ABSTRACT

Title of dissertation: Guided Self-Organizing Particle Systems for Basic Problem Solving

Alejandro Rodríguez, Doctor of Philosophy, 2007

Dissertation directed by: Professor James A. Reggia
Department of Computer Science

In recent years researchers have shown increasing interest in *swarm intelligence* as a promising approach to adaptive distributed problem solving. Swarm intelligence consists of techniques inspired by nature, especially social insects and aggregations of animals, and even human interactions. They are based on self-organization (a system's overall behavior emerges from the local interactions among its relatively simple components) and are often decentralized and massively distributed. Particle systems are an approach to swarm intelligence that focus on collective movements, and have been used successfully for applications such as computer animation in graphics and control of movements of autonomous robotic vehicle teams. However, particle system techniques have not been applied substantially to problem solving beyond merely collective navigational tasks.

In this dissertation, I present an extension to particle systems that incorporates top-down, high-level control to self-organizing mobile agents, thereby guiding the self-organizing process and making it possible for particle systems to undertake problem solving directed by goal-oriented behavior while retaining their decentralized, local nature. This extended particle system approach is critically evaluated through three experimental studies that are adapted from well-known problems in multi-agent systems: search and collect, cooperative transport and logistics. The results provide evidence that extended particle systems are capable of exhibiting behavior important for distributed problem solving, such as cooperative sensing, division of labor, sharing of information, and developing global strategies through local interactions. They also show that aggregated movements can be utilized to create coordination at different levels and phases of the performance of a task, whether those include navigation or not, making extended particle systems a useful tool in the construction of adaptive distributed systems.

Guided Self-Organizing Particle Systems for Basic Problem Solving

by

Alejandro Rodríguez

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

DEDICATION

A la persona que anelaba

leer esta tesis,

mucho antes de que yo siquiera

soñara en escribirla.

Esta vez no me olvidé de ti...

ACKNOWLEDGMENTS

I would like to express my deep gratitude towards my advisor, Dr. Reggia, for his amazing support and guidance. I also like to acknowledge the members of my advisory committee for their valuable input and insightful comments in the termination of this dissertation.

I am very grateful to my family, specially my mother and sisters, for their continuous support and unwavering faith in me. I also want to thank Grecia for putting up with me during all these years. I cannot but effusively thank Siggi, for his multiple and useful comments, and his invaluable friendship.

Finally, my eternal gratitude goes to the Muses of the Mall, to whose summer liveliness and youth I owe the inspiration for several pages of this book.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Recent years have seen an increase in interest in complex systems, i.e. systems of numerous elements interacting in a non-linear manner that are difficult to model or understand at a global level based only on an understanding of the individual elements [107]. This interest is due in part to raising awareness that many systems in nature are of this kind [44], resisting analysis by traditional methods, but also by the fact that an increasing number of applications in engineering and social sciences resemble such systems [10], given the high number of elements and the complex interactions among them, especially in such fields as transportation and communication networks, social and economic networks, and robotics. Complex systems may differ greatly in their nature, but they share one common aspect: the behavior of the system is difficult to understand no matter how simple the behavior of its parts, even though a global pattern or structure certainly occurs. Although mathematical models exist for the analysis of such systems, designing intelligent complex systems remains an elusive problem, due precisely to this difficulty of inferring how small variations on the conditions would influence the structure of the system, and to the lack of a single, centralized control. In complex system, control is distributed among its many components, potentially dispersed along large spaces, physical or conceptual, and these components need not possess any explicit drive for cooperation or coordination, as is the case in biological systems, where the components are agents (insects, humans, etc.) that usually do not explicitly coordinate their actions.

