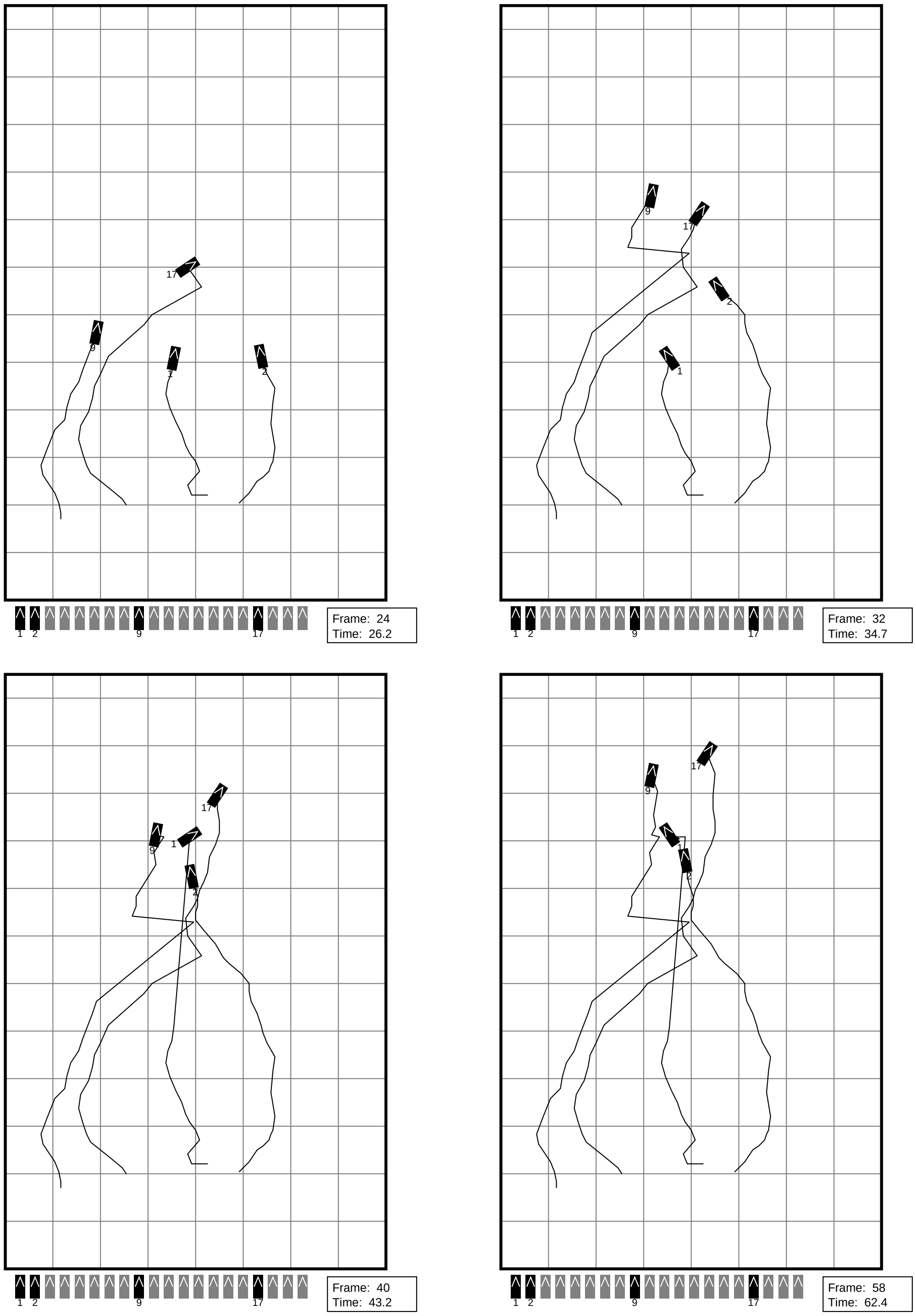


Figure 5-12: *Flocking* behavior of four robots. The robots are initiated in a line and they quickly move into a flock. There are no fixed leaders so robots at the front of the flock occasionally exchange places with others due to velocity and other control variations, all the while maintaining the flock formation.

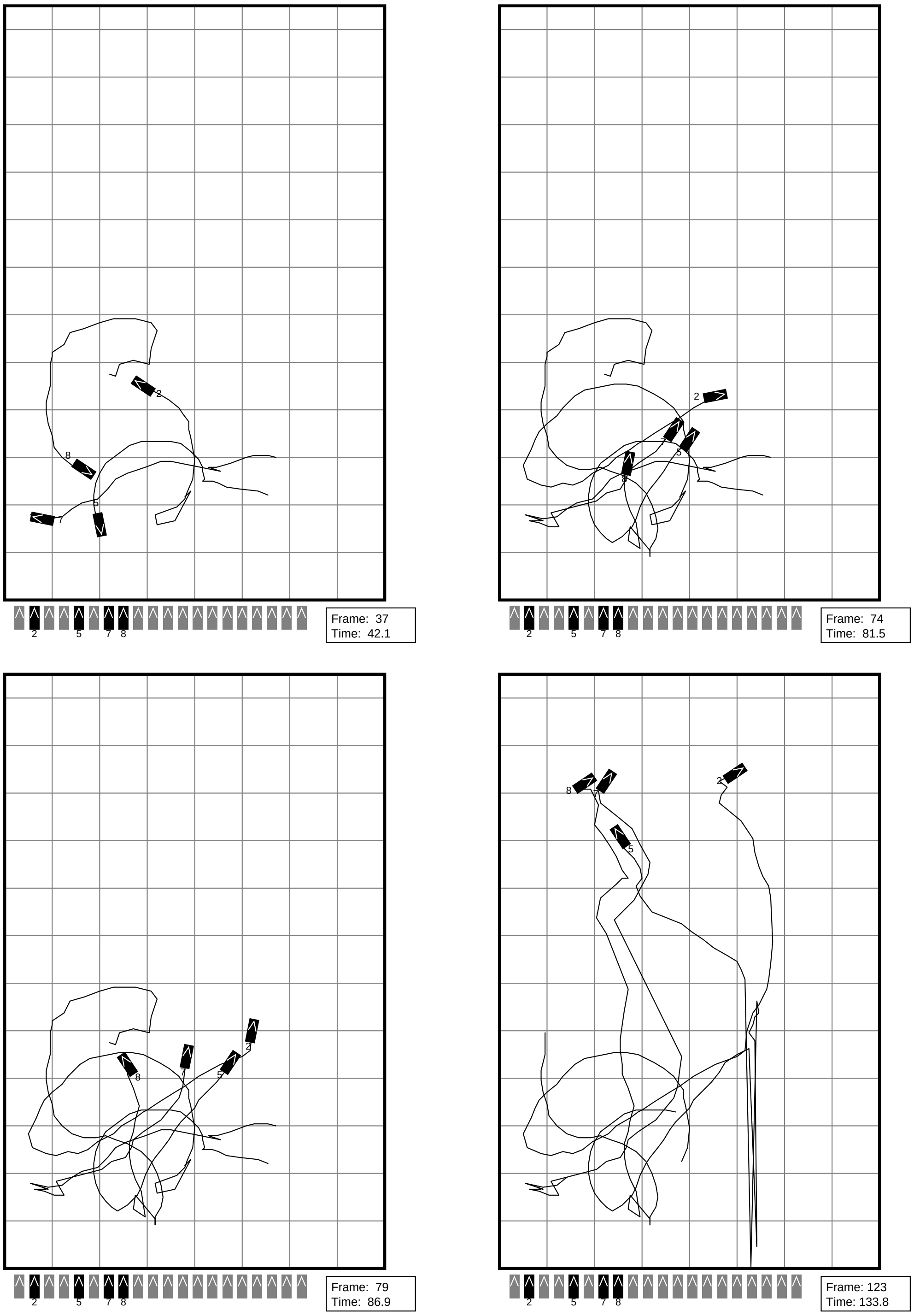


Figure 5-13: Another run of four *flocking* robots. The robots are started in a difficult initial configuration: facing each other. After an initial reordering they establish a flock and maintain it as they move across the workspace. As shown in the last frame, the position sensors on robot #2 faltered so its path appears discontinuous, but its actual trajectory keeps it with the flock.

the flock split into two groups around the obstacle and rejoined on the other side. Empirical data for this and other experiments is available on video tape.

The idea that flocking can be generated by simple rules has been popular among many researchers. For example, DeScnutter & Nuyts (1993) and Goss, Deneubourg, Beckers & Henrotte (1993) show a similar approach by demonstrating how simple rules can result in gull flock formation in simulation. Even more directly, Reynolds (1987) presents an elegant graphical simulation of bird flocking. However, the robot implementation required more rules due to the more complex dynamics.

5.2.2 Foraging

Foraging consists of finding pucks in the environment, picking them up, and delivering them to the home region. An efficient implementation of foraging serves to validate our proposed behavior combination strategy. Foraging was tested on two different types of robots and environments, and its performance was repeatable and robust.

The shown implementation of foraging did not attempt to directly optimize the amount of time required to collect all of the pucks, although this criterion was indirectly minimized by diminishing interference between agents. Foraging was tested to validate that basic behavior sequencing was appropriate and robust, and that the higher–level task of collecting pucks was accomplished effectively. Figures 5-14, 5-15, and 5-16 demonstrate typical foraging performance by showing snapshots at different stages during the foraging process. Most foraging runs were terminated after 15 minutes, at which time about two thirds of the pucks were collected. The duration of the runs was largely due to the inefficient search strategy: the robots did not remember where the pucks were. An improved strategy, in which the robots remembered the location of the pucks and returned to it repeatedly until all of the pucks were transported, was used as a part of the group learning algorithm described in Chapter 8.

Not taking advantage of exact puck location was at least partially justified since, over the course of an experimental run, the pucks outside the home region were pushed around and gradually dispersed over an expanding area. This, in turn, affected the global behavior of the system, since the more dispersed the pucks became the more likely the robots were to stumble onto one of them by random search.

Puck dispersion was a side–effect, a result of the dynamics of interaction between the robots and the environment. It was also an influence on the dynamics since it affected the global behavior and performance of the system. Although a relatively simple effect, it would not be predicted by standard analytical models since it would

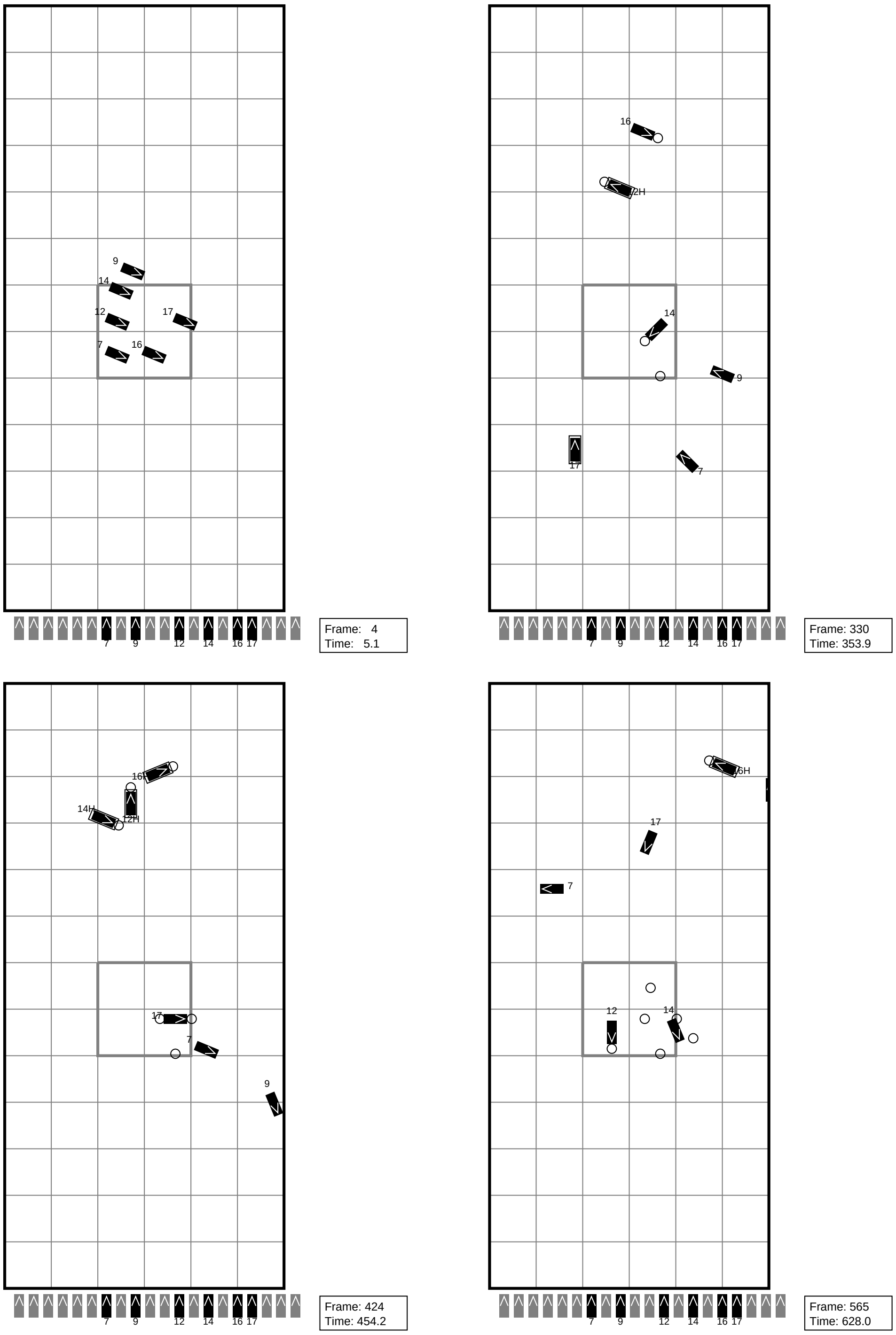


Figure 5-14: *Foraging* behavior of six robots. The robots are initiated in the home region. The pucks are initially clustered at the bottom center of the workspace. After *dispersing*, they *safe–wander* and search for pucks, pick them up, and take them *home*. If they encounter another robot with a puck while they are carrying one, they *follow*, as shown in the third frame of the data. After some time the pucks accumulate in the home region.

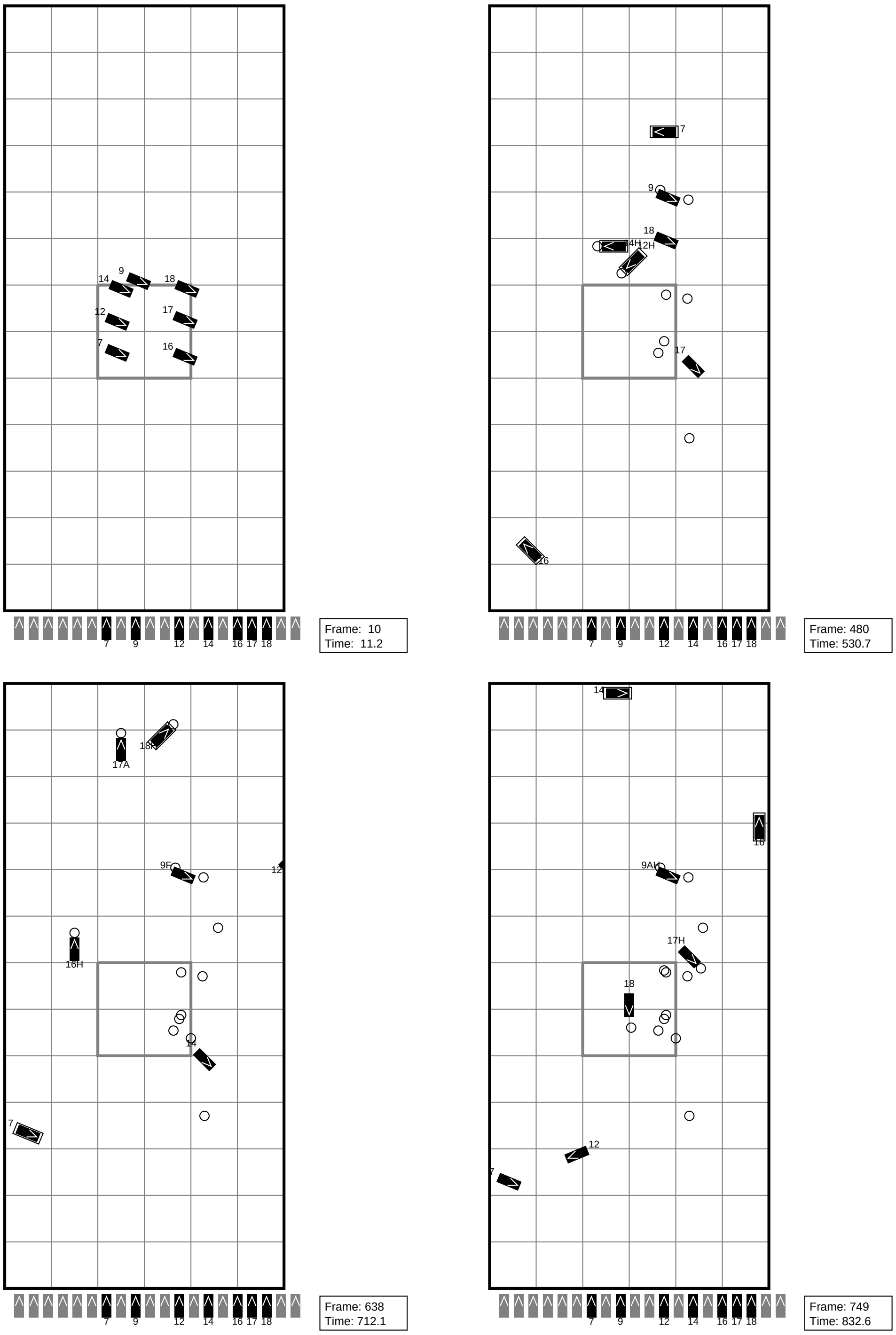


Figure 5-15: *Foraging* behavior of seven robots. In this experiment more robots effectively manage to transport a larger number of pucks home than the group of six robots shown above. Boxes around robots indicate they are executing avoidance in *safe–wandering* (e.g. see robots #14 and #16 in the last frame of the data, avoiding the walls of the workspace).