A method that has proven useful in the design of complex adaptive systems is to take inspiration from such systems in nature. Insect colonies, for instance, are not only complex, but they provide a clear example of distributed problem solving. These colonies can perform several activities, such as path finding and optimization, foraging, nest building, division of labor, despise the limited cognitive abilities of their relatively simple members. Studying and modeling the behavior of these natural systems when collectively performing a task could result in tools for the design of robust, efficient, distributed systems that solve a problem by the use of self-organization, the emergence of global structure through local interactions. In fact, this technique has been successfully employed in the design of algorithms and systems that might perform tasks similar to those found in nature, such as coordinated navigation [81, 82, 100, 105], collective transport [53, 62, 77], and division of labor [21, 85], but also in problems with little biological resemblance, such as numerical optimization [41, 59, 60] or combinatorial optimization [39, 40]. The approach of imitating nature, specifically self-organizing natural systems, is known as *swarm intelligence*, and as the mentioned applications suggest, presents a promising alternative to solve problems whose complexity makes them unattainable or especially difficult for traditional approaches. As nature has developed several strategies to tackle many different problems, several models have been suggested to capture and exploit these strategies, usually with little regard for biological plausibility, as the purpose is to obtain insight in the fundamental properties of such systems and their application in different problems (except in those cases when the artificial systems are developed precisely for the prediction or modeling of the original, natural system).

One type of swarm intelligence approach is *particle systems*. Particle systems consist of collections of very simple entities, or particles, that occupy a position in space and move through it by following simple reactive rules that model "forces" exerted by the environment on the particles. They were originally developed for the modeling of fuzzy objects, or entities of loosely connected parts such as fire, gases and fluids, etc., with the purpose of representing and animating them in computer-generated graphics [79]. By extending these particles into relatively complex objects, with orientation, state and complex geometries, and more importantly, by allowing the particles to react to each other, more interesting structures and global behavior can be achieved. The original extension to particle systems was inspired by the coordinated movement of flocking birds [81, 82], and was developed as a method for the behavioral animation of groups of aggregated animals, such as birds, schools and herds (Figure 1.1). Even though the particles are autonomous, acting simply by constantly reacting to the environment and to other particles, the resulting behavior of the flock or group of particles is easily identifiable by human observers as coordinated, with simulated birds being able to fly in groups in coherent ways, avoiding obstacles, splitting into subgroups when necessary or joining other groups when encountering them. This behavior is clearly self-organizing, being adaptive, completely decentralized, with no single particle controlling the group, and insensitive to the number of particles in the group or the area they occupy [84]. It is

Figure 1.1: A flock of galah and sparrows. *Image taken by user Fir0002 at wikimedia.org and distributed under GNU Free Documentation license 1.2.*

produced by the repeated interaction of particles following three simple rules: avoidance of collisions, velocity matching with neighboring particles, and approaching the perceived center of neighboring particles. All the complex, apparently coordinated behavior is the result of each particle individually applying these rules. It is worth noticing that, as expected from a complex system, the movement of the flock may show seemingly random elements that make it hard to predict, although all of the three rules are entirely deterministic.

The most obvious applications of interacting particle systems lie in the simulation of biological groups, such as aggregations of animals (flocks, schools, herds), human groups (crowds) or autonomous vehicles (cars in a convoy, teams of flying vehicles, etc.). These applications can be for artistic purposes, as in the case of computer graphics, for the study of the behavior of a group, as in human crowds [22, 55], or even as a control system for other applications, as in the case of teams of cooperative mobile robots. In this latter application, and by a large margin, most of the examples found in the literature fall in the category of purely navigational control, that is, the aim of the system is to move a group of autonomous agents between two points while avoiding obstacles and other similar hazards in the environment and while keeping formation. The formation of the agents is a restriction externally imposed on the system for reasons that are not directly related to the task of traveling from one point to another; for instance, a military convoy must move in a given formation to increase sensor coverage. However variate the conditions of the system, in terms of dynamics of the agents, formation requirements and properties of the environment, the main goal of the system remains the same: the coherent navigation between two points. In principle, it seems striking that a technique with such properties as interacting particle systems has not been more abundantly exploited in more general systems. Specifically, it has not been used for problem solving other than purely navigational tasks (although some important exceptions do exist as is discussed on the next chapter). One of the main reasons for this is that the simplicity of the control mechanism of the particles makes it difficult for the particles to exhibit more complex behavior, both individually and globally. That is, purely reactive forces acting in each particle at all times, although responsible for one of the main features of the system, namely its general simplicity, are also limiting in the sense that particles cannot adapt their behavior to widely different conditions in the environment or the system, nor present a varying behavior as may be necessary in pursuing a long-term goal that implies the satisfaction of several intermediary goals. A second, equally important reason comes from the fact that it is difficult in general to design the rules governing each agent that will produce a determined global behavior as agents interact, even when this global behavior is relatively simple and with a single fixed goal over time, such as ‘navigate to point B’, and even more so when the desired global behavior consists of pursuing different global basic goals that may vary with time and depend of each other in complex ways, such as ‘go to point A, search and retrieve an object, protect it, bring it back and inform other teams to stop looking for it’. In the previous example, it seems as if the set of goals could be serialized and pursued independently, although in the general case more complex scenarios arise and different goals may need to be pursued simultaneously by different agents.