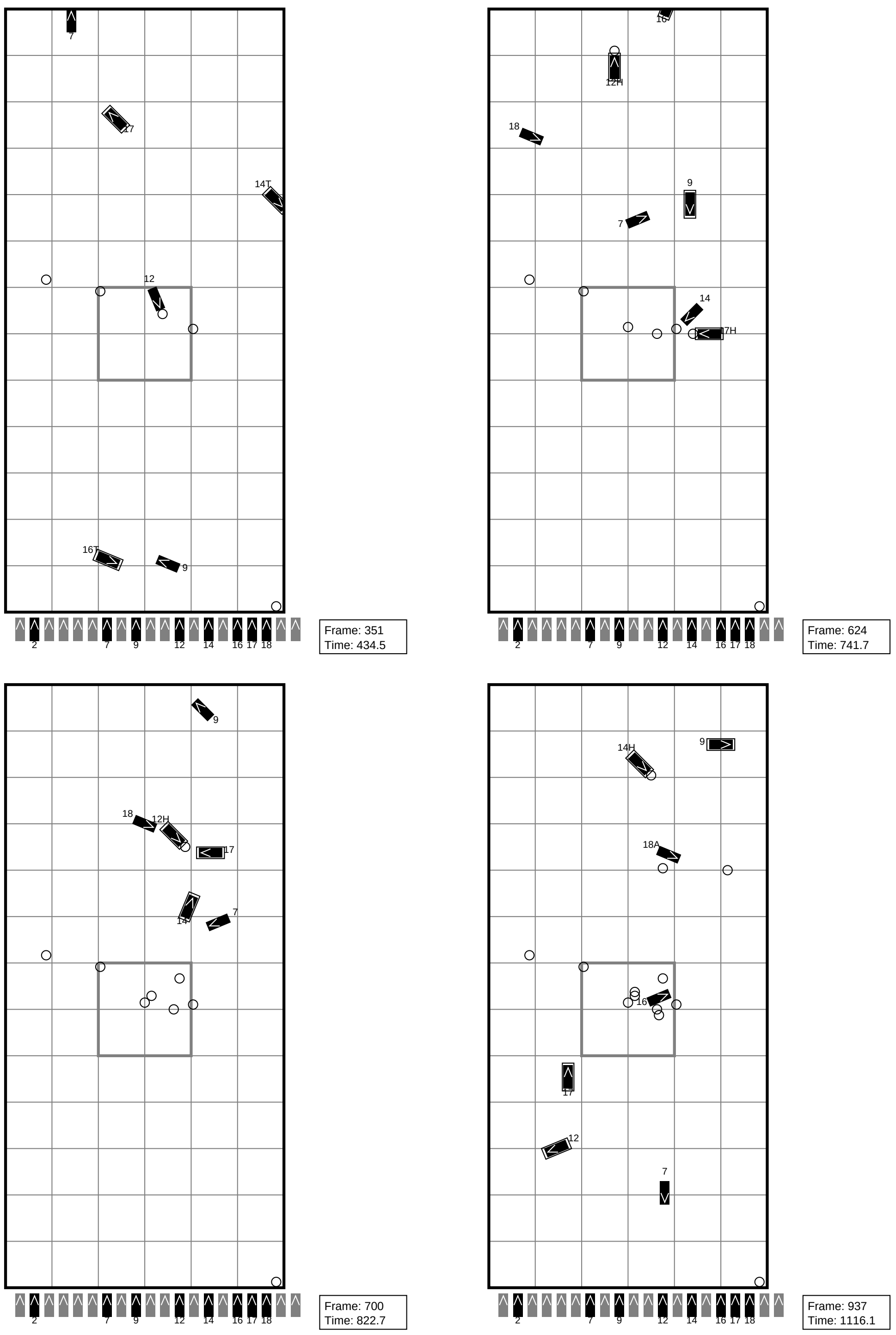


Figure 5-16: Another example of *foraging* behavior with seven robots. As before, the robots gather around the area with pucks (at the "top" of the workspace), picking them up, and gathering them in the home region. Interference is resolved by *safe–wandering* and *following*.

likely fall below the granularity of the model precision level.

In our system, foraging could be accomplished by a single agent, so the task itself does not require cooperation. Thus, the goal of the collective solution is to accelerate convergence with the growing size of the group. Arkin et al. (1993) describe simulation results of a similar task with varying numbers of agents and inter–agent communication. Complementary to the results presented here, they find that performance improves with simple communication. They also report an improvement of performance with growing group size up to a fixed point for the particular retrieval and gathering task. This result is in agreement with the results shown here that illustrate the interference effects of larger and thus higher–density groups in confined workspaces. Given the number of pucks to be collected, the collective solutions proposed here always outperformed a single agent, but as the group size grew, so did the importance of behaviors that minimized interference. This relationship will be further elaborated on in Chapter 8 which describes the approach to group learning.

5.2.3 Docking & Parking

This section gives another example of combining behaviors through the use of temporal switching and environmental constraints. Achieving arbitrary agent behaviors can be difficult as there is a often minimal match between the dynamics of the agent and its environment and the human–specified task. We now describe how *docking*, another group behavior that, if programmed top–down, would be difficult to achieve, can be simply generated by taking advantage of the system dynamics, and the interaction of simple basic behaviors.

Docking behavior "parks" the robots along some kind of a boundary. In general, getting a collection of robots to park along a line is difficult. A guaranteed solution can be found by geometric planning, but is intractable for uncertain, dynamic environments with multiple agents. In contrast to a tightly–controlled top down approach, we demonstrate the following a bottom–up alternative.

Our *docking* algorithm takes advantage of environmental constraints, i.e. the existence of other agents, the boundary, and the walls (see figure 5-17). The individual robot's goal is to keep moving safely and avoid collisions and drops. The collective goal is to achieve a state in which all of the robots are parked along the edge of a step, which they can detect using their downward–pointing IR sensors.

The algorithm consists of two behaviors:

- *safe–wandering* – keeps the robot moving forward and avoiding collisions,
- *avoiding-drops* – stops the robot from falling off an edge.

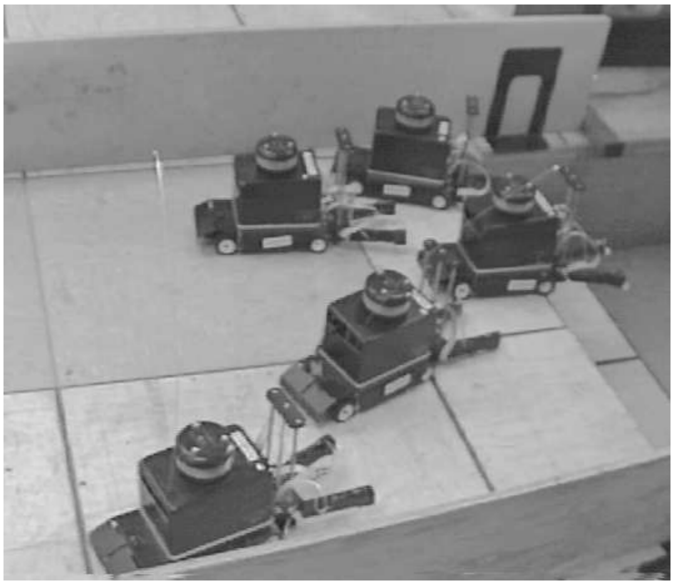


Figure 5-17: *Docking* behavior in progress, based on the constraints of the environment and three rules: avoid, go forward, and don't fall of the edge.

The behaviors are combined, in parallel, with avoiding behaviors taking precedence over wandering, as shown in Algorithm 5.2 below.

```
Dock:
If ground cannot be sensed
   stop.
If another robot is near by
   avoid.
If all is clear
   go forward.
```

Algorithm 5.2:

If the three behaviors are executed in a confined space with a vertical boundary, they will produce in a tight *docking* behavior. Since no position control is used, no specific docking positions are determined *a priori*. The algorithm is insensitive to initial conditions, to the number of robots, and to their avoidance strategies.

We tested the above algorithm in over 20 trials on groups from one to five robots. In all cases, it quickly resulted in all of the robots lined up along the edge, as shown in Figures 5-18 and 5-19.

Although we only explored the simplest case of *docking*, the environment con-

Figure 5-18: The end result of the *docking* behavior of five robots.

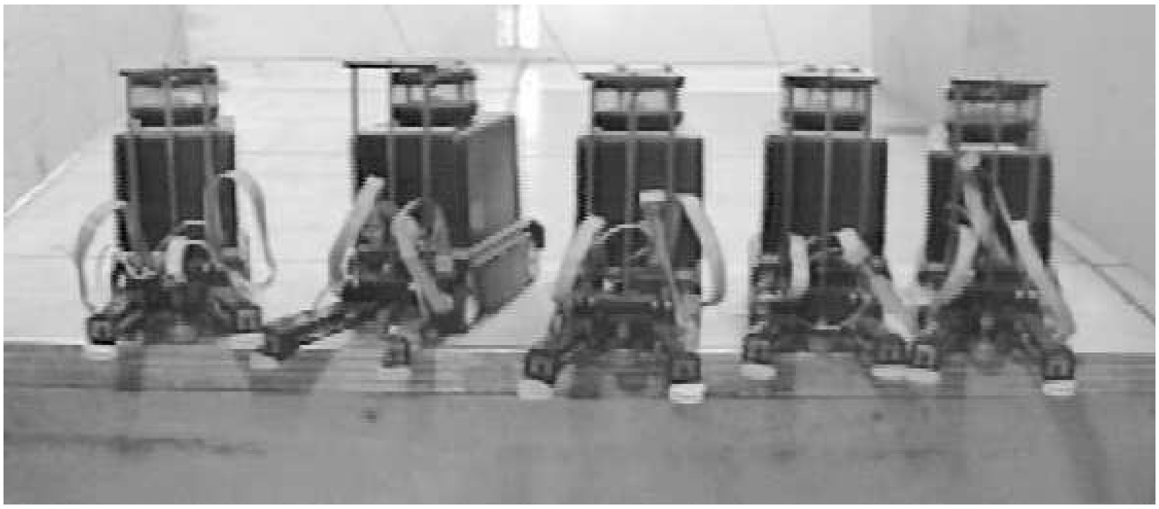


Figure 5-19: Another view of the *docking* robots.

straints can be eliminated if the robots use some position control. Similar simple behaviors and combinations have been used in other behavior–based systems. For instance, Matarić (1990*a*) uses three rules to achieve boundary following on a sonar–based mobile robot. Steels (1994*a*) implements docking onto a charger with two rules: one that approaches a light source above the charger, and another that avoids obstacles.

5.3 Combining Behaviors on Different Agents

This chapter has discussed ways of combining behaviors into higher–level composites, within a control system of a single agent. The described direct and temporal combination operators both rely on all of the agents' ability to respond to external conditions consistently. As long as all of the agents follow consistent social rules, i.e. use compatible social repertoires, conflict within and between agents is minimized. Homogeneity simplifies the task of combining behaviors, since the concern of conflict between behaviors is reduced by consistent social rules followed by all the agents.

However, although our agents are homogeneous in terms of their high–level goals, their immediate goals may differ at any point in time, i.e. they can be "locally heterogeneous." The arbitration of an encounter between two or more such agents is in fact analogous to the behavior selection problem at the level of control of a single agent. Consequently, similar strategies apply to the multi–agent case: the behaviors of the agents can be combined in some form, or one of the agents will take precedence over the rest.

As previously argued (Matarić 1992*a*), an unambiguous precedence hierarchy between the competing behaviors or agents is the simplest way to guarantee a globally consistent result. Thus, ensuring minimal higher–order effects and interference in a (locally) heterogeneous society can be accomplished by a strict hierarchy of control. This type of social organization appears quite stable and ubiquitous in animal and human societies. It often employs rather elaborate dominance structures requiring the maintenance of identities, distinguishing characteristics, and histories of previous encounters (McFarland 1987, Gould 1982), thus demanding higher cognitive overhead than the agents we have experimented with. As discussed in Section 4.5.2, such overhead may be necessary for certain types of complex, time–extended interactions.

5.4 Summary

This chapter has addressed methods for minimizing interference between behaviors on a single agent and behaviors between two or more interacting agents. A general architecture was introduced for combining a finite set of basic behaviors into an unbounded repertoire of higher–level behaviors based on direct and temporal combinations. The two types of combination operators used by the architecture were demonstrated on compound spatial behaviors of *flocking*, *foraging*, and *docking*, implemented and tested on the collection of mobile robots.

The next chapter introduces a methodology for automating the behavior combination process through the use of learning.

Chapter 6

Learning in Situated Systems

So far we have dealt with the problem of synthesizing intelligent group behavior by hand. We now extend the presented ideas to include learning, an ability that allows the agent to acquire new and adapt old behaviors for individual and group benefit.

6.1 Motivation

Why learn?
Learning has two purposes universal across domains. It is useful for:

1. adapting to external and internal changes
2. simplifying built–in knowledge

The ability to cope with changes in the environment is termed *adaptability*. It allows agents to deal with noise in their internal and external sensors, and with inconsistencies in the behavior of the environment and other agents. Adaptability comes at a phenotypical and cognitive cost, so creatures are adapted only to a specific niche. Consequently, all creatures, natural and otherwise, fail at their tasks under certain conditions. The purpose of learning is to make the set of such conditions smaller.

Adaptability does not necessitate learning. Many species are genetically equipped with elaborate "knowledge" and abilities, from the very specific, such as the ability to record and utilize celestial maps (Waterman 1989), to the very general, such as plasticity in learning motor control (McFarland 1987) and language (Pinker 1994). But genetic code is finite. In fact, primate and human cortical neural topology is too complicated to fully specify in the available genome, and is instead established by

Problem	Learning in complex situated domains.
Assertion	Traditional reinforcement learning must be reformulated.
Approach	Replace states, actions and reinforcement with conditions, behaviors, heterogeneous reward functions and progress estimators.
Validation	Implement learning on a group of mobile robots learning to forage.

Table 6.1: A summary of the situated learning problem addressed here, and the structure of the proposed solution.

spontaneous synaptic firing *in utero* and in the first decade of life (Vander et al. 1980). In addition to compensating for genetic parsimony, learning is useful for optimizing the agent's existing abilities, and necessary for coping with complex and changeable worlds. It is often argued that societies exist largely for conservation and propagation of behavior strategies too complex to be passed on genetically.

The answer to the built–in versus learned tradeoff varies across species and environments. The work described here addresses this fundamental tradeoff in the domain of situated multi–agent systems.

The rest of the thesis will address the following problem: how can a collection of situated agents learn in a group environment? This problem will be addressed in a nondeterministic, noisy and error–prone domain with stochastic dynamics, in which the agent does not have an *a priori* model of the world.

We propose a formulation of reinforcement learning that uses a level of description that makes the state space manageable, thus making learning possible. Furthermore, we present two methods for shaping reinforcement to take advantage of information readily available to the agent, and to make learning more efficient. These ideas are validated by demonstrating an effective learning algorithm on a group of robots learning to forage. Table 6.1 summarizes the problem and the approach.

6.2 Relevant Learning Models

There are many things an agent can learn, but not many ways in which it can learn it. According to what is being learned, existing approaches can be classified into the following categories:

- learning declarative knowledge
- learning control
- learning new behaviors
- learning to select behaviors/actions

6.2.1 Learning Declarative Knowledge

Learning declarative knowledge is one of the founding areas of AI but also one that is least directly related to the work in this thesis. The only type of declarative knowledge that situated agents have had to deal with to date are maps of the environment. Much of the robotics literature deals with the problem of constructing and updating such maps in variety of situated domains (see Matarić (1990*a*) for a review of the literature). Maps and world models are closely tied to action in the world, which is why they are the primary type of declarative knowledge so far used in situated agents[1]. In contrast, this thesis focuses on procedural knowledge that is directly tied to acting and interacting in the world. The remaining learning categories are directly tied to action[2].

6.2.2 Learning Control

Learning control is a growing field based on *adaptive control*, a branch of control theory. Problems in adaptive control deal with learning the forward or inverse model of the system, i.e., the plant. Forward models provide predictions about the output expected after performing an action in a given state. Analogously, inverse models provide an action, given the current state and a desired output (Jordan & Rumelhart 1992). Learning control has been applied to a variety of domains and has used a number of different learning methodologies. Connectionist algorithms are most popular, (see Miller, Sutton & Werbos (1990) for a representative collection), but

[1]Note: not all maps are explicit and declarative. See Matarić (1990*a*) for examples.

[2]Author's bias: declarative learning can be further divided into as many interesting categories, but is not the area pursued here.

other approaches have also been studied (e.g., Atkeson, Aboaf, McIntyre & Reinkensmeyer (1988), Atkeson (1990), Schaal & Atkeson (1994)). Adaptive control problems typically deal with learning complex dynamical systems with non–linearly coupled degrees of freedom usually involved in moving multi–jointed manipulators, objects, and physical bodies.

6.2.3 Learning New Behaviors

Learning new behaviors deals with the problem of acquiring strategies for achieving particular goals. Because the notion of behavior is not well defined, neither is the behavior learning problem.

We defined behavior to be a control law with a particular goal, such as *wall–following* or *collision avoidance*. The definition is general and meant to refer to a level of description above basic control without specifying what that level is, since it varies with the domain. Furthermore, the concept of behavior contains informal notions about generality and adaptivity that are difficult to state precisely without domain–specific grounding.

Consequently, most learning control problems appear to be instances of behavior learning, such as learning to balance a pole (Barto, Sutton & Anderson 1983), to play billiards (Moore 1992), and to juggle (Schaal & Atkeson 1994). Furthermore, work on action selection, deciding what action to make in each state, can be viewed as learning a higher–level behavior as an abstraction on the state–action space. For example, a maze–learning system can be said to learn a specific *maze–solving* behavior.

Genetic learning has also addressed learning behaviors in simulated worlds (Koza 1990). Since learning behaviors requires finding appropriate parameter settings for control, it can be cast as an optimization problem, for which genetic algorithms are particularly well suited (Goldberg 1989). However, since genetic algorithms operate on an abstract encoding of the learning problem, the encoding requires a good model of the agent and the environment in order to generate useful behaviors. Since the problem of modeling situated worlds is notoriously difficult, only a few genetic algorithms have produced behaviors that successfully transferred to physical systems (Steels 1994*b*, Cliff, Husbands & Harvey 1993, Gallagher & Beer 1993).

However, none of the above learning approaches can be said to learn new behaviors according to the precise definition of the problem. The posed "behavior learning problem" (Brooks & Matarić 1993) requires that the agent acquire a new behavior using its own perceptual and effector systems, as well as to assign some semantic label to the behavior, in order to later recognize and use it as a coherent and independent unit. Behavior learning appears to require bridging the elusive signal–to–symbol gap, even for the most limited notion of "symbol."

Given this definition, no existing work performs behavior learning. Learning control and learning action selection are not strictly instances of behavior learning because in both cases, by definition, only a single behavior is learned and no further abstraction is performed. Similarly, genetic algorithms do not address the stated behavior learning problem either, because in their domain the semantics are also provided by the designer.

The signal–to–symbol problem is one of the hallmark challenges in AI. Because it bridges a gap between two already estranged communities, it has not received much attention. Another challenge of the problem is setting it up to avoid biasing the learner inappropriately, but still be able to evaluate its performance. It is unlikely that "behaviors", "concepts", and "symbolic representations" that are automatically generated by a situated agent will map neatly from the agent's sensorium into the human observer's semantic space. Nonetheless, the situated domain is particularly well suited for this type of work as it allows for grounding the agents' learning in physical behavior that is observable and thus can be evaluated externally from its mechanism and representation.

6.2.4 Learning to Select Behaviors

If learning new behaviors is learning *how* to do something, then learning to select behaviors is learning *when* to do it. Behavior selection has not been extensively studied so far, largely due to the lack of formalization of "behavior" as a building block for control. The work that has been done on the topic has used reinforcement learning techniques (e.g., Maes & Brooks (1990) and Maes (1991)). Learning behavior selection is by definition a *reinforcement learning problem* as it is based on correlating the behaviors the agent performs and the feedback it receives as a result.

6.3 Reinforcement Learning

Reinforcement learning (RL) is a class of learning methodologies in which the agent learns based on external feedback received from the environment. The feedback is interpreted as positive or negative scalar reinforcement. The goal of the learning system is to maximize positive reinforcement (reward) and/or minimize negative reinforcement (punishment) over time. Traditionally, the learner is given no explicit built–in knowledge about the task. If the learner receives no direct instruction or answers from the environment the learning is considered *unsupervised* (Barto 1990). The learner produces a mapping of states to actions called a *policy*.

Reinforcement learning originated in Ivan Pavlov's classical conditioning experiments (Gleitman 1981). Embraced by behaviorism, stimulus–response learning became the predominant methodology for studying animal behavior in psychology and biology. Ethology, the study of animals in their natural habitats, developed in response to the tightly controlled laboratory experimental conditions commonly used by behaviorists. In the mean time, RL was adopted and adapted by the computational community, and applied to various machine learning problems.

Maze–learning was formulated as a reinforcement learning problem based on reward and punishment in the first well known application of RL (Minsky 1954). Soon thereafter, the problem of learning a scoring functions for playing checkers was successfully addressed with an RL algorithm (Samuel 1959). Subsequently, RL was applied to a variety of domains and problems, most notably in the Bucket Brigade algorithm used in Classifier Systems (Holland 1985), and in a class of learning methods based on Temporal Differencing (Sutton 1988). Reinforcement learning has been implemented with a variety of algorithms ranging from table–lookup to neural networks, and on a broad spectrum of applications, including tuning parameters and playing backgammon.

Our work is concerned with reinforcement learning on situated, embodied agents. In particular, it is focused on issues that arise when traditional models of RL, and algorithms applied to those models, are used in the complex multi–agent domain we are working with. To address these issues, we begin by describing the most commonly, but not exclusively, used RL model.

6.3.1 Markov Decision Process Models

Most computational models of reinforcement learning are based on the assumption that the agent–environment interaction can be modeled as a Markov Decision Process (MDP), as defined below:

1. The agent and the environment can be modeled as synchronized finite state automata.

2. The agent and the environment interact in discrete time intervals.

3. The agent can sense the state of the environment and use it to make actions.

4. After the agent acts, the environment makes a transition to a new state.

5. The agent receives a reward after performing an action.

While many interesting learning domains can be modeled as MDPs, situated agents learning in nondeterministic, uncertain environments do not fit this model. The next section describes the reasons why, by addressing each of the model assumptions in turn.

6.3.2 State

Most RL models are based on the assumption that the agent and the environment are always in a clearly–defined state that the agent can sense. In situated domains, however, the world is not readily prelabeled into appropriate states, and the world state is not readily and consistently accessible to the agent. Instead, the world is continuous and partially observable.

Continuity

The state of a situated agent consists of a collection of properties, some of which are discrete, such as the inputs from binary sensors, others continuous, like the velocities of wheels. Even for the simplest of agents, a monolithic descriptor of all state properties is prohibitively large. It scales poorly with increased sensory capabilities and agent complexity in general, and results in a combinatorial explosion in standard RL.

Most models to date have bypassed continuous state by presuming higher–level sensory operators such as "I see a chair in front of me." But such operators have been shown to be unrealistic and largely unimplementable in systems using physical sensors (Agre & Chapman 1990, Brooks & Matarić 1993). In general, the problem of partitioning continuous state into discrete states is hard (Košecká 1992), and even if a reasonable partitioning of the world is found, there may be no mapping from the space of sensor readings to this partitioning.

Observability

Although continuous and often complex, sensors have limited abilities. Instead of providing descriptions of the world, they return simple properties such as presence of and distance to objects within a fixed sensing region. Consequently, they cannot distinguish between all potentially relevant world states. The collapse of multiple states into one results in partial observability, i.e. in **perceptual aliasing**, a many–to–one mapping between world and internal states. The inability to distinguish different states makes it difficult and often impossible for the learning algorithm to assign appropriate utility to actions associated with such states (Whitehead & Ballard 1990).

Partially Observable Markov Decision Processes (POMDPs) have been developed by the operation research community for dealing with this problem. Partial observability is added into a Markov model by introducing a discrete probability distribution over a set of possible observations for a given state. POMDPs have been studied and successfully applied to theoretical learners (Cassandra, Kaelbling & Littman 1994), but have not yet been used empirically largely due to the fact that observability models of situated systems are not generally available.

Generalization

Any learner is caught in a paradox: it must disambiguate the relevant inputs, but it also must discard all irrelevant inputs in order to minimize its search space. However it may be structured, the learner's space in traditional RL is exponential in the size of the input, and thus marred by the curse of dimensionality (Bellman 1957). Some form of **input generalization**, or collapsing of states into functional equivalence classes, is necessary for almost all problems.

Human programmers perform generalization implicitly whenever they use clever orderings of rules, careful arbitration, and default conditions, in crafting control strategies. They minimize ambiguity and maximize parsimony by taking advantage of their domain knowledge.

In RL, in the absence of domain knowledge, state generalization has been addressed with statistical clustering methods using recursive partitioning of the state space based on individual bit relevance (Chapman & Kaelbling 1991, Mahadevan & Connell 1991*a*, Moore 1991, Moore 1993). It is also confronted in Classifier Systems that use binary strings as state descriptors (Holland 1986). The state can contain wild cards (#'s) that allow for clustering states, with the flexible grouping potential of full generality (all #'s) to full specificity (no-#'s). Generalization results in so-called "default hierarchies" based on the relevance of individual bits changed from #'s to specific values. This process is analogous to statistical RL methods (Matarić 1991).

The input generalization problem is also addressed by the connectionist RL literature. Multi–layer networks have been trained on a variety of learning problems in which the hidden layers constructed a generalized intermediate representation of the inputs (Hinton 1990). While all of the RL generalization techniques are non–semantic, the table–based methods and Classifier System approaches are somewhat more readable as their results are a direct consequence of explicit hand–coded criteria. Connectionist approaches, in contrast, utilize potentially complex network dynamics and produce effective but largely inscrutable generalizations.

All of the described generalization techniques are effective but require large numbers of trials to obtain sufficient statistical information for clustering states. As such, they are an incremental improvement of the overwhelmingly slow exponential learning algorithms. Our work will explore a different alternative, one that takes principled advantage of domain knowledge instead of purely statistical generalization.

Paradoxically, the unwieldy fully–exponential[3] state–action search space used by standard RL models gives them one of their main positive properties: asymptotic completeness. While hand coded reactive policies take advantage of the cleverness of the designer, they are rarely provably complete. Most irrelevant input states are easily eliminated, but potentially useful ones can be overlooked. On the other hand, complete state spaces guarantee that, given sufficient time and sufficiently rich reinforcement, the agent will produce a provably complete policy. Unfortunately, this quality is of little use in time–bounded situated domains.

6.3.3 State Transitions

Simple MDP–based models employ discrete, synchronized state transitions. In contrast, in situated domains the world state and the agent state change asynchronously in response to various events. In dynamic domains, only a subset of those events are directly caused by the agent's actions or are in agent's control. In general, events can take different amounts of time to execute, can have delayed effects, and can have different consequences under identical conditions. In short, situated domains are difficult to model properly.

Deterministic models do not capture the dynamics of most situated domains, so nondeterministic alternatives have been considered (Lin 1991). Unfortunately, most are based on unrealistic models of sensor and effector uncertainty with overly simplified error properties. They are typically based on adding Gaussian noise to each sensed state and each commanded action. However, uncertainty in situated domains does not follow Gaussian distributions but instead results from structured dynamics of interaction of the system and the environment. These dynamics play an important role in the overall behavior of the system, but are generally at a description level too low to be accurately modeled or simulated.

As an example, consider the properties of realistic proximity and distance sensors. The accuracy of ultrasound sensors is largely dependent on the incident angle of the sonar beam and the surface, as well as on the surface materials, both of which are difficult and tedious to model accurately. Infra–red and vision sensors also have similarly detailed yet entirely different properties, none of which are accurately represented with simple models. Simple noise models are tempting, but they produce artificial dynamics that, while potentially complex, do not model the true complexity

[3] In the number of state bits.

of realistic physical systems. Consequently, many elegant results of simple simulations have not been successfully repeated on more complex agents and environments.

Given the challenges of realistic modeling, it is generally very difficult to obtain transition probabilities for nondeterministic models of situated domains. Models for such domains are not readily available, and must be obtained empirically for each system by a process analogous to learning a world model. It is difficult to estimate if obtaining a world model for a given domain requires any more or less time than learning a policy for some set of goals. Consequently, insightful work on learning world models for more intelligent exploration (Sutton 1990, Kaelbling 1990) is yet to be made applicable to complex situated domains.

We have argued that accurate models of situated domains are difficult to obtain or learn. Instead, we will focus in this work on learning policies in systems without explicit world models. The next section describes the general form of RL algorithms that have been used for such policy learning.

6.3.4 Algorithms

Reinforcement learning algorithms have the following general form (Kaelbling 1990):

1. Initialize the learner's internal state I to I_0.
2. Do Forever:
   a. Observe the current world state s.
   b. Choose an action $a = F(I, s)$
   using the evaluation function F.
   c. Execute action a.
   d. Let r be the immediate reward for
   executing a in world state s.
   e. Update the internal state $I = U(I, s, a, r)$
   using the update function U.

The internal state I encodes the information the learning algorithm saves about the world, usually in the form of a table maintaining state and action data. The update function U adjusts the current state based on the received reinforcement, and maps the current internal state, input, action, and reinforcement into a new internal state. The evaluation function F maps an internal state and an input into an action based on the information stored in the internal state. Different RL algorithms vary in their definitions of U and F.

The predominant methodology used in RL is based on a class of *temporal differencing* (TD) techniques (Sutton 1988). All TD methods deal with assigning credit or blame to past actions by attempting to predict long–term consequences of each action in each state. Sutton's original formalization of temporal differencing (TD(λ)) deals with such predictions in Markovian environments, and covers a large class of learning approaches. For example, Bucket Brigade, the delayed reinforcement learning method used in Classifier Systems, is an instance of TD (Matarić 1991). Q-learning (Watkins 1989), the most commonly known and used TD algorithm, is defined and explained in Appendix A, as background for subsequent comparison.

6.3.5 Learning Trials

Performance properties of various forms of TD applied to Markovian environments have been extensively studied (Watkins & Dayan 1992, Barto, Bradtke & Singh 1993, Jaakkola & Jordan 1993). Provable convergence of TD and related learning strategies based on dynamic programming is asymptotic and requires infinite trials (Watkins 1989). Generating a complete policy, however incorrect, requires time exponential in the size of the state space, and the optimality of that policy converges in the limit as the number of trials approaches infinity. In practice, this translates into hundreds of thousands of trials for up to ten–bit states. Thus, even in ideal Markovian worlds the number of trials required for learning is prohibitive for all but the smallest state spaces.

The situated learning problem is even more difficult, however. Assuming an appropriately minimized state space, a learner may still fail to converge, due to insufficient reinforcement.

6.3.6 Reinforcement

Temporal credit assignment, assigning delayed reward or punishment, is considered to be one of the most difficult and important problems in reinforcement learning[4]. Temporal credit is assigned by propagating the reward back to the appropriate previous state–action pairs. Temporal differencing methods are based on predicting the expected value of future rewards for a given state–action pair, and assigning credit locally based on the difference between successive predictions (Sutton 1988).

Reward functions determine how credit is assigned. The design of these functions is not often discussed, although it is perhaps the most difficult aspect of setting up

[4]The first statement of the problem is due to Samuel (1959), whose checkers–learning program learned to reward moves that eventually lead to "a triple jump."

a reinforcement learning algorithm. The more delayed the reward, the more trials the learning algorithm requires, the longer it takes to converge. Algorithms using immediate reinforcement naturally learn the fastest.

Most reinforcement learning work to date has used one of the following two types of reward: immediate or very delayed. We postulate, however, that situated domains tend to fall in between the two popular extremes, providing some immediate rewards, plenty of intermittent ones, and a few very delayed ones. Although delayed reinforcement, and particularly *impulse reinforcement* that is delivered only at the single goal, eliminates the possibility for biasing the learning, it usually makes it prohibitively difficult. Most situated learning problems do not resemble mazes in which the reward is only found at the end. Instead, some estimates of progress are available along the way. These estimate can be intermittent, internally biased, inconsistent, and occasionally incorrect, but if used appropriately, can significantly speed up learning. The approach presented in the next chapter takes advantage of such intermediate estimates to shape reinforcement and accelerate learning.

6.3.7 Multiple Goals

We have argued that impulse reinforcement related to a single goal makes learning prohibitively slow. Furthermore, single goal agents are rare in situated domains. Instead, situated agents are best viewed as having multiple goals, some of which are maintained concurrently, while others are achieved sequentially. For example, in our previously described foraging task, an agent maintains a continuous low–level goal of collision avoidance, also keeps a minimal distance from other agent in order to minimize interference, may attempt to remain in a flock, and may be heading home with a puck.

Most RL models require that the learning problem be phrased as a search for a single goal optimal policy, so that it can be specified with a global reward function. Not surprisingly, if the world or the goal changes, a new policy must be learned, using a new reward function. The existing policy will conflict with the new learning and will need to be "forgotten."

In order to enable learning of a multi–goal policy, the goals must be formulated as subgoals of a higher–level single optimal policy. Therefore they must be sequential and consistent. To enforce a specific goal sequence, the state space must explicitly encode what goals have been reached at any point in time, thus requiring added bits in the input state vector (Singh 1991). Although a natural extension of the RL framework, this method requires the state space to grow with each added goal, and cannot address concurrent goals. Sequences of goals fail to capture the dynamics of complex situated worlds and agents that may have one or more high–level goals of achievement, and also a number of maintenance goals, the interaction of which has important effects on the agents' behavior and rate of learning.

A more general solution to multiple goals within the traditional framework is to use separate state spaces and reinforcement functions for each of the goals and merge them Whitehead, Karlsson & Tenenberg (1993). However, merging policies assumes that the necessary information for utility evaluation is available to the agent. However, as previously discussed in relation to game–theoretic approaches (see Section 2.4.7), that assumption may not hold in many situated domains.

6.3.8 Related Work

Work in computational RL has been active since the fifties and has become particularly lively in the last decade. The majority of the contributions have been theoretical in nature. For thorough reviews of reinforcement learning as applied to well–behaved learning problems see Watkins (1989) and Sutton (1988). For more recent work on improved learning algorithms for situated agents, largely applied to simulated domains, see Kaelbling (1990) and Whitehead (1992). This section will focus on empirical learning work with situated agents.

Whitehead & Ballard (1990) and Whitehead (1992) addressed the perceptual aliasing problem in situated RL. They proposed an approach to adaptive active perception and action that divided the control problem into two stages: a state identification stage and a control stage, and applied appropriate learning methods to each. The approach was demonstrated on a simulated block stacking task, but has not been tested in an embodied domain.

Kaelbling (1990) used a simple mobile robot to validate several RL algorithms using immediate and delayed reinforcement applied to learning obstacle avoidance.

Maes & Brooks (1990) applied a statistical reinforcement learning technique using immediate reward and punishment in order to learn behavior selection for walking on a six–legged robot. The approach was appropriate given the appropriately reduced size of the learning space and the available immediate and accurate reinforcement derived from a contact sensor on the belly of the robot, and a wheel for estimating walking progress.

More delayed reinforcement was used by Mahadevan & Connell (1991*a*) in a box–pushing task implemented on a mobile robot, in which subgoals were introduced to provide more immediate reward. Mahadevan & Connell (1991*b*) experimented with Q–learning using monolithic and partitioned goal functions for learning box–pushing, and found subgoals necessary.

Chapman & Kaelbling (1991) and Mahadevan & Connell (1991a) demonstrated complementary approaches for generalization. Chapman & Kaelbling (1991) started with a single most general state and iteratively split it based on statistics accumulated over time. Splitting is based on the relevance of each state bit; when one is found to be relevant, the state space is split in two, one with that bit on, and the other with it off. In contrast, Mahadevan & Connell (1991a) started with a fully differentiated specific set of states, and consolidated them based on similarity statistics accumulated over time.

Aside from traditional unsupervised reinforcement learning methods described above, other techniques have also been explored. Pomerleau (1992) used a supervised connectionist learning approach to train steering control in an autonomous vehicle based on generalizing visual snapshots of the road ahead.

Thrun & Mitchell (1993) demonstrated a connectionist approach to learning visual features with a camera mounted on a mobile robot. The features are not assigned by the designer but are instead selected by the network's intermediate representations. Not surprisingly, the result is not semantically meaningful to a human observer, but is nonetheless well suited for the robot's navigation task.

The work presented here is, to the best of the author's knowledge, the first attempt at applying reinforcement learning to a collection of physical robots learning a complex task consisting of multiple goals. Parker (1994) implemented a non–RL memory–based style of parameter–learning for adjusting activation thresholds used to perform task allocation in a multi–robot system. Tan (1993) has applied traditional RL to a simulated multi–agent domain. Due to the simplicity of the simulated environment, the work has relied on an MDP model that was not applicable to this domain. Furthermore, Tan (1993) and other simulation work that uses communication between agents relies on the assumption that agents can correctly exchange learned information. This often does not hold true on physical systems whose noise and uncertainty properties extend to the communication channels.

6.4 Summary

This chapter has overviewed the key properties of reinforcement learning strategies based on Markov Decision Process models, and their implications on learning in situated domains. Learning algorithms based on dynamic programming and traditionally applied to such Markovian domains were also discussed. Finally, related robot learning and reinforcement learning work was reviewed.

Two main problems arise when the standard MDP formulation is applied to our multi–agent domain: 1) the state space is prohibitively large, and 2) delayed reinforcement is insufficient for learning the foraging task. The next chapter introduces a method of reformulating the learning problem in order to make learning both possible and efficient in the complex domain used in this work.

Chapter 7

The Learning Approach

This chapter describes a formulation of the proposed reinforcement learning problem in order to make learning possible and efficient in the complex situated domain at hand, as well as in situated domains in general.

In order to deal with the complexity and uncertainty of situated domains, a learning algorithm must use an appropriate level of description. A learner using too low a level of description will result in a state space so large as to make the learning prohibitively slow. In contrast, a learner based on too corse a level of description cannot discover any novel and potentially useful strategies outside the structured space allowed by the coarse representation.

An appropriate representation shapes the state space into an expressive but tractable learning space. An effective learning algorithm, then, searches this learning space efficiently. Thus, given the complexities of situated agents and environments, as well as those of reinforcement learning algorithms, any approach to situated learning should have the following properties.

A model for situated learning should:

1. minimize the learner's state space
2. maximize learning at each trial

This chapter will address each of the desired properties in turn. First, an approach will be described for minimizing the state space in order to make the learning problem tractable. Second, an approach for shaping reinforcement will be proposed that makes learning more efficient. In both cases, the traditional primitives of reinforcement learning (states, actions, and reinforcement) will be reformulated ($\longrightarrow$) into subtly different but pragmatically more effective counterparts, as follows:

1. states & actions $\longrightarrow$ conditions & behaviors
2. reinforcement $\longrightarrow$ multi–modal feedback

7.1 Reformulating the Problem

Traditional state–action models used by many RL approaches tend to be based on a level of description inappropriate for complex situated domains. Their representations abstract away important control details, but still must search excessively large state spaces representing the agent's entire world. A large state space is not so much a sign of a difficult problem as it is of a poorly formulated one. We propose the following reformulation that uses a more appropriate representation for the problem of learning in noisy and inconsistent worlds:

> *Reinforcement learning in situated domains can be formulated as learning the conditions necessary and sufficient for activating each of the behaviors in the repertoire such that the agent's behavior over time maximizes received reward.*

This formulation accomplishes the desired goal of diminishing the learning space by using conditions and behaviors instead of states and actions, with the effect of elevating the level of description of the learning problem.