In this dissertation, I present an extension to the conventional interacting particle systems that allows them to tackle such scenarios, and reduces the complexity of designing simple rules for multi-goal problem solving systems. This is achieved by introducing mechanisms to guide the self-organization process, which allows the global behavior of the system to be partially directed in a more easily controllable/predictable manner without resorting to actual global control. This extension is based on the concept of hybrid control, which combines the basic reactive behavior normally found in particle systems, with a discrete controller that keeps track of the different goals of an agent and an abstract representation of the environment surrounding the agent, as it is usually found on conventional artificial intelligence systems. Hybrid control approaches has been applied before to particle systems, as it is discussed in the next chapter; however, they have not been used before to build adaptive, distributed, self-organizing problem solving systems based on interacting particle systems. My new particle systems are provided with a layered architecture. The bottom layer consists of the conventional particle system controller, that is, reactive controllers that are capable of handling the continuous, noisy, fast changing environment, and additionally produces global self-organization through the interaction between groups of agents. The top layer comprises an abstract representation of the problem to be solved and the necessary actions or sub-goals, and the relations between them, at a high level of representation, and is capable of changing or adjusting the bottom-layer controller according to these goals. It also keeps an abstract, discrete representation of the current situation of the agent. When introducing this top-layer controller in each agent in the context of particle systems, the agents, even if originally homogeneous, are capable of pursuing different goals or sub-goals, splitting or joining in sub-teams as necessary, affecting with their interactions the state and goals of other agents, which creates a new, higher level of self-organization, essential for the problem-solving capabilities of the system.

Particle systems are mainly known for their capacity to produce coordinated movements, where groups of agents move at unison seemingly as a single entity. However, among the few problem-solving systems based or inspired by particle systems, there is a tendency to disregard or minimize this feature and exploit other properties [35, 59, 74]. One of the aims of the extension to particle systems introduced here is to give new capabilities applicable to problem solving while preserving as much as possible the relative simplicity of the original systems, and specially preserving the coordinated collective movements. In the case studies explored in this dissertation, coordinated movements are not only a sub-product of the self-organization, but fundamental to achieve a higher level of coordination and global patterns, not only on the physical space but in other, abstract areas of the problem solving effort, such as division of labor and division of information. The new system is developed so as to exploit coordinated movements as a vehicle for the implicit communication of information and the self-organization of agents at all levels of the performance of a task.

The proposed methods are studied and demonstrated through three applications of increasing complexity. The first application consists of two problems well known in the context of multi-agent systems, *foraging* [28, 33, 65] and *capture the flag* [2, 6]. Both scenarios require, independently, the cooperation and repeated interaction of mobile agents with limited sensing and acting capabilities, which makes these problems useful in the study of different issues in systems involving cooperative agents. Additionally, they require the performance of different sub-tasks and the pursue of different, intermediate sub-goals, such as searching, recruiting, exploiting, protecting, stealing, that present complex interactions as they need to be pursued simultaneously by a team of homogeneous agents. The second problem is an extension to another well known problem in systems of mobile agents, cooperative transport [62, 66, 96, 109]. As the previous problem, it exhibits biological inspiration, but also it possesses clear practical applications as systems of cooperative mobile robots are currently an area of intensive study. In addition to the issues usually found in cooperative transport problems, dealing with the navigation of an object around obstacles while being carried by agents with limited and imperfect knowledge of the world, this extension requires agents to distribute several objects to different locations according to certain demands not known to the agents beforehand. This converts the scenario into a logistics problem that demands of the system to achieve shared global knowledge and to develop a shared global strategy, both of which must be achieved exclusively through the local interactions among the agents. The final problem is a full logistics problem consisting of the routing of items of several types through a network of stations with specific demands and capabilities. This problem presents qualities of both routing, common in computer networking and other applications, and scheduling and flow in graphs, which is family of problems with several real life applications. The logistics problem presents additional issues, as the solution, a scheduling and routing strategy, lies in the collective behavior of groups of agents with no explicit means of communication and requires the distributed learning by part of the agents of the conditions of the world and the effect of their actions on the environment.