7.1.1 Behaviors

The first part of the thesis has argued that behaviors are an intuitive and effective level of description for control, and described a methodology for selecting and combining basic behaviors for a given domain and set of goals. *Behaviors* were defined as goal–driven control laws that hide the details of control. The same reasons that made behaviors a useful abstraction in control make them an appropriate and efficient basis for learning.

Behaviors are more general than actions because they are not tied to specific detailed states but instead triggered by a set of general conditions. For instance, a *wall-following* behavior applies to any environment and any wall that the agent can sense, and is not dependent on the agent's exact state including such information as its (x, y) position, whether it is carrying a puck, and what is in front or behind it.

It can be said that much of the RL literature already uses behaviors without labeling them as such. For example, an action called "left" which transports an agent to the next square on a grid and turns it by 90 degrees, requires a complex control sequence. It is a control law that guarantees an output, such as the agent's position and orientation, and is thus identical in effect to our definition of behavior. Such a behavior, however, may not be realistic in continuous, noisy domains. In general, atomic actions of simulated grid worlds can translate into arbitrarily complex behaviors on embodied systems. Consequently, situated, embodied agents often use a very different set of behavior primitives, specifically designed for the particular dynamics of the agent and its interaction with the world.

Behaviors elevate control to a higher and more realizable level. However, the complexity of reinforcement learning lies in the size of the learning space, which is traditionally exponential in the state space of the agent. In order to significantly accelerate learning, we must minimize this space as well. We propose to do so by abstracting the learning space to a higher level, structured by the granularity of the conditions necessary for executing each of the behaviors.

Using behaviors abstracts away the details of the low–level controller, while still using realizable units of control, and thus guaranteeing the results, or postconditions, of each behavior. Similarly, conditions abstract away the low–level details of the agent's state space, and define the learning space at a higher level, by state clustering.

7.1.2 Conditions

Conditions are predicates on sensor readings that map into a proper subset of the state space. Each condition is defined as the part of the state that is necessary and sufficient for activating a particular behavior. For instance, the necessary and sufficient conditions for picking up a puck are that a puck is between the fingers of the robot.

The space of conditions is usually much smaller than the complete state space of the agent, resulting in a smaller space for the learning algorithm. Furthermore, the fewer state elements need to be sensed the less the system will suffer from error and uncertainty. Finally, the only events relevant to the agent are those that change the truth value of the predicates, i.e. the current condition. Those events are used to trigger and terminate behaviors.

Reformulating states and actions into conditions and behaviors effectively reduces the state space to a manageable size, thus making learning possible in a complex domain. The next step is to make learning efficient, by using appropriate reinforcement.

7.2 Reinforcement for Accelerated Learning

The amount and quality of the reinforcement determines how quickly the agent will learn. In nondeterministic uncertain worlds, learning in bounded time requires shaping of the reinforcement in order to take advantage of as much information as is available to the agent.

In general, reinforcement learning can be accelerated in two ways: 1) by building–in more information, and 2) by providing more reinforcement. The reward function implicitly encodes domain knowledge and thus biases what the agent can learn. Simplifying and minimizing reinforcement, as practiced by some early RL algorithms (Sutton 1990), does diminish this bias, but it also greatly handicaps, and in situated domains, completely debilitates the learner.

Domain knowledge can be embedded through a reward–rich and complex reinforcement function. This approach is effective, but the process of embedding semantics about the world into the reward function is usually *ad hoc*. In the ideal case, reinforcement is both immediate and meaningful. Immediate error signals that provide not only the sign but also the magnitude of the error result in fastest learning. As in *supervised learning*, then provide the agent with the correct answer after each trial. In learning control (Jordan & Rumelhart 1992, Atkeson 1990, Schaal & Atkeson 1994), such error signals are critical as the learning problem is usually finding a complex mapping between a collection of input parameters and the desired output. Immediate reinforcement in RL is typically a weak version of an error signal, reduced to only the sign of the error but not the magnitude or the direction.

We propose an intermediate solution based on *shaping* as a version of an error signal based on principled embedding of domain knowledge.

7.2.1 Heterogeneous Reward Functions

Monolithic reward functions with a single high–level goal, when applied to situated domains, require a large amount of intermediate reinforcement in order to aid the agent in learning. Intuitively, the more subgoals are used the more frequently reinforcement can be applied, and the faster the learner will converge. We have already argued that situated agents maintain multiple concurrent goals, and that such goals can be achieved and maintained by using behaviors as the basic unit of control and learning. Thus, a task in a situated domain can be represented with a collection of such concurrent goal–achieving behaviors. Reaching each of the goals generates an event[1] that provides primary reinforcement to the learner. The following is the

[1] A change in the conditions.

general form of such event–driven reinforcement functions:

$$R_e(c,t) = \begin{cases} r & \text{if the event E occurs} \\ 0 & \text{otherwise} \end{cases} \quad e \neq 0$$

Event–driven reinforcement for any event E is a function of conditions c and time t. The received reinforcement r may be positive or negative.

If necessary information about the task and the appropriate sensors are available, each of the goals can be further broken down into one or more subgoals, with associated secondary reinforcement. In general, the specification of a high–level behavior provides a collection of subgoals that need to be achieved and maintained. If the achievement of a subgoal can be detected, it can be directly translated into a reinforcement function.

A general heterogeneous reward function has the following form:

$$R_e(c,t) = \begin{cases} r_{E1} & \text{if event E1 occurs} \\ r_{E2} & \text{if event E2 occurs} \\ . & . \\ . & . \\ . & . \\ r_{En} & \text{if event En occurs} \\ 0 & \text{otherwise} \end{cases}$$

The complete reward function is a sum of inputs from the individual event–driven functions. Thus, if multiple events occur simultaneously, appropriate reinforcement for all of them is received from multiple sources.

Even–driven reinforcement functions are illustrated with the following example:

- A robot receives reward R_a whenever it avoids an obstacle, and reward R_h whenever it reaches home.
- The corresponding reward function appears as follows:

$$R(c,t) = \begin{cases} r_a & \text{if an obstacle is avoided} \\ r_h & \text{if home is reached} \\ 0 & \text{otherwise} \end{cases}$$

- If the robot happens to be avoiding an obstacle and reaches home at the same time, it receives reinforcement from both sources concurrently:

$$R(c, t) = r_a + r_h$$

As the above example illustrates, each of the heterogeneous reward functions provides a part of the structure of the learning task, and thus speeds up the learning.

Event–driven reward functions associate reinforcement with the achievement of goals and subgoals through the application of associated behaviors. They deliver reward or punishment in response to events, i.e. between behaviors. The next section describes a shaping mechanism for providing reinforcement during the execution of a behavior.

7.2.2 Progress Estimators

Many goals have immediately available measures of progress, since few tasks need to be defined as long sequences of behaviors without any feedback. Progress estimators use domain knowledge to measure progress during a behavior and, if necessary, to trigger principled behavior termination.

Feedback as a learning signal can be received from a one or more goals. Consider the following example:

- The robot's task is to learn to take pucks home.
- Having found a puck, the robot can wait until it accidentally finds home and then receives a reward.
- Alternatively, it can use a related subgoal, such as getting away from the food/puck pile, for feedback.
- In such a scheme, the longer the robot with a puck stays near food, the more negative reinforcement it receives.
- This strategy will encourage the behaviors that take the robot away from the food, one of which is *homing*.

While immediate reinforcement is not available in many domains, intermittent reinforcement can be provided by estimating the agent's progress relative to its current goal and weighting the reward accordingly. Measures of progress relative to a particular goal can be estimated with standard sensors, and furthermore feedback is available from different sensory modalities.

The following are the two general forms of progress estimator functions.

$$R_p(c,t) = \begin{cases} m & \text{if } c \in C' \wedge \text{progress is made} \\ n & \text{if } c \in C' \wedge \text{no progress} \end{cases} \qquad m > 0, \quad n < 0, \quad C' \subset C$$

$$R_s(c,t) = \begin{cases} i & \text{if } c \in C' \wedge \text{progress is made} \\ j & \text{if } c \in C' \wedge \text{regress is made} \\ 0 & \text{otherwise} \end{cases} \qquad i > 0, \quad j < 0, \quad C' \subset C$$

C is the set of all conditions, and C' is the set of conditions associated with the given progress estimator, i.e. those conditions for which the given progress estimator is active.

R_p and R_s have different dynamics. R_p is a two–valued function that monitors only the presence and absence of progress. R_s is a three–valued function that monitors the presence and absence of progress, as well as negative progress or regress.

Progress estimators diminish brittleness of the learning algorithm in the following ways:

- decrease sensitivity to noise
- encourage exploration in the behavior space
- decrease fortuitous rewards

Each is described in turn.

Decreasing Sensitivity to Noise

Progress estimators provide implicit domain knowledge to the learner. They strengthen appropriate condition–behavior correlations and serve as filters for spurious noise. Noise–induced events are not consistently supported by progress estimator credit, and thus have less impact on the learner. Consider the following example:

- Agent A is executing behavior B in condition c and receives positive reinforcement r_p by the progress estimator R_p.
- A receives negative reinforcement r_e from R_e as a result of an event induced by a sensor error.
- The impact of the negative reinforcement is diminished by the continuous reinforcement received from R_e throughout the execution of B.

The domain knowledge behind progress estimators provides a continuous source of reinforcement to counter intermittent and potentially incorrect credit.

Encouraging Exploration

Exploration versus exploitation is one of the critical tradeoffs in machine learning. The agent must do enough exploration to discover new and potentially more efficient condition–behavior combinations, but must also optimize its performance by using the best known pairings. Ineffective exploration results in thrashing, repeatedly attempting of one or more inappropriate behaviors.

Since situated environments are event–driven, any given behavior may persist for a potentially long period of time. An agent has no impetus for terminating a behavior and attempting alternatives, since any behavior may eventually produce a reward. The learning algorithm must use some principled strategy for terminating behaviors in order to explore the condition–behavior space effectively. Progress estimators provide such a method: if a behavior fails to make progress relative to the current goal, it is terminated and another one is tried. By using domain knowledge to judge progress, progress estimators induce exploration by terminating behaviors according to common sense, rather than according to an arbitrary internal clock or some *ad hoc* heuristic.

Decreasing Fortuitous Rewards

A *fortuitous reward* is one received for an inappropriate behavior that happened to achieve the desired goal in the particular situation, but would not have that effect in general. Consider the following scenario:

- The agent has a puck and is attempting various behaviors.
- While executing avoidance in $safe-wandering$, A fortuitously enters the home region.
- Without a progress estimator, A will receive a reward for reaching home, and will thus positively associate the avoiding behavior with the goal of getting home. It will require repeated trials in order to discover, implicitly, that the correlation is based on the direction it is moving rather than on $safe - wandering$.
- Now suppose a progress estimator H is added into the learning algorithm. H generates a reward when the agent decreases its distance to home. If it fails to do so in a given time interval, the behavior is terminated.
- Although A can still receive fortuitous rewards, their impact will be smaller compared to that of the consistent progress estimator. The continuous reward for approaching home will have a discounting effect on any

fortuitous rewards the agent receives. Thus, H will bias the agent toward behaviors that decrease the distance to home.

In general, the only way to eliminate fortuitous rewards is to know the relevance of context *a priori*. Progress estimators achieve this effect incrementally, because behaviors have some measurable duration which allows progress estimators to contribute reinforcement.

7.3 Summary

This chapter has introduced a formulation of reinforcement learning based on conditions, behaviors, and shaped reinforcement in order to: 1) make learning possible and 2) make learning efficient in complex situated domains.

The described formulation is a direct extension of behavior–based control (Matarić 1992*a*, Brooks 1991*b*, Brooks 1986). The presented heterogeneous reward functions are related to subgoals (Mahadevan & Connell 1991*a*) as well as subtasks (Whitehead et al. 1993). However, unlike previous work, which has focused on learning action sequences, this work used a higher level of description. The proposed subgoals are directly tied to behaviors used as the basis of control and learning. Similarly, progress estimators are mapped to one or more behaviors, and expedite learning of the associated goals, unlike a single complete external critic used with a monolithic reinforcement function (Whitehead 1992).

Elevating the description, control, and learning level of the system to one based on perceptual conditions and behaviors instead of perceptual states and atomic actions greatly diminishes the agent's learning space and makes learning tractable. The use of heterogeneous reward functions and progress estimators builds in domain knowledge and contextual information thus making learning more efficient.

The proposed reformulation forms a better foundation for situated learning, but does not impose any constraints on the kind of learning algorithm that can be applied. Indeed, it is completely general and compatible with any reinforcement learning approaches.

The next chapter demonstrates how this formulation was applied to the task of learning foraging in a situated, multi–robot domain.

Chapter 8

Learning Experiments

This chapter describes the learning experiments conducted to test the presented approach to setting up the learning space to enable learning, and shaping reinforcement to accelerate learning in situated domains.

8.1 The Robots

The learning experiments were performed on a group of up to four fully autonomous R2 mobile robots with on–board power and sensing (Figure 8-1). Each robot consists of a differentially steerable wheeled base and a gripper for grasping and lifting objects (Figure 8-2). The robots' sensory capabilities include piezo–electric bump sensors for detecting contact–collisions and monitoring the grasping force on the gripper, and a set of infra–red (IR) sensors for obstacle avoidance and grasping (Figure 8-3).

Finally, the robots are equipped with radio transceivers, used for determining absolute position and for inter–robot communication. Position information is obtained by triangulating the distance computed from synchronized ultrasound pulses from two fixed beacons. Inter–robot communication consists of broadcasting 6–byte messages at the rate of 1 Hz. In the experiments described here, the radios are used to determine the presence of other nearby robots. As in the first set of robot experiments, the robots are programmed in the Behavior Language (Brooks 1990*a*).

8.2 The Learning Task

The learning task consists of finding a mapping of all conditions and behaviors into the most effective policy for group foraging. Individually, each robot learns to select

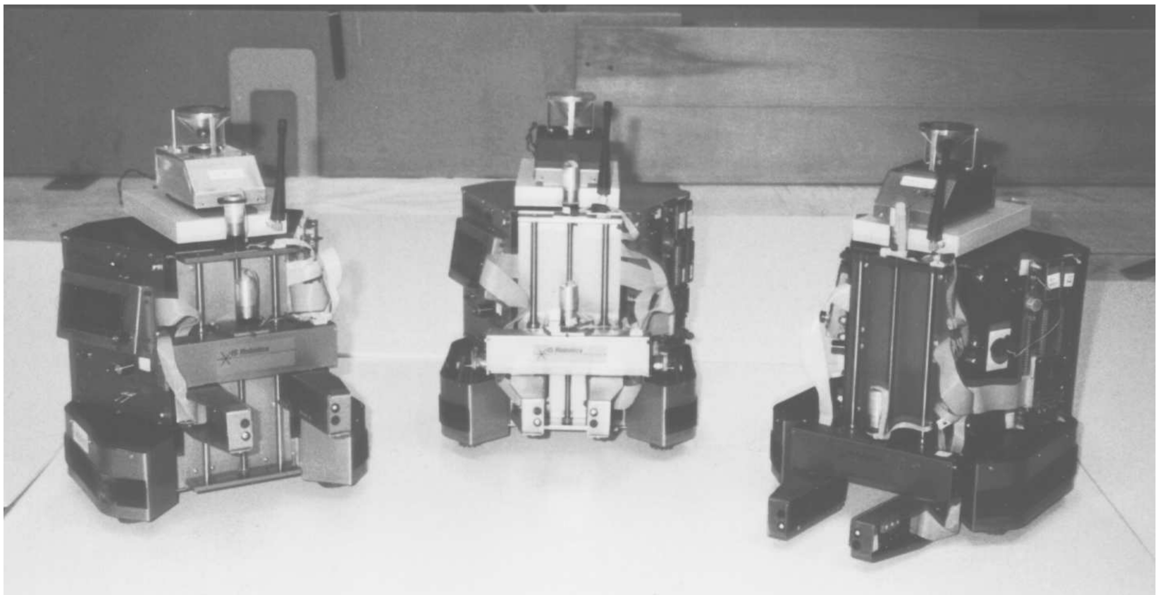


Figure 8-1: Three of the four robots used in the learning experiments. These robots demonstrated learning to forage by selecting among a basic behavior repertoire under appropriate sensory conditions.

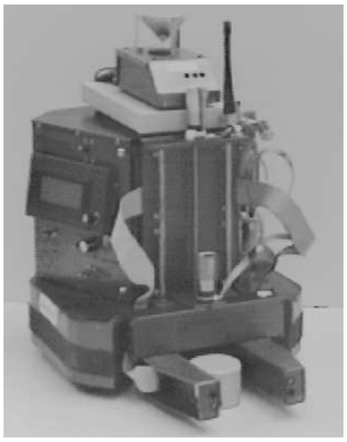


Figure 8-2: Each of the learning robots consists of a differentially steerable wheeled base and a gripper for grasping and lifting objects. The robot's sensory capabilities include piezo–electric bump and gripper sensors, infra–red sensors for collision avoidance, and a radio transmitter for absolute positioning.

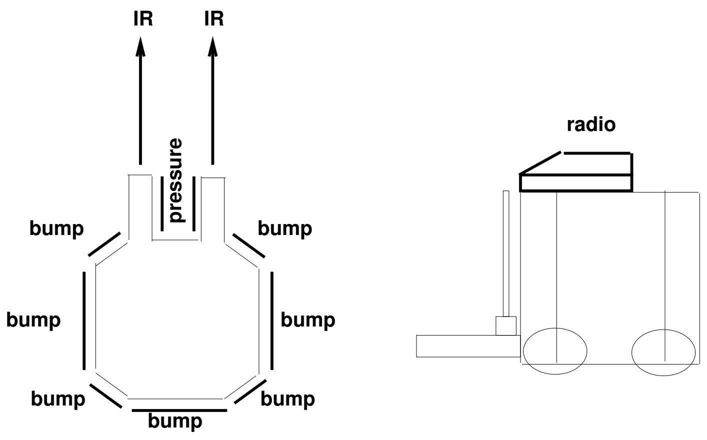


Figure 8-3: The robot's sensory capabilities include piezo–electric bump and gripper sensors used to detect collisions and to grasp pucks, infra–red sensors for collision avoidance, and a radio transmitter for absolute positioning and message passing.

the best behavior for each condition, in order to find and take home the most pucks. Foraging was chosen because it is a complex and biologically inspired task, and because our previous group behavior work, described in earlier sections and in Matarić (1992*b*) and Matarić (1993), provided the basic behavior repertoire from which to learn behavior selection. As was described in Section 5.1.2, foraging can be achieved from a small basic behavior set. Such a set, given to the robots *a priori*, consisted of the following fixed behavior repertoire:

- *safe–wandering*
- *dispersion*
- *resting*
- *homing*

Resting was introduced to expand the agents' behavior space, as well as to introduce an internal clock that can trigger internally–generated events. The internal clock imposed a cyclic "circadian" schedule consisting of periods of "day–time" and shorter periods of "night–time". *Resting* could be used as a part of a regular recharging cycle, or as a chance for the robots to aggregate and exchange information[1].

Utility behaviors for *grasping* and *dropping* objects were also included in the robots' capabilities, but since their conditions were not learned, they are not included in the above basis set nor in the learning space.

[1]Neither of these options were used in the shown generation of robots.

Given the behavior repertoire, the robots were given the task of learning the appropriate conditions for triggering each of the behaviors. By considering only the space of conditions necessary and sufficient for triggering the behavior set, the state space is reduced to the power set of the following clustered condition predicates:

- *have-puck?*
- *at-home?*
- *near-intruder?*
- *night-time?*

The conditions for *grasping* and *dropping* were built–in. As soon as a robot detects a puck between its fingers, it grasps it. Similarly, as soon as a robot reaches the home region, it drops the puck if it is carrying one. Finally, whenever a robot is too near an obstacle, it avoids. The three reflexive behaviors were deemed to be "instinctive" because learning them has a high cost. Learning to avoid has a potentially prohibitive damaging cost for the robot, and is not a natural learning task, as it appears to be innate in nature, and can be easily programmed on most systems. Puck manipulation requires a fast and accurate response from the gripper motors, and, like the other basic behaviors, is best suited for parameter learning. The remaining behaviors: *dispersion*, *safe–wandering*, *homing*, and *resting* formed a more appropriate basis for learning, because they are general and executable in a variety of situations, so finding the appropriate subset of such situations (conditions) for their activation is both an interesting learning problem and a useful application for control.

As described, the foraging task may appear quite simple, since its learning space has been appropriately minimized to include only the clustered conditions and the few basic behaviors. In theory, an agent should be able to quickly explore it and learn the optimal policy. In practice, however, such quick and uniform exploration is not possible. Even the relatively small learning space presents a challenge to an agent situated in a nondeterministic, noisy and uncertain world. As we will soon demonstrate, even in its reformulated version this problem poses a challenge for traditional RL methodologies using delayed reward, and thus also justifies the proposed shaped reinforcement strategy.

Improved reinforcement is necessary partially because, in our domain, the learner is not provided with a model of the world. As discussed earlier, such a model is difficult to obtain. Without it, the agent is faced with implicitly deducing the structure of a dynamic environment that includes other agents whose behavior occasionally facilitates, but largely interferes with, the individual learning process (see Figures 8-4

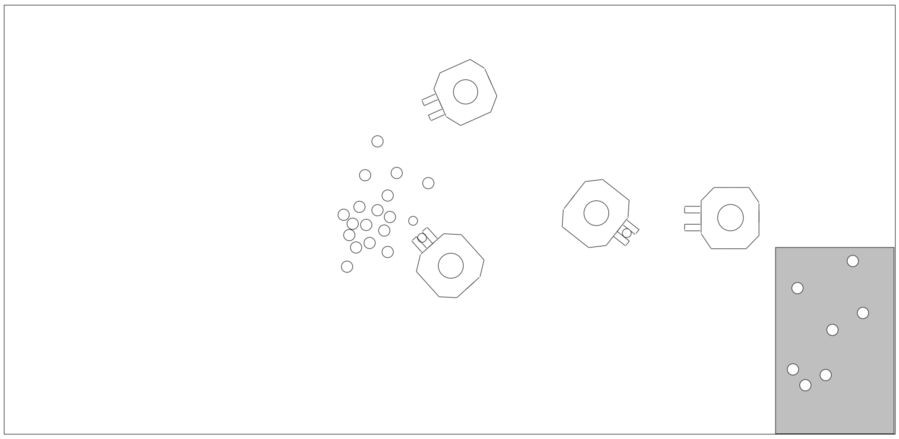


Figure 8-4: A scaled top view of the experimental area in which the learning experiments were conducted. The workspace is small enough to result in frequent interaction and interference between the robots. The home region is shaded.

and 8-5). Thus, the shown scenario poses a difficult challenge for the reinforcement learning paradigm. The next section describes our solution.

8.3 The Learning Algorithm

The learning algorithm produces and maintains a total order on the appropriateness of behaviors associated with every condition, expressed as a matrix $A(c,b)$. The value of any condition–behavior pair (c,b) is the sum of the reinforcement R received up to that point:

$$A(c,b) = \sum_{t=1}^{T} R(c,t)$$

The values in the matrix fluctuate over time based on received reinforcement. They are updated asynchronously by any received learning signal.

The following events produce immediate positive reinforcement:

- E_p: grasped-puck
- E_{gd}: dropped-puck-at-home
- E_{gw}: woke-up-at-home

"Waking–up" refers to the event of the internal clock indicating the end of night–time and the beginning of day–time.

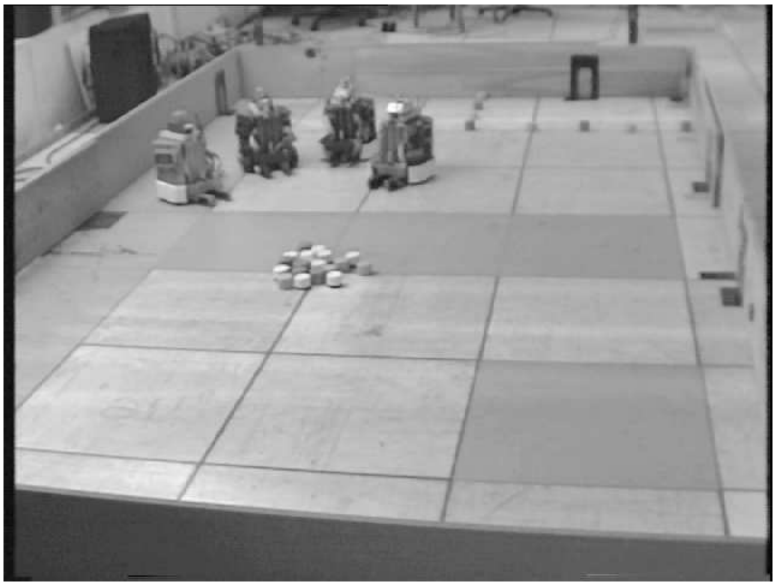


Figure 8-5: The camera's view of the experimental environment used for learning. The boundary of the home region is indicated with a row of pucks for the purposes of the photo. The pile of pucks is also marked.

The following events result in immediate negative reinforcement:

- E_{bd}: dropped-puck-away-from-home
- E_{bw}: woke-up-away-from-home

The events are combined into the following heterogeneous reinforcement function:

$$R_E(c) = \begin{cases} p & \text{if Ep occurs} \\ gd & \text{if Egd occurs} \\ bd & \text{if Ebd occurs} \\ gw & \text{if Egw occurs} \\ bw & \text{if Ebw occurs} \\ 0 & \text{otherwise} \end{cases}$$

$$p, gd, gw > 0, \quad bd, bw < 0$$

Two progress estimating functions are used: I and H. I is associated with minimizing interference and is triggered whenever an agent is close to another agent. If the behavior being executed has the effect of increasing the physical distance to the other agent, the agent receives positive reinforcement. Conversely, lack of progress away from the other agent is punished, and after a fixed time period of no progress, the current behavior is terminated.

Formally, I is the intruder avoidance progress function such that:

$$R_I(c,t) = \begin{cases} m & \text{distance to intruder increased} \\ n & \text{otherwise} \end{cases}$$

$$near\_intruder \in c, \quad m > 0, \quad n < 0$$

The other progress estimator, H, is associated with *homing*, and is initiated whenever a puck is grasped. If the distance to home is decreased while H is active, the agent receives positive reinforcement, *status quo* delivers no reinforcement, and movement away from home is punished.

Formally, H is the homing progress function such that:

$$R_H(c,t) = \begin{cases} n & \text{nearer to home} \\ f & \text{farther from home} \\ 0 & \text{otherwise} \end{cases}$$

$$have\_puck \in c, \quad n > 0, \quad f < 0$$

The simplest learning algorithm that uses the above reinforcement functions was implemented and tested. The algorithm simply sums the reinforcement over time:

$$A(c,b) = \sum_{t=1}^{T} R(c,t)$$

The influence of the different types of feedback was weighted by the values of the feedback constants. This is equivalent to the alternative of weighting their contributions to the sum, as follows:

$$R(c,t) = uR_E(c,t) + vR_I(c,t) + wR_H(c,t)$$

$$u,\ v,\ w \geq 0, \quad (u+v+w) = 1$$

Binary–valued and real–valued R_E, R_H, and R_I functions were tested. Our results showed that different weights on the reinforcement functions did not result in faster or more stable learning. This is not surprising, since the subgoals in the foraging task are independent and thus their learning speed is uncorrelated.

8.4 The Control Algorithm

The following is the complete control algorithm used for learning foraging. Behavior selection is induced by events, each of which is a change in the condition predicates. Events can be triggered:

1. **externally:** e.g., a robot gets in the way of another. External events include: E_p, E_{gd}, and E_{bd}.
2. **internally:** e.g., the internal clock indicates night–time. Internal events include: E_{gw} and E_{bw}.
3. **by progress estimators:** e.g., the interference estimator detects a lack of progress and terminates the current behavior. Estimator events are triggered by: $R_I(c,t) < intruder - threshold$ and $R_H(c,t) < homing - threshold$.

Whenever an event is detected, the following control sequence is executed:

1. appropriate reinforcement is delivered for the current condition–behavior pair
2. the current behavior is terminated
3. another behavior is selected, according to the following rule:
   (a) choose an untried behavior if one is available,
   (b) otherwise choose the best behavior.

Choosing untried behaviors first encourages exploration. Since a policy is a total ordering of the condition–behavior pairs, the agent must explore the entire behavior space before it can be said to have converged. Given the small size of the behavior set, this strategy has an accelerating effect on establishing an initial ordering of the behaviors for each condition.

Best behavior b for a given condition c is defined to be the one with the highest associated $A(c,b)$ value. Since the number of behaviors is small, this selection is easy to compute. Because of its use of positive and negative reinforcement, as well as the progress estimator induced exploration strategy, the learning algorithm does not tend to fall into local maxima. Consequently, we did not need to add randomness to the selection mechanism.

Figure 8-6: Typical initial conditions for learning trials. The robots are initiated either in the home region or in random positions around the workspace.

Figure 8-7: A typical environment state during the course of a learning experiment. Since they are learning independently, the robots have likely acquired different parts of the policy. Through interactions with objects in the world and each other they accumulate learning trials in order to complete their learning.

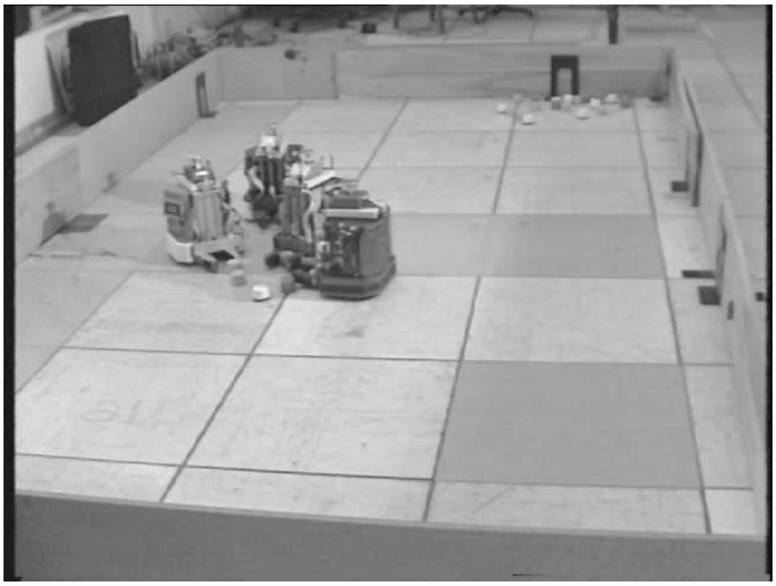


Figure 8-8: A typical environment state after learning. Most pucks have been collected and brought to the home region. The robots have all learned when to go get the pucks, and are thus competing for those remaining to be moved.

Learning is continuous and incremental over the lifetime of the agent, thus ensuring that the agent remains responsive to changes in the environment (e.g., no more pucks are left at a particular location) and internal changes in function (e.g., dying battery slows motion down.

In the described learning task, the optimal policy was derived by hand, based on empirical data from the foraging experiments described in Section 5.1.2, and with the addition of the new *resting* behavior. This policy is shown in Figure 8.1. The performance of the desired policy was tested independently and compared to alternative solutions in order to establish its superiority relative to the imposed evaluation criteria.

Snapshots of a learning experiment are shown to illustrate the progression of a typical of experiment. Figure 8-6 shows typical initial conditions, Figure 8-7 demonstrates a stage during the course of learning, and Figure 8-8 shows the environment toward the end of the experiment, when most of the pucks have been collected. Figure 8-9 illustrates the *resting* behavior.

The learning process consists of adjusting the values in a table with a total of 64 entries: 2^4 conditions $*$ 4 behaviors. Table 8.2 shows the table and the policy the agents were initialized with. The utility of all behaviors in all conditions is equal, and initialized to the average of the minimum and maximum $A(c,b)$ value.

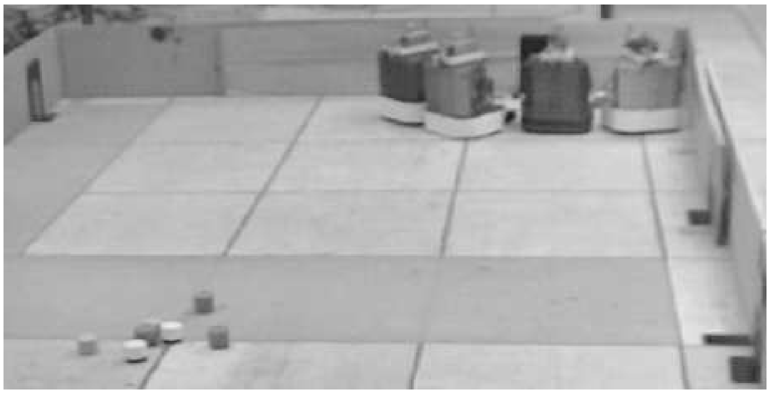


Figure 8-9: An example of the resting (or recharging) behavior of four robots, triggered by their internal clocks. In this case, the robots have all learned to go home to rest, as this photo illustrates a late stage in the learning, as demonstrated by the small number of remaining pucks.

Condition				**Behavior**
near-intruder?	have-puck?	at-home?	night-time?	
0	0	0	0	safe–wandering
0	0	0	1	homing
0	0	1	0	safe–wandering
0	0	1	1	resting
0	1	0	0	homing
0	1	0	1	homing
0	1	1	0	safe–wandering
0	1	1	1	resting
1	0	0	0	safe–wandering
1	0	0	1	safe–wandering
1	0	1	0	dispersion
1	0	1	1	resting
1	1	0	0	homing
1	1	0	1	homing
1	1	1	0	safe–wandering
1	1	1	1	resting

Table 8.1: The optimal foraging policy. Only the top–ranked behavior is shown for each condition. The full table has a total numerical ordering of four behaviors for each condition, a total of 64 entries.

Condition	Behavior			
	safe–wandering	homing	dispersion	resting
0000	50	50	50	50
0001	50	50	50	50
0010	50	50	50	50
0011	50	50	50	50
0100	50	50	50	50
0101	50	50	50	50
0110	50	50	50	50
0111	50	50	50	50
1000	50	50	50	50
1001	50	50	50	50
1010	50	50	50	50
1011	50	50	50	50
1100	50	50	50	50
1101	50	50	50	50
1110	50	50	50	50
1111	50	50	50	50

Table 8.2: The policy agents are initiated with. The utility of all behaviors in all conditions is equal, and initialized to the average of the minimum and maximum.

8.5 Experimental Results and Evaluation

The effectiveness of the proposed reinforcement functions was evaluated by testing three different types of reinforcement. The following three approaches were compared:

1. A monolithic single–goal (puck delivery to the home region)
   reward function $R(c,t) = R_{E_{gd}}(c,t)$,
   and using the Q-learning algorithm,

2. A heterogeneous reward function using multiple goals: $R(t) = R_E(t)$,
   and using the reinforcement summation algorithm $A(c,b) = \sum_{t=1}^{T} R(c,t)$,

3. A heterogeneous reward function using multiple goals $R(t) = R_E(t)$
   and two progress estimator functions $R_H(c,t)$ and $R_I(c,t)$,
   and using the reinforcement summation algorithm $A(c,b) = \sum_{t=1}^{T} R(c,t)$.

Data from sixty trias, twenty of each of the three strategies, were collected and averaged. The experiments were run on four different robots, and no significant robot–specific differences were found. Data from runs in which persistent sensor failures occurred were discarded.

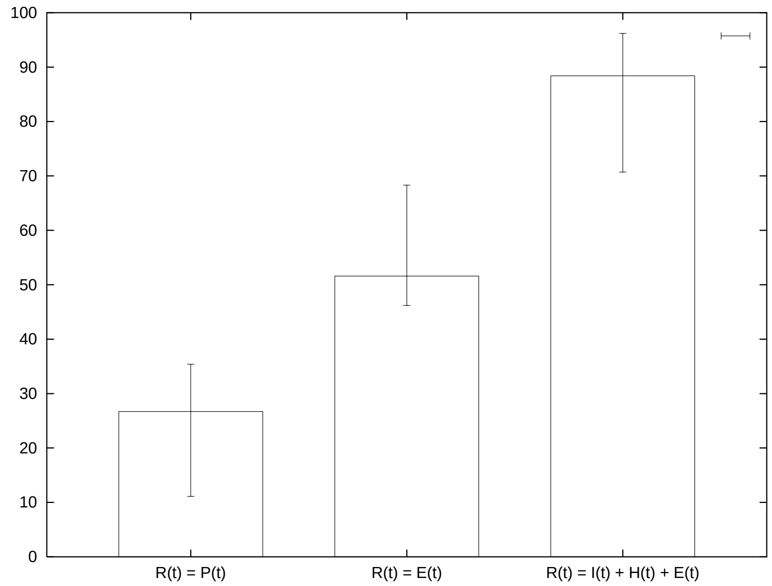


Figure 8-10: The performance of the three reinforcement strategies on learning to forage. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned in 15 minutes, averaged over twenty trials. The error bars show the best and worst performance, and the histograms the average value.

The data were based on values of $A(c, b)$, which were collected twice per minute during each learning experiment and once at the completion of the experiment, showing the final values. All experiments lasted 15 minutes. The 15 minute threshold was empirically derived, since the majority of the learning trials reached a steady state after about 10 minutes, except for a small number of rare conditions, as discussed below.

8.5.1 Evaluation

Evaluating performance of situated systems is notoriously difficult among other reasons because standard metrics for evaluating learning mechanisms, such as absolute time–to–convergence, do not directly apply. The amount of time required for a robot to discover the correct policy depends on the frequency of external events that trigger different states in its learning space. Additionally, noise and error can make certain parts of the policy fluctuate so waiting for a specific point of absolute convergence is not feasible. Instead, convergence is defined as a relative ordering of condition–behavior pairs.

The performance of the three approaches is compared in Figure 8-10. The x-axis shows the three reinforcement strategies. The y-axis maps the percent of the correct policy the agents learned, in 15 minutes, averaged over twenty trials, i.e., the ratio of correct condition–behavior pairings according to the optimal policy. The error bars show the best and worst performance, and the histograms the averaged value.

Q-Learning Performance

As described above, Q-learning was tested on the reduced learning space using the enumerated conditions and behaviors. In terms of reinforcement, Q-learning used a simplified version of the second algorithm, based on an impulse function delivering positive reinforcement for the single goal of dropping a puck in the home region. Given the nondeterminism of the world, and the uncertainty in sensing and state transitions, the single goal provides insufficient feedback for learning all aspects of foraging, in particular those that rely on accurate delayed credit assignment. The performance of Q-learning was vulnerable to interference from other robots, and declined most rapidly of the three approaches when tested on increased group sizes.

Q performs poorly, but the partial policy it discovers is consistent over all trials and is made up of the few condition–behavior pairs that receive immediate and reliable reinforcement. Thus, the performance of Q indicates the difficulty of the learning task at least to the extent of demonstrating the immediately reinforced parts as the only parts it is capable of learning.

It is important to note that Q is unable to take advantage of reward discounting because there is no particularly useful ordering to the sequence of behaviors an agent executes at any time in our domain, because the agent's behavior is dependent on the behavior of all of the others that interact with it during that time. These interactions are not individually modeled and learned in order to avoid a prohibitively large learning space as well as high sensing overhead. Consequently, the agent cannot deduce any structure about sequential behaviors from discounting, because at its representational level there is no structure. It needs to acquire a fully reactive policy which does not benefit from temporal discounting.

Multiple Goal Performance

The second learning strategy, utilizing reinforcement from multiple goals, outperforms Q because it detects the achievement of the subgoals on the way of the top–level goal of depositing pucks at home. However, it also suffers from the credit assignment problem in the cases of delayed reinforcement, since the nondeterministic environment with other other agents does not guarantee consistency of rewards over time.

Furthermore, this strategy does not prevent thrashing, so certain behaviors are active unnecessarily long. For example, *safe–wandering* and *grasping* are pursued persistently, at the expense of behaviors with delayed reinforcement, such as *homing*. The performance of heterogeneous reinforcement gives us another evaluation of the difficulty of the proposed learning task. With around 60% of the correct policy learned on the average, it demonstrates that additional structure is necessary to aid the learner in acquiring the rest. This structure is provided by progress estimators.

Progress Estimator Performance

The complete heterogeneous reinforcement and progress estimator approach maximizes the use of all potentially available information for every condition and behavior. As predicted, thrashing is eliminated both in the case of learning the conditions for *dispersion* and *homing* because the progress estimator functions encourage exploration. Furthermore, fortuitous rewards have less impact than in the alternative algorithms. The implicit domain knowledge is effectively spread over the reinforcement in order to guide the learning process continually, thus maximizing the utility of each of the learning trials and consequently speeding up the learning.

The design of the foraging task using basic behaviors guarantees that its subgoals are independent of each other. Consequently, the associated reinforcement functions do not directly affect each other, and the simple ones we used are mutually consistent as they all contribute to a common high–level goal. Although in theory the more reinforcement is used the faster the learning should be, in practice noise and error in the different reinforcement sources could have the opposite effect. Our experiments demonstrated that a significant amount of noise and inconsistency in the different reinforcers and progress estimators did not adversely affect the learner.

For example, each robot's estimate of its position and the proximity of others was frequently inaccurate due to radio transmission delays. These errors resulted in faulty homing and interference progress estimates. Nonetheless, all condition–behavior pairs that involved carrying a puck converged quickly. Furthermore, their $A(c, b)$ values did not tend to oscillate. The fast rate of convergence for associations with behaviors that involved *dispersion* and *homing* result directly from the effects of the two progress estimators. When the two are removed, as in the second tested algorithm, the performance declines accordingly.

Conversely, the set of conditions associated with finding pucks uniformly took longer to learn, since they had no direct progress measure to accelerate the learning. Furthermore, the learned values initially tended to oscillate, since the differences between the behavior alternatives were not great, again due to a lack of intermediate rewards. Empirical results show that noise and error–induced inconsistencies in the progress estimators did not significantly diminish the benefit of their use in this

Reinforcement	Effect
$R(c,t) = R_{E_{gd}}(t)$	converges for at most 1/3 of the policy
$R(c,t) = R_E(t)$	converges for at least 1/2 of the policy
$R(c,t) = R_E(t) + R_I(t) + R_H(t)$	converges for at least 2/3 of the policy

Table 8.3: A qualitative summary of the performance for the three types of reinforcement used on the foraging task.

domain.

8.5.2 Further Evaluation

Table 8.3 shows a coarse performance ordering of the three approaches. Although intuitive, this ordering is not particularly informative. A better way to analyze the approaches is to evaluate each part of the policy separately, thus measuring when and what each robot was learning. Table 8.4 illustrates the final state of a learner using heterogeneous reward functions and progress estimators. The table provides additional information for analysis.

To capture the dynamics of the learning process, each condition–behavior pair was evaluated according to the following three criteria:

1. number of trials required,
2. correctness,
3. stability.

The number of trials was measured relative to a stable solution, whether the solution was optimal or not. The second criterion sought out incorrect (in terms of optimality) but stable solutions. Finally, the third criterion focused on unstable policies, looking for those in which the behavior orderings tended to fluctuate.

Based on those criteria, some condition–behavior pairs proved to be much more difficult to learn than others. The most prominent source of difficulty was the delay in reinforcement, which had predictable results clearly demonstrated by the performance differences between the three strategies. Learning the conditions for *safe–wandering* was difficult as there was no available progress estimator, and the robot could be executing the correct behavior for a long while before reaching pucks and receiving reward. In the mean time it could be repeatedly interrupted by other activities, such as avoiding obstacles and intruders, as well as *homing* and *resting* at the onset of night–time.

Condition	**Behavior**			
	safe–wandering	homing	dispersion	resting
0000	100	45	40	35
0001	45	100	35	45
0010	100	40	45	30
0011	30	45	40	100
0100	55	100	40	35
0101	65	100	35	40
0110	100	45	65	30
0111	30	40	30	100
1000	100	40	75	35
1001	100	80	60	45
1010	85	30	100	45
1011	40	45	30	100
1100	100	95	45	40
1101	45	100	60	40
1110	100	45	90	30
1111	65	30	45	100

Table 8.4: An example policy learned by one of the robots using heterogeneous reward functions and progress estimators.

Another source of difficulty was rareness of occurrence of some combinations of conditions. In particular, the condition consisting of the onset of night–time while a robot is carrying a puck and avoiding another robot rarely occurred. Consequently, the correct mapping was difficult to learn since the robots did not get a chance to explore all behavior alternatives. This accounts for the incomplete policy even in the case of the most successful reinforcement strategy.

The combination of positive and negative reinforcement pushes the learner out of any local maxima, but allows oscillations and instabilities in the ordering of the $A(c, b)$ values in the table. Two of the conditions oscillated because the alternatives resulted in equally effective solutions. In situations when the robot is not carrying a puck and encounters an intruder, any motion away from the intruder will be beneficial and rewarded by the progress estimator R_I. Consequently, *homing* and *safe–wandering* are often as effective as *dispersion*. In contrast, if the robot is carrying a puck, then *dispersion* and *homing* are effective and rewarded by contributions of the R_I and R_H progress estimators. As described earlier, it is the combination of the two estimators that speeds up exploration as well as minimizes fortuitous rewards. Only a specific progress measure that minimizes the travel time to the goal can eliminate this effect.

Such optimization is difficult in systems using largely local sensing and control and dealing with interference from other agents. Given those challenges, the policy the robots found was appropriate for the properties of their domain.

8.5.3 Scaling

We evaluated the three reinforcement alternatives on groups of three and four robots and found that interference was a detriment to all three. In general, the more robots were learning at the same time, the longer it took for each to converge. This was particularly pronounced for condition–behavior pairs without directly associated progress estimators, such as those involved in the conditions that did not involve carrying a puck.

The only behavior capable of reaping benefits from interference was *dispersion*, which was learned faster and more accurately in crowded situations. We have considered adding a social behavior called *yielding* in order to minimize interference by having only one robot move at a time in crowded situations. Our previous results, described in section 4.5.2, showed that such "hierarchical" behavior had little effect on individual basic behaviors *aggregation* and *dispersion*. However, we believe *yielding* would be more effective in the case of foraging. Because of fixed home and puck locations, the task is more structured and can take advantage of rules that produce more structured motion.

8.6 Discussion and Extensions

8.6.1 Social Rules

We have noted that a decline in performance of all of the algorithms was observed with the increased group size and the associated increased interference between agents. Although not surprising, this is an undesirable effect. In an ideal scenario, the presence of other agents would speed up rather than slow down individual learning. However, such synergy is only possible in societies where individuals benefit from each other's experience and interact according to mutually beneficial social rules.

Our most recent work has addressed the problem of learning such social rules. This is a challenging learning problem since social rules do not necessarily have immediate or even delayed payoff to the individual but may only benefit the individual on average from having a global effect. Consequently, social rules involve some "altruistic" behavior, even at the simplest of levels, such as *yielding* in traffic. Such behavior is difficult to learn with individualist reinforcement learning strategies. We are currently working on an algorithm that utilizes the observation of neighboring agents' behavior and received reinforcement in order to acquire and practice social behaviors (Matarić 1994).

8.6.2 Transition Models

The learning problem presented here, involving a collection of concurrently learning agents in a noisy and uncertain environment, was purposefully chosen for its complexity. The fact that a state transition model was not available to aid the learner presented one of the major challenges.

As argued earlier, such models are not generally available, but partial models could be constructed empirically, either prior to or during the learning process. The implemented reinforcement functions take advantage of immediate information from the world to generate reinforcement. Thus, they would have an accelerating effect on any learning domain, regardless of whether a transition model is available. An interesting extension of this work would apply the described reinforcement approach to problems that involve incomplete and approximate state transition models in order to study the effects of combining immediate reinforcement with discounted future rewards commonly applied to RL problems.

8.6.3 Heterogeneous Learning

One of the key advantages of heterogeneous reinforcement is the possibility of learning multiple types of behaviors in parallel. Such concurrent multi–modal learning is biologically and pragmatically inspired, and has been an ongoing challenge in the learning community (Franklin & Selfridge 1990, Brooks & Matarić 1993).

In our foraging task, basic behaviors were designed by hand and behavior selection was learned. However, basic behaviors themselves could be learned or optimized in parallel with learning behavior selection. For example, the agents could use a parameter learning scheme to optimize their grasping behaviors whenever in a puck–carrying state. In order to avoid extending the learner's state space and reverting to the traditional problems of monolithic learners, multi–modal learning would be implemented using multiple correlation mechanisms instead of a monolithic $A(c, b)$ matrix.

The described reinforcement techniques can be applied at every learning level. No explicit merging of learned policies is needed since the learning modules would be independent.

8.6.4 Structuring Learning

One of the difficulties facing the learning community is the lack of structure that taxonomizes the existing learning methodologies and delineates their applicability. Consequently, the choice of methodology is often based on passing trends and dogma rather than on objective applicability and performance criteria. One of the goals of the learning work described in this thesis has been to introduce some structure into the popular methodology broadly characterized as reinforcement learning. By applying reinforcement learning to a novel and more complex domain than has been experimented with to date, we were able to establish its limitations for that domain, and propose a reformulation of the representation and reinforcement that makes learning in that domain both possible and efficient.

By appropriately setting up the learning task, effective results were achieved from a single learning methodology. An interesting direction to pursue would be to deal with learning problems complex enough to require more than one learning strategy as a means of relating different techniques.

8.6.5 Signal–to–Symbol Learning

Signal–to-symbol learning encapsulates the entire learning process from the grounding of the agent's experiences in the world to the resulting comparatively high–level representations. To date, systems that have learned from low–level signals, such as sensory information, have either bypassed symbolic representations all together, or had them built–in by the designer. An the other end of the spectrum, symbolic high–level learning has not traditionally concerned itself with grounding in the physical world. However, for situated systems, which must make a connection between direct sensory experiences and high–level cognitive activities, symbol grounding is an important problem that must be addressed (Harnad 1990).

Most work on situated agents to date has not dealt with what are considered to be highly cognitive tasks. However, even learning of "lower–level" capacities, such as complex motor behaviors, requires intermediate and increasingly abstract representations. The process of relabeling information into forms that can be used by other subsystems for achieving different goals is already a step in the direction of bridging the signal–to–symbol gap.

The learning work presented here has been at a level that could use a simple mapping between conditions and behaviors. Nonetheless, even the process of constructing the presented reusable behavior combinations requires some way of labeling the combinations. As most other work, the learning strategy described here was able to use a built–in mapping to labeled behaviors. A more general solution to the problem is desirable, and we hope to address it in future work.

8.7 Summary

The goal of the described learning work has been to bring to light some of the important properties of situated domains, and their impact on reinforcement learning strategies. We have described why MDP models of agent–world interactions are not effective in the noisy multi–agent domain, how the traditional notions of state and action present an inappropriately low level of system description for control and learning, and how delayed reinforcement is not sufficient for learning in our domain and other domains of similar level of complexity.

We introduced a higher–level description of the learning system, based on conditions and behaviors, that greatly diminishes the learner's state space and results in more robust control. We also introduced a methodology for shaping reinforcement in order to take advantage of more information available to the agent. In our domain shaping was necessary given the complexity of the environment–agent and agent–agent interactions. The approach consists of two methods: one that partitions the learning task into natural subgoals (behaviors) and reinforces each separately, and one that employs progress estimators to generate more immediate feedback for the agent.

The proposed formulation was evaluated on a group of physical robots learning to forage and was shown to be effective as well as superior to two alternatives. The approach is general and compatible with the existing reinforcement learning algorithms, and should thus serve to make learning more efficient in a variety of situated domains and with a variety of methodologies.

Chapter 9

Summary

The aim of this thesis has been to gain insight into intelligent behavior by increasing the level of complexity of the systems being designed and studied. In contrast to many AI systems that have focused either on complex cognition situated in simple worlds, or vice versa, the work described here has addressed situated, embodied agents coexisting and interacting in a complex domain (Figure 9-1). We hope that the methodologies and results presented here have extended the understanding of synthesis, analysis, and learning of group behavior.

Selection of the appropriate representation level for control, planning, and learning is one of the motivating forces behind this work. We have proposed a methodology for using constraints in order to derive *behaviors*, control laws that guarantee the achievement and maintenance of goals. Furthermore, we described a methodology for selecting *basic behaviors*, a basis set of such behaviors to be used as a substrate for control and learning for a given agent and environment.

We demonstrated these ideas on the problem of synthesizing coherent group behavior in the domain of planar spatial interactions. We devised a basic behavior set and showed that it meets the defining criteria, including no mutual reducibility and simple combination. We then showed how basic behaviors and their conditions can be used as a substrate for learning. Furthermore, we described a methodology for shaping reinforcement by using *heterogeneous reinforcement functions* and *progress estimators* in order to make learning possible and more efficient in dynamic multi–agent domains.

The main idea behind this work is the approach to combining constraints from the agent, such as its mechanical and sensory characteristics, and the constraints for the environment, such as the types of interactions and sensory information the agent can obtain, in order to construct constraint–based primitives for control. At the

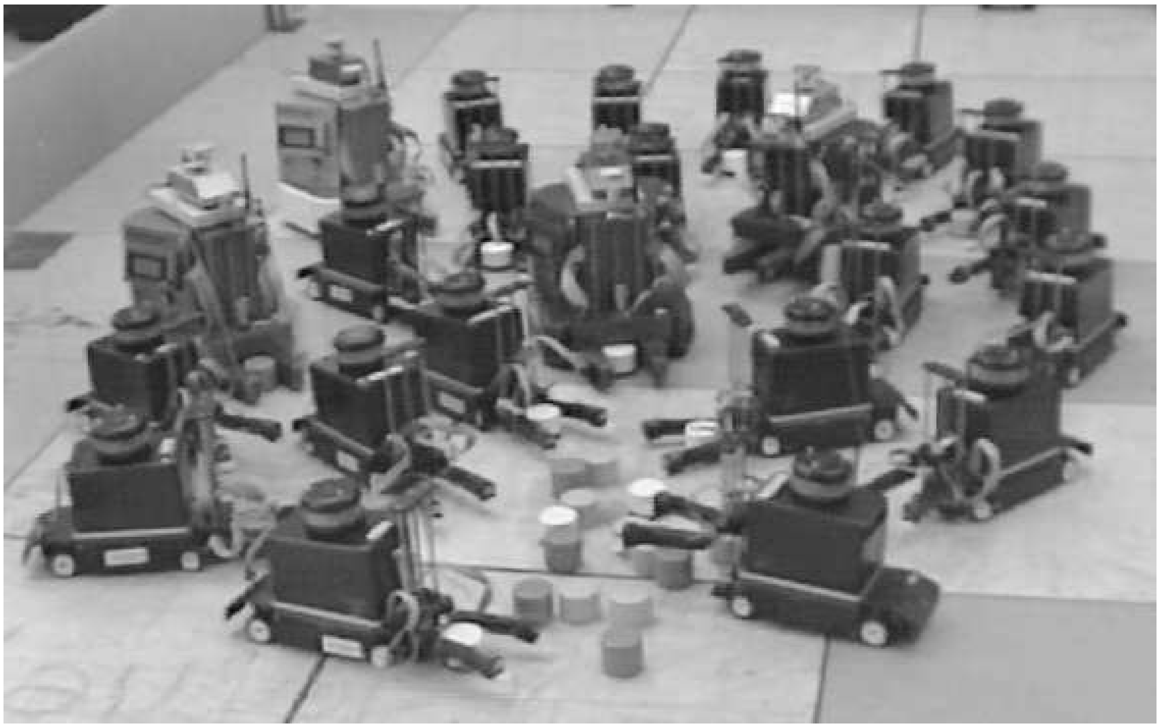


Figure 9-1: A family photo of the physical experimental agents used to demonstrate and verify the group behavior and learning work described in this thesis.

sensory end we called these primitives *conditions* and at the action end we referred to them as *behaviors*. In both cases they are a clustering of constraints that provide an abstraction at a level that makes control and learning efficient.

We have dealt with a complex multi–agent domain and a complex learning problem in order to fully confront the issues in selecting the right abstraction and representation level for situated agents. The complexity of our chosen environment, combined with the requirement of acting in real time, enforced the necessity for using a representation level that was not so low as to be computationally intractable or so high as to remove the potential of novel behavior strategies to be designed or learned by the agents.

This work is intended as a foundation in a continuing effort toward studying increasingly more complex behavior, and through it, more complex intelligence. The work on basic behaviors distills a general approach to control, planning, and learning. The work also brings to light some theoretically and empirically challenging problems and offers some effective solutions to situated learning. Future work should both analytically tighten and experimentally broaden our understanding of all those issues. The demonstrated results in group behavior and learning are meant as stepping stones toward studying increasingly complex social agents capable of more complex learning, ultimately leading toward better understanding of biological intelligence.

Appendix A

Q-learning

Watkins (1989) introduced a family of methods he called Q-learning for solving Markov decision problems with incomplete information, through the use of delayed reinforcement. The simplest version, called one-step Q-learning, is the most commonly used and is thus described below.

Q-learning is based on a temporal differencing strategy that attempts to maximize $Q(s, a)$ at each time step. $Q(s, a)$ is the expected discounted reward of taking action a in the input state s. The Q values for all state–action pairs are stored in the Q table and updated at each time step. The utility E of a state is the maximum Q value of all actions that can be taken in that state. The Q value of doing an action in a state is defined as the sum of the immediate reward r and the utility $E(s')$ of the next state s' according to the state transition function T, discounted by the parameter γ.

Formally:

$$s' \leftarrow T(s, a)$$

$$E(s) = \max_a Q(s, a)$$

$$Q(s, a) = r + \gamma E(s'),\ 0 \leq \gamma \leq 1$$

Q values are updated by the following rule:

$$Q(s, a) \leftarrow Q(s, a) + \beta(r + \gamma E(s') - Q(s, a))$$

$$0 \leq \beta \leq 1$$

An RL algorithm using Q-learning has the following form:

1. Initialize all $Q(s, a)$; select s_0.
2. Do Forever:
    a. Observe the current world state s.
    b. Choose an action a that maximizes $Q(s, a)$.
    c. Execute action a.
    d. Let r be the immediate reward for executing a in state s.
    e. Update $Q(s, a)$ according to the rule above.
    Let the new state be $s' \leftarrow T(s, a)$.

β and γ are the tunable learning parameters. β determines the learning rate. $\beta = 1$ disregards all history accumulated in the current Q value and resets Q to the sum of the received and expected reward at every time step, usually resulting in oscillations.

γ is the discount factor for future reward. Ideally, γ should be as close to 1 as possible so that the relevance of future reward is maximized. In deterministic worlds γ can be set to 1, but in the general case two algorithms with $\gamma = 1$ cannot be compared since, in the limit, the expected future reinforcement of both will go to infinity.

The choice of initial Q values can affect the speed of convergence since the farther they are from the optimal policy the longer it takes to converge. If initialized to 0's in a problem set up to have a positive optimal policy, the algorithm will tend to converge to the first positive value, without exploring alternatives, so random actions must be added to guarantee that the entire action space is explored (Kaelbling 1990). Alternatively, if the optimal policy can be roughly estimated, Q values can be initialized to be higher and decreased over time. However, Q is sensitive to the coupling between the initial values and the reinforcement function. If the reinforcement function is strictly positive and the Q table is initialized to values exceeding the optimal policy, the system will take longer to converge than if the reinforcement function contains some negative signals.

Appendix B

Glossary

adaptability the ability to cope with internal and external changes.

agent an entity or computational process that senses its world and acts on it.

arbitration the problem of coordinating the activity of multiple input behaviors in order to produce desired output behavior.

basic behaviors building blocks for control, planning, and learning.

basic behavior set a basis set of behaviors that are directly, or by combination, sufficient for reaching all goals of a system. The elements of the set are not mutually reducible.

behavior a control law that achieves and/or maintains some goal.

behavior conditions proper subsets of the state space necessary and sufficient for activating a behavior.

collective behavior an observer–subjective definition of some spatial and/or temporal pattern of interactions between multiple agents.

condition a predicate on sensor readings that maps into a proper subset of the state space.

cooperation a form of interaction, usually based on communication.

ensemble behavior observable global behavior of a group or collection of agents

event a change in the agent's perceptual or condition vector.

external state externally observable state of an agent.

fortuitous reward a reward received for an inappropriate behavior that happened to achieve the desired goal.

group density the ratio of the sum of the agents' footprints and the size of available interaction space.

direct communication an action with the sole purpose of transmitting information.

directed communication communication aimed at a particular receiver or set of receivers.

direct behavior combination a temporal overlap of two or more behaviors. More than one behavior is active at a time. Implemented with a summation operator.

embodiment the state of being embodied, having a body with physical constraints and properties.

explicit cooperation a set of interactions which involve exchanging information or performing actions in order to benefit another agent.

footprint the sphere of an agent's its influence.

implicit cooperation a form of interactions consisting of actions that are a part of the agent's own goal–achieving behavior, but may have effects in the world that help other agents achieve their goals.

impulse reinforcement reinforcement delivered only when the agent reaches a single goal state.

group a collection of size three or more.

homogeneity the property of being situated in the same world, embodied with similar dynamics and executing identical control programs.

heterogeneity the property of being different from another agent in terms of one's environment, embodiment, or control.

interaction mutual influence on behavior.

interference any influence that partially or completely blocks an agents' goal–driven behavior.

multi-agent control generating the desired behavior for a multi–agent system.

niche a habitat, a class of environments for which an agent is adapted.

non-directed communication communication not limited to a particular receiver or set of receivers; includes indirect and direct communication.

multi-agent system a system consisting of at least two agents.

policy a mapping of inputs, states, or conditions, to actions or behaviors.

situatedness the property of being situated, of existing in some context, in an environment which involves interaction dynamics.

stigmergic communication communication based on modifications of the environment rather than direct message passing.

temporal behavior combination a temporal sequence of two or more behaviors. Only one behavior is active at a time. Implemented with a switching operator.

thrashing repeated execution of one or more inappropriate behaviors.

Bibliography

Abraham, R. H. & Shaw, C. D. (1992), *Dynamics: The Geometry of Behavior*, Addison–Wesley, California.

Abravanel, E. & Gingold, H. (1985), 'Learning Via Observation During the Second Year of Life', *Developmental Psychology* **21**(4), 614–623.

Agre, P. E. & Chapman, D. (1987), Pengi: An Implementation of a Theory of Activity, *in* 'Proceedings, AAAI-87', Seattle, WA, pp. 268–272.

Agre, P. E. & Chapman, D. (1990), What Are Plans for?, *in* P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', The MIT Press, pp. 17–34.

Altenburg, K. (1994), Adaptive Resource Allocation for a Multiple Mobile Robot System using Communication, Technical Report NDSU–CSOR–TR–9404, North Dakota State Univeristy.

Altenburg, K. & Pavicic, M. (1993), Initial Results in the Use of Inter-Robot Communication for a Multiple, Mobile Robotic System, *in* 'Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots', Chambery, France, pp. 96–100.

Arkin, R. C. (1989), Towards the Unification of Navigational Planning and Reactive Control, *in* 'AAAI Spring Symposium on Robot Navigation', pp. 1–5.

Arkin, R. C. (1992), 'Cooperation without Communication: Multiagent Schema Based Robot Navigation', *Journal of Robotic Systems*.

Arkin, R. C., Balch, T. & Nitz, E. (1993), Communication of Behavioral State in Multi-agent Retrieval Tasks, *in* 'IEEE International Conference on Robotics and Automation', pp. 588–594.

Asendorpf, J. B. & Baudonniere, P.-M. (1993), 'Self-Awareness and Other-Awareness: Mirror Self-Recognition Synchronic Imitation Among Unfamiliar Peers', *Developmental Psychology* **29**(1), 89–95.

Assad, A. & Packard, N. (1992), Emergent Colonization in an Artificial Ecology, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 143–152.

Atkeson, C. G. (1989), 'Learning Arm Kinematics and Dynamics', *Annual Review of Neuroscience* **12**, 157–183.

Atkeson, C. G. (1990), Memory-Based Approaches to Approximating Continuous Functions, *in* 'Proceedings, Sixth Yale Workshop on Adaptive and Learning Systems'.

Atkeson, C. G., Aboaf, E. W., McIntyre, J. & Reinkensmeyer, D. J. (1988), Model-Based Robot Learning, Technical Report AIM-1024, MIT.

Axelrod, R. (1984), *The Evolution of Cooperation*, Basic Books, New York.

Bandura, A. (1971), Analysis of Modeling Processes, *in* A. Bandura, ed., 'Psychological Modeling: Conflicting Theories', Aldine–Atherton, pp. 1–62.

Bandura, A. (1977), *Social Learning Theory*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Bandura, A. & Walters, R. H. (1963), *Social Learning and Personality Development*, Holt, Rinehart and Winston, Inc, New York.

Barman, R. A., Kingdon, J. J., Mackworth, A. K., Pai, D. K., Sahota, M. K., Wilkinson, H. & Zhang, Y. (1993), Dynamite: A Testbed for Multiple Mobile Robots, *in* 'Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots', Chambery, France, pp. 38–44.

Barto, A. G. (1990), Some Learning Tasks from a Control Perspective, Technical Report COINS TR 90–122, University of Massachusetts.

Barto, A. G., Bradtke, S. J. & Singh, S. P. (1993), 'Learning to Act using Real-Time Dynamic Programming', *AI Journal*.

Barto, A. G., Sutton, R. S. & Anderson, C. W. (1983), 'Neuronlike Elements That Can Solve Difficult Learning Control Problems', *IEEE Transactions on Systems, Man, and Cybernetics* **13**, 835–846.

Beckers, R., Holland, O. E. & Deneubourg, J. L. (1994), From Local Actions to Global Tasks: Stigmergy and Collective Robotics, *in* R. Brooks & P. Maes, eds, 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', The MIT Press.

Belić, M. R., Skarka, V., Deneubourg, J. L. & Lax, M. (1986), 'Mathematical Model of Honeycomb Construction', *Mathematical Biology* **24**, 437–449.

Bellman, R. E. (1957), *Dynamic Programming*, Princeton University Press, Princeton, New Jersey.

Benhamou, S. & Bovet, P. (1990), Modeling and Simulation of Animal's Movements, *in* J. A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press.

Beni, G. & Hackwood, S. (1992), The Maximum Entropy Principle and Sensing in Swarm Intelligence, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 153–160.

Bizzi, E. & Mussa-Ivaldi, F. A. (1990), Muscle Properties and the Control of Arm Movement, *in* D. N. Osherson, S. Kosslyn & J. M. Hollerbach, eds, 'Visual Cognition and Action, Vol. 2', The MIT Press, pp. 213–242.

Bizzi, E., Mussa-Ivaldi, F. A. & Giszter, S. (1991), 'Computations Underlying the Execution of Movement: A Biological Perspective', *Science* **253**, 287–291.

Bonabeau, E. W. (1993), On the appease and dangers of synthetic reductionism, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 86–10.

Brock, D. L., Montana, D. J. & Ceranowicz, A. Z. (1992), Coordination and Control of Multiple Autonomous Vehicles, *in* 'IEEE International Conference on Robotics and Automation', pp. 2725–2730.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. A. (1990*a*), The Behavior Language; User's Guide, Technical Report AIM-1127, MIT Artificial Intelligence Lab.

Brooks, R. A. (1990*b*), Elephants Don't Play Chess, *in* P. Maes, ed., 'Designing Autonomous Agents', The MIT Press, pp. 3–15.

Brooks, R. A. (1991*a*), Artificial Life and Real Robots, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press.

Brooks, R. A. (1991*b*), Intelligence Without Reason, *in* 'Proceedings, IJCAI-91'.

Brooks, R. A. (1991*c*), 'Intelligence Without Representation', *Artificial Intelligence* **47**, 139–160.

Brooks, R. A. & Connell, J. H. (1986), Asynchronous Distributed Control System for a Mobile Robot, *in* 'SPIE', Cambridge, Massachusetts.

Brooks, R. A. & Matarić, M. J. (1993), Real Robots, Real Learning Problems, *in* 'Robot Learning', Kluwer Academic Press, pp. 193–213.

Brooks, R. A., Maes, P., Matarić, M. J. & Moore, G. (1990), Lunar Base Construction Robots, *in* 'IEEE International Workshop on Intelligent Robots and Systems (IROS-90)', Tokyo, pp. 389–392.

Brown, T. A. & McBurnett, M. (1993), Political Life on a Lattice, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 113–126.

Calenbuhr, V. & Deneubourg, J. L. (1992), 'A Model for Osmotropotactic Orientation (I)', *Journal of Theoretical Biology*.

Caloud, P., Choi, W., Latombe, J., LePape, C. & Yim, M. (1990), Indoor Automation with Many Mobile Robots, *in* 'IROS-90', Tsuchiura, Japan, pp. 67–72.

Camazine, S. (1993), Collective Intelligence in Insect Colonies by Means of Self-Organization, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 158–173.

Canny, J. F. (1988), *The Complexity of Robot Motion Planning*, The MIT Press, Cambridge, Massachusetts.

Cassandra, A. R., Kaelbling, L. P. & Littman, M. L. (1994), Acting Optimally in Partially Observable Stochastic Domains, *in* 'Proceedings, AAAI–94', Seattle, Washington.

Chapman, D. (1987), 'Planning for Conjunctive Goals', *Aritifical Intelligence* **32**, 333–377.

Chapman, D. & Kaelbling, L. P. (1991), Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons, *in* 'Proceedings, IJCAI-91', Sydney, Australia.

Chase, I. D. (1982), 'Dynamics of Hierarchy Formation: The Sequential Development of Dominance Relationships', *Behaviour* **80**, 218–240.

Chase, I. D. (1993), Generating Societies: Collective Social Patterns in Humans and Animals, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 174–191.

Chase, I. D. & Rohwer, S. (1987), 'Two Methods for Quantifying the Development of Dominance Hierarchies in Large Groups with Application to Harris' Sparrows', *Animal Behavior*.

Chase, I. D., Bartolomeo, C. & Dugatkin, L. A. (1994), 'Aggressive interactions and inter-contest interval: how long do winners keep winning?', *Animal Behavior, in press*.

Chatila, R. & Laumond, J.-C. (1985), Position Referencing and Consistent World Modeling for Mobile Robots, *in* 'IEEE International Conference on Robotics and Automation'.

Cheney, D. L. & Seyfarth, R. M. (1990), *How Monkeys See the World*, The University of Chicago Press, Chicago.

Cheney, D. L. & Seyfarth, R. M. (1991), Reading Minds or Reading Behaviour?, *in* A. Whiten, ed., 'Natural Theories of Mind', Basil Blackwell.

Clearwater, S. H., Huberman, B. A. & Hogg, T. (1991), 'Cooperative Solution of Constraint Satisfaction Problems', *Science* **254**, 1181–1183.

Cliff, D., Husbands, P. & Harvey, I. (1993), Evolving Visually Guided Robots, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press, pp. 374–383.

Colorni, A., Dorigo, M. & Maniezzo, V. (1992), Distributed Optimization by Ant Colonies, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 134–142.

Connell, J. H. (1990), *Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature*, Academic Press.

Connell, J. H. (1991), SSS: A Hybrid Architecture Applied to Robot Navigation, *in* 'IEEE International Conference on Robotics and Automation', Nice, France, pp. 2719–2724.

Corbara, B., Drogoul, A., Fresneau, D. & Lalande, S. (1993), Simulating the Sociogenesis Process in Ant Colonies with MANTA, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 224–235.

Dallas, J. (1990), Co-operative Search Behavior in a Group of Lego Robots, Master's thesis, University of Edinburgh.

Dario, P. & Rucci, M. (1993), An Approach to Disassembly Problem in Robotics, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 460–468.

Dario, P., Ribechini, F., Genovese, V. & Sandini, G. (1991), Instinctive Behaviors and Personalities in Societies of Cellular Robots, *in* 'IEEE International Conference on Robotics and Automation', pp. 1927–1932.

Darley, V. (1994), Emergent Phenomena and Complexity, *in* R. Brooks & P. Maes, eds, 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', The MIT Press.

DeAngelis, D. L., Post, W. M. & Travis, C. C. (1986), 'Positive Feedback in Natural Systems', *Biomathematics*.

Decker, K. & Lesser, V. (1993*a*), A One-shot Dynamics Coordination Algorithm for Distributed Sensor Networks, *in* 'Proceedings, AAAI-93', Washington, DC, pp. 210–216.

Decker, K. & Lesser, V. (1993*b*), Quantitative Modeling of Complex Computational Task Environments, *in* 'Proceedings, AAAI-93', Washington, DC, pp. 217–224.

Deneubourg, J.-L. & Goss, S. (1989), Collective Patterns and Decision-Making, *in* 'Ethology, Ecology and Evolution 1', pp. 295–311.

Deneubourg, J. L., Aron, S., Goss, S., Pasteels, J. M. & Duernick, G. (1986), 'Random Behaviour, Amplification Processes and Number of Participants: How they Contribute to the Foraging Properties of Ants', *Physica 22D* pp. 176–186.

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C. & Chretien, L. (1990), The Dynamics of Collective Sorting, *in* 'From Animals to Animats:

International Conference on Simulation of Adaptive Behavior', The MIT Press, pp. 356–363.

Deneubourg, J. L., Goss, S., Pasteels, J. M., Fresneau, D. & Lachaud, J. P. (1987), 'Self-Organization Mechanisms in Ant Societies, II: Learning in Foraging and Division of Labor', *From Individual to Collective Behavior in Social Insects* **54**, 177–196.

Deneubourg, J. L., Theraulax, G. & Beckers, R. (1992), Swarm-Made Architectures, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 123–133.

Dennett, D. C. (1987), *The Intentional Stance*, The MIT Press, Cambridge, Massachusetts.

DeScnutter, G. & Nuyts, E. (1993), Birds use self-organized social behaviours to regulate their dispersal over wide areas: evidences from gull roosts., *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 285–309.

Donald, B. R., Jennings, J. & Rus, D. (1993), Experimental Information Invariants for Cooperating Autonomous Mobile Robots, *in* 'Proceedings, International Symposium on Robotics Research', Hidden valley, PA.

Doyle, J. & Sacks, E. P. (1989), Stochastic Analysis of Qualitative Dynamics, Technical Report LCS-TM-418, MIT Laboratory for Computer Science.

Drogous, A., Ferber, J., Corbara, B. & Fresneau, D. (1992), A Behavioral Simulation Model for the Study of Emergent Social Structures, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 161–170.

Dudek, G., Jenkin, M., Milios, E. & Wilkes, D. (1993), A Taxonomy for Swarm Robotics, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 441–447.

Durfee, E. H., Lee, J. & Gmytrasiewicz, P. J. (1993), Overeager Reciprocal Rationality and Mixed Strategy Equilibria, *in* 'Proceedings, AAAI–93', Washington, DC, pp. 225–230.

Ephrati, E. (1992), Constrained Intelligent Action: Planning Under the Influence of a Master Agent, *in* 'Proceedings, AAAI-92', San Jose, California, pp. 263–268.

Erdmann, M. (1989), On Probabilistic Strategies for Robot Tasks, PhD thesis, MIT.

Erdmann, M. & Lozano-Pérez, T. (1987), 'On Multiple Moving Objects', *Algorithmica* **2**, 477–521.

Ferrell, C. (1993), Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators, Technical Report AI-TR-1443, MIT Artificial Intelligence Laboratory.

Fikes, R. E. & Nilsson, N. J. (1971), 'STRIPS: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence* **2**, 189–208.

Firby, R. J. (1987), An investigation into reactive planning in complex domains, *in* 'Proceedings, Sixth National Conference on Artificial Intelligence', Seattle, pp. 202–206.

Floreano, D. (1993), Patterns of Interactions in Shared Environments, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 347–366.

Forrest, S. (1989), *Emergent Computation: Self–Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks*, North Holland, Amsterdam.

Foster, T. C., Castro, C. A. & McNaughton, B. L. (1989), 'Spatial Selectivity of Rat Hippocampal Neurons: Dependence on Preparedness for Movement', *Science* pp. 1589–1582.

Franklin, J. A. & Selfridge, O. G. (1990), Some New Directions for Adaptive Control Theory in Robotics, *in* W. T. Miller, R. S. Sutton & P. J. Werbos, eds, 'Neural Networks for Control', The MIT Press, pp. 349–360.

Franks, N. R. (1989), 'Army Ants: A Collective Intelligence', *American Scientist* **77**, 139–145.

Fukuda, T., Nadagawa, S., Kawauchi, Y. & Buss, M. (1989), Structure Decision for Self Organizing Robots Based on Cell Structures - CEBOT, *in* 'IEEE International Conference on Robotics and Automation', Scottsdale, Arizona, pp. 695–700.

Fukuda, T., Sekiyama, K., Ueyama, T. & Arai, F. (1993), Efficient Communication Method in the Cellular Robotics System, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 1091–1096.

Gallagher, J. C. & Beer, R. D. (1993), A Qualitative Dynamics Analysis of Evolved Locomotion Controller, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press, pp. 71–80.

Gasser, L. & Huhns, M. N. (1989), *Distributed Artificial Intelligence*, Pitman, London.

Georgeff, M. P. & Lansky, A. L. (1987), Reactive Reasoning and Planning, *in* 'Proceedings, Sixth National Conference on Artificial Intelligence', Seattle, pp. 677–682.

Giralt, G., Chatila, R. & Vaisset, M. (1983), An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots, *in* M. Brady & R. Paul, eds, 'First International Symposium on Robotics Research', The MIT Press, Cambridge, Massachusetts.

Gleitman, H. (1981), *Psychology*, W. W. Norton & Co., New York.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Gomez, J. C. (1991), Visual Behavior as a Window for Reading the Mind of Others in Primates, *in* A. Whiten, ed., 'Natural Theories of Mind', Basil Blackwell.

Goss, S., Deneubourg, J. L., Beckers, R. & Henrotte, J. (1993), Recipes for Collective Movement, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 400–410.

Gould, J. L. (1982), *Ethology; The Mechanisms and Evolution of Behavior*, W. W. Norton & Co., New York.

Gould, J. L. (1987), Flower-shape, Landmark, and Locale Memory in Honeybees, *in* R. Menzel & A. Mercer, eds, 'Neurobiology and Behavior of Honeybees', Springer–Verlag.

Gutzwiller, M. (1992), 'Quantum Chaos', *Scientific American*.

Harnad, S. (1990), 'The Symbol Grounding Problem', *Physica D* **42**, 335–346.

Hillis, W. D. (1990), 'Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure', *Physica D* **42**, 228–234.

Hinton, G. E. (1990), Connectionist Learning Procedures, *in* Kodratoff & Michalski, eds, 'Machine Learning, An Artificial Intelligence Approach, Vol. 3', Morgan Kaufmann, pp. 555–610.

Hodgins, J. K. & Brogan, D. C. (1994), Robot Herds: Group Behaviors from Systems with Significant Dynamics, *in* R. Brooks & P. Maes, eds, 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', The MIT Press.

Hogeweg, P. & Hesper, B. (1985), 'Socioinformatic Processes: MIRROR Modelling Methodology', *Journal of Theoretical Biology* **113**, 311–330.

Hogg, T. & Williams, C. P. (1993), Solving the Really Hard Problems with Cooperative Search, *in* 'Proceedings, AAAI-93', Washington, DC, pp. 231–236.

Holland, J. H. (1985), Properties of the bucket brigade algorithm, *in* 'Proceedings, International Conference on genetic Algorithms and Their Applications', Pittsburgh, PA, pp. 1–7.

Holland, J. H. (1986), Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems, *in* R. S. Michalski, J. G. Carbonell & T. M. Mitchell, eds, 'Machine Learning, An Artificial Intelligence Approach, Vol. 2', Morgan Kaufmann, Los Altos, CA.

Huang, Q. & Beni, G. (1993), Stationary Waves in 2–Dimensional Cyclic Swarms, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 433–440.

Huber, M. J. & Durfee, E. H. (1993), Observational Uncertainty in Plan Recognition Among Interacting Robots, *in* 'Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots', Chambery, France, pp. 68–75.

Huberman, B. A. (1990), 'The Performance of Cooperative Processes', *Physica D* **42**, 38–47.

Jaakkola, T. & Jordan, M. I. (1993), 'On the Convergence of Stochastic Iterative Dynamic Programming Algorithms', *Submitted to Neural Computation.*

Jordan, M. I. & Rumelhart, D. E. (1992), 'Forward Models: Supervised Learning with a Distal Teacher', *Cognitive Science* **16**, 307–354.

Kaelbling, L. P. (1990), Learning in Embedded Systems, PhD thesis, Stanford University.

Kephart, J. O., Hogg, T. & Huberman, B. A. (1990), 'Collective Behavior of Predictive Agents', *Physica D* **42**, 48–65.

Keshet, L. E. (1993), Trail Following as an Adaptable Mechanism for Population Behaviour, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 326–346.

Kessin, R. H. & Campagne, M. M. V. L. (1992), 'The Development of a Social Amoeba', *American Scientist* **80**, 556–565.

Kolen, J. F. & Pollack, J. B. (1993), Apparent Computational Complexity in Physical Systems, *in* 'Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society', Boulder, Colorado, pp. 617–622.

Kosoresow, A. P. (1993), A Fast First-Cut Protocol for Agent Coordination, *in* 'Proceedings, AAAI-93', Washington, DC, pp. 237–242.

Košecká, J. (1992), Control of Discrete Event Systems, Technical Report MS–CIS–92–35 GRASP LAB 313, University of Pennsylvania.

Koza, J. R. (1990), Evolution and Co-evolution of Computer Programs to Control Independently-acting Agents, *in* 'Proceedings, Simulation of Adaptive Behavior SAB-90', The MIT Press, Paris, France, pp. 366–375.

Kraus, S. (1993), Agents Contracting Tasks in Non-Collaborative Environments, *in* 'Proceedings, AAAI-93', Washington, DC, pp. 243–248.

Kube, C. R. (1992), Collective Robotic Intelligence: A Control Theory for Robot Populations, Master's thesis, University of Alberta.

Kube, C. R. & Zhang, H. (1992), Collective Robotic Intelligence, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', pp. 460–468.

Kube, C. R., Zhang, H. & Wang, X. (1993), Controlling Collective Tasks With an ALN, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 289–293.

Kurosu, K., Furuya, T. & Soeda, M. (1993), Fuzzy Control of Group With a Leader and Their Behaviors, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 1105–1109.

Laird, J. E. & Rosenbloom, P. S. (1990), Integrating, Execution, Planning, and Learning in Soar for External Environments, *in* 'Proceedings, AAAI-90', pp. 1022–1029.

Langton, C. G. (1989), *Artificial Life*, Addison–Wesley.

Langton, C. G. (1990), 'Computation at the Edge of Chaos: Phase Transitions and Emergent Computation', *Physica D* **42**, 12–37.

Lin, L.-J. (1991), Self-improvement Based on Reinforcement Learning, Planning and Teaching, *in* 'Proceedings, Eighth International Conference on Machine Learning', Morgan Kaufmann, Evanston, Illinois, pp. 323–327.

Lozano-Pérez, T., Mason, M. T. & Taylor, R. H. (1984), 'Automatic Synthesis of Fine Motion Strategies for Robots', *International Journal of Robotics Research* **3**(1), 3–24.

Lynch, N. A. (1993), Simulation Techniques for Proving Properties of Real–Time Systems, Technical Report MIT–LCS–TM–494, MIT.

Lynch, N. A. & Tuttle, M. R. (1987), Hierarchical Correctness Proofs for Distributed Algorithms, Technical Report MIT–LCS–TR–387, MIT.

MacLennan, B. J. (1990), Evolution of Communication in a Population of Simple Machines, Technical Report Computer Science Department Technical Report CS–90–104, University of Tennessee.

Maes, P. (1989), The Dynamics of Action Selection, *in* 'IJCAI-89', Detroit, MI, pp. 991–997.

Maes, P. (1991), Learning Behavior Networks from Experience, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', The MIT Press, pp. 48–57.

Maes, P. & Brooks, R. A. (1990), Learning to Coordinate Behaviors, *in* 'Proceedings, AAAI-91', Boston, MA, pp. 796–802.

Mahadevan, S. & Connell, J. (1991*a*), Automatic Programming of Behavior-based Robots using Reinforcement Learning, *in* 'Proceedings, AAAI-91', Pittsburgh, PA, pp. 8–14.

Mahadevan, S. & Connell, J. (1991*b*), Scaling Reinforcement Learning to Robotics by Exploiting the Subsumption Architecture, *in* 'Eight International Workshop on Machine Learning', Morgan Kaufmann, pp. 328–337.

Matarić, M. J. (1990*a*), A Distributed Model for Mobile Robot Environment-Learning and Navigation, Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory.

Matarić, M. J. (1990*b*), Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation, *in* J. A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press, pp. 169–175.

Matarić, M. J. (1991), A Comparative Analysis of Reinforcement Learning Methods, Technical Report AIM-1322, MIT Artificial Intelligence Lab.

Matarić, M. J. (1992*a*), Behavior-Based Systems: Key Properties and Implications, *in* 'IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems', Nice, France, pp. 46–54.

Matarić, M. J. (1992*b*), Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior'.

Matarić, M. J. (1992*c*), 'Integration of Representation Into Goal-Driven Behavior-Based Robots', *IEEE Transactions on Robotics and Automation* **8**(3), 304–312.

Matarić, M. J. (1993), Kin Recognition, Similarity, and Group Behavior, *in* 'Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society', Boulder, Colorado, pp. 705–710.

Matarić, M. J. (1994), Learning to Behave Socially, *in* 'The Third International Conference on Simulation of Adaptive Behavior'.

McFarland, D. (1985), *Animal Behavior*, Benjamin Cummings.

McFarland, D. (1987), The Oxford Companion to Animal Behavior, *in* 'Oxford, University Press'.

Meier-Koll, A. & Bohl, E. (1993), Time-structure analysis in a village community of Columbian Indians; A mathematically simulated system of ultradian oscillators, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 718–749.

Miceli, M. & Cesta, A. (1993), Strategic Social Planning: Looking for Willingness in Multi-Agent Domains, *in* 'Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society', Boulder, Colorado, pp. 741–746.

Miller, W. T., Sutton, R. S. & Werbos, P. J. (1990), *Neural Networks for Control*, The MIT Press.

Minsky, M. L. (1954), Theory of Neural–Analog Reinforcement Systems and Its Application to the Brain–Model Problem, PhD thesis, Princeton.

Minsky, M. L. (1986), *The Society of Mind*, Simon and Schuster, New York.

Miramontes, O., Sole, R. V. & Goodwin, B. C. (1993), Antichaos in ants: the excitability metaphor at two hierarchical levels, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 790–807.

Moore, A. (1991), Variable Resolution Dynamic Programming: Efficiently Learning Action Maps in Multivariate Real-valued State-Spaces, *in* 'Eight International Workshop on Machine Learning', Morgan Kaufmann.

Moore, A. W. (1992), 'Fast, Robust Adaptive Control by Learning only Forward Models', *Advances in Neural Information Processing 4* pp. 571–579.

Moore, A. W. (1993), 'The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-spaces', *Advances in Neural Information Processing 6* pp. 711–718.

Moravec, H. P. & Cho, D. W. (1989), A Bayesian Method for Certainty Grids, *in* 'AAAI Spring Symposium on Robot Navigation', pp. 57–60.

Muller, M. & Wehner, R. (1988), Path Integration in Desert Ants: Cataglyphis Fortis, *in* 'Proceedings of the Natural Academy of Sciences'.

Mussa-Ivaldi, F. A. & Giszter, S. (1992), 'Vector field approximation: a computational paradigm for motor control and learning', *Biological Cybernetics* **67**, 491–500.

Noreils, F. R. (1992), An Architecture for Cooperative and Autonomous Mobile Robots, *in* 'IEEE International Conference on Robotics and Automation', pp. 2703–2710.

Noreils, F. R. (1993), 'Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment', *The International Journal of Robotics Research* **12**(1), 79–98.

Parker, L. E. (1993*a*), An Experiment in Mobile Robotic Cooperation, *in* 'Proceedings, Robotics for Challenging Environment', Albuquerque, New Mexico.

Parker, L. E. (1993*b*), Learning in Cooperative Robot Teams, *in* 'Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots', Chambery, France, pp. 12–23.

Parker, L. E. (1994), Heterogeneous Multi–Robot Cooperation, PhD thesis, MIT.

Payton, D. (1990), Internalized Plans: a representation for action resources, *in* P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', The MIT Press.

Payton, D., Keirsey, D., Kimble, D., Krozel, J. & Rosenblatt, K. (1992), 'Do Whatever Works: A robust approach to fault-tolerant autonomous control', *Journal of Applied Intelligence* **3**, 226–249.

Piaget, J. (1962), *Play, Dreams and Imitation in Children*, W. W. Norton & Co., New York.

Pinker, S. (1994), *The Language Instinct*, William Morrow and Company, Inc., New York.

Pomerleau, D. A. (1992), Neural Network Perception for Mobile Robotic Guidance, PhD thesis, Carnegie Mellon University, School of Computer Science.

Premack, D. & Woodruff, G. (1978), 'Does the chimpanzee have a theory of mind?', *Behavior and Brain Science* **1**, 515–526.

Read, S. J. & Miller, J. C. (1993), Explanatory coherence in the construction of mental models of others, *in* 'Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society', Boulder, Colorado, pp. 836–841.

Resnick, M. (1992), Beyond the Centralized Mindset; Exploration in Massively-Parallel Microworlds, PhD thesis, MIT.

Reynolds, C. W. (1987), 'Flocks, Herds, and Schools: A Distributed Behavioral Model', *Computer Graphics* **21**(4), 25–34.

Rosenschein, G. & Genesereth, M. R. (1985), Deals Among Rational Agents, *in* 'IJCAI-85', pp. 91–99.

Rosenschein, J. S. (1993), Consenting Agents: Negotiation Mechanisms for Multi-Agent Systems, *in* 'IJCAI-93', pp. 792–799.

Rosenschein, S. J. & Kaelbling, L. P. (1986), The Synthesis of Machines with Provable Epistemic Properties, *in* J. Halpern, ed., 'Theoretical Aspects of Reasoning About Knowledge', Morgan Kaufmann, Los Altos, CA, pp. 83–98.

Rosenthal, T. L. & Zimmerman, B. J. (1978), *Social Learning and Cognition*, Academic Press, New York.

Samuel, A. L. (1959), 'Some studies in machine learning using the game of checkers', *IBM Journal of Research and Development* **3**, 211–229.

Sandini, G., Lucarini, G. & Varoli, M. (1993), Gradient Driven Self–Organizing Systems, *in* 'IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan, pp. 429–432.

Schaal, S. & Atkeson, C. G. (1994), 'Robot Juggling: An Implementation of Memory-Based Learning', *Control Systems Magazine*.

Schmieder, R. W. (1993), A Knowledge-Tracking Algorithm for Generating Collective Behavior in Individual-Based Populations, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 980–989.

Schoppers, M. J. (1987), Universal Plans for Reactive Robots in Unpredictable Domains, *in* 'IJCAI-87', Menlo Park, pp. 1039–1046.

Seeley, T. D. (1989), 'The Honey Bee Colony as a Superorganism', *American Scientist* **77**, 546–553.

Shoham, Y. & Tennenholtz, M. (1992), On the synthesis of useful social laws for artificial agent societies, *in* 'Proceedings, AAAI-92', San Jose, California, pp. 276–281.

Simon, H. (1969), *The Sciences of the Artificial*, The MIT Press.

Singh, S. P. (1991), Transfer of Learning Across Compositions of Sequential Tasks, *in* 'Proceedings, Eighth International Conference on Machine Learning', Morgan Kaufmann, Evanston, Illinois, pp. 348–352.

Sismondo, E. (1990), 'Synchronous, Alternating, and Phase–Locked Stridulation by a Tropical Katydid', *Science* **249**, 55–58.

Smithers, T. (1994), On Why Better Robots Make it Harder, *in* 'The Third International Conference on Simulation of Adaptive Behavior'.

Steels, L. (1989), Cooperation Between Distributed Agents Through Self-Organization, *in* 'Workshop on Multi-Agent Cooperation', North Holland, Cambridge, UK.

Steels, L. (1994*a*), 'The Artificial Life Roots of Artificial Intelligence', *to appear in The Artificial Life Journal*.

Steels, L. (1994*b*), Emergent Functionality of Robot Behavior Through On-line Evolution, *in* R. Brooks & P. Maes, eds, 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', The MIT Press.

Sussman, G. J. & McDermott, D. V. (1972), From PLANNER to CONNIVER–A genetic approach, *in* 'Proceedings, Fall Joint Computer Conference', pp. 1171–1179.

Sutton, R. (1988), 'Learning to Predict by Method of Temporal Differences', *The Journal of Machine Learning* **3**(1), 9–44.

Sutton, R. S. (1990), Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming, *in* 'Proceedings, Seventh International Conference on Machine Learning', Austin, Texas.

Tan, M. (1993), Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents, *in* 'Proceedings, Tenth International Conference on Machine Learning', Amherst, MA, pp. 330–337.

Thrun, S. B. & Mitchell, T. M. (1993), Integrating Indictive Neural Network Learning and Explanation–Based Learning, *in* 'Proceedings, IJCAI-93', Chambery, France.

Tomasello, M., Kruger, A. C. & Rather, H. H. (1992), 'Cultural Learning', *to appear in The Journal of Brain and Behavior Sciences*.

Travers, M. (1988), Animal Construction Kits, *in* C. Langton, ed., 'Artificial Life', Addison–Wesley.

Vander, A. J., Sherman, J. H. & Luciano, D. S. (1980), *Human Physiology*, McGraw-Hill Book Company, New York.

Waterman, T. H. (1989), *Animal Navigation*, Scientific American Library, New York.

Watkins, C. J. C. H. (1989), Learning from Delayed Rewards, PhD thesis, King's College, Cambridge.

Watkins, C. J. C. H. & Dayan, P. (1992), 'Q-Learning', *Machine Learning* **8**, 279–292.

Wehner, R. (1987), 'Matched Filters – Neural Models of the External World', *Journal of Computational Physiology* **A**(161), 511–531.

Weisbuch, G. (1991), Complex System Dynamics, *in* 'Lecture Notes Vol. II, Santa Fe Institute Studies in the Sciences of Complexity', Addison–Wesley, New York.

Werner, G. M. & Dyer, M. G. (1990), Evolution of Communication in Artificial Organisms, Technical Report UCLA–AI–90–06, University of California, Los Angeles.

Whitehead, S. D. (1992), Reinforcement Learning for the Adaptive Control of Perception and Action, PhD thesis, University of Rochester.

Whitehead, S. D. & Ballard, D. H. (1990), Active Perception and Reinforcement Learning, *in* 'Proceedings, Seventh International Conference on Machine Learning', Austin, Texas.

Whitehead, S. D., Karlsson, J. & Tenenberg, J. (1993), Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging, *in* J. H. Connell & S. Mahadevan, eds, 'Robot Learning', Kluwer Academic Publishers, pp. 45–78.

Wiggins, S. (1990), *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer–Verlag, New York.

Yanco, H. & Stein, L. A. (1993), An Adaptive Communication Protocol for Cooperating Mobile Robots, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', The MIT Press, pp. 478–485.