1.1 Contributions

As a result of the work described in this dissertation, namely an extension to interacting particle systems, and its application to different problems of varying degree of difficulty and varying properties, the following general contributions are presented:

- Introduced the use of top-down control mechanism using state-transition and memory/goal style controls into reactive interacting particle systems, allowing distributed problem solving through the cooperation of autonomous agents without the requirement of explicit communication. By this architecture, self-organization is achieved using coordinated movements, while the individual behavior is affected via top-down control, which guides the system-wide behavior.

- Established through experimental simulations that the resulting guided self-organizing process was capable of solving cooperative search-and-retrieve problems effectively; demonstrated that collectively-moving agent teams worked more efficiently than independently moving teams, in part due to the ability of agents to perform teammate recruitment to collectively accomplish a task.
- Extended the basic guided self-organizing approach to solve more complex collective transport problems in environments with obstacles, a problem that strictly requires the coordinated work of several agents with limited communication capabilities, demonstrating the ability of this approach to handle global strategies. Additionally, such extension caused agent teams to exhibit dynamic task division and information sharing over areas largely beyond the range of individual sensors. All of these properties could be used in in real-world robotic applications and multi-agent systems.
- Added basic learning and inter-particle communication to support adaptive distributed solving of logistic problems in transportation networks, showing the ability of guided self-organizing particle systems to perform dynamic dis-

tributed non-linear optimization.

1.2 Organization

The rest of this dissertation is organized as follows. In Chapter 2, a brief discussion is presented of the main concepts employed on this dissertation and of the relevant previous work that served as inspiration for the ideas here or that are related in a significant way. Chapter 3 presents the extension to particle systems introduced in this dissertation accompanied by global ideas on its application. It then describes the first of three problems, foraging and capture the flag, that serve as experimental test beds of the application of extended particle systems. This chapter describes the problem and presents a concrete example of the application of the system, studying its properties through a series of experiments and comparisons, followed by discussion of the results. Chapter 4 and 5 follow a similar structure to Chapter 3, presenting the problems of cooperative transport and logistics, respectively, with the corresponding solution using the proposed system and discussion of experiments and results. Finally, Chapter 6 presents a general discussion of the findings of this dissertation, including limitations and future work.

Chapter 2

Swarm Intelligence and Particle Systems

As mentioned in the previous chapter, swarm intelligence provides a promising technique for the design of complex systems, and from these techniques arises interacting particle systems, which is the special focus of this dissertation. This chapter presents a brief introduction to swarm intelligence, particle systems and some related concepts, and includes a discussion of relevant previous work that has been done in the area of swarm intelligence as a problem-solving strategy and in previous efforts to extend particle systems.

2.1 Swarm Intelligence

The term *swarm intelligence*, initially introduced by Beni in the context of cellular robotics [14, 16, 17, 18], refers to a collection of techniques inspired in part by the behavior of social insects, such as ants, bees, termites, etc., and of aggregations of animals, such as flocks, herds, schools, and even human groups and economic models. These swarms possess the ability to present remarkably complex and "intelligent" behavior, despite the apparent lack of relative complexity in the individuals that form them. These behaviors can include cooperative, synchronized hunting, coordinated raiding, migration, foraging, path finding, bridge construction, allocation of labor, and nest construction. Recent discoveries [37] have led investigators to the belief that such behaviors, although in part produced by the genetic and physiological structure of the individuals, are mostly caused by the *self-organization* of the systems they form [5, 19], that is, out of the direct or indirect local interactions between the individuals, the collective behavior emerges in a way that may have the appearance of being globally organized, although no centralized control or global communication actually exists. It is precisely this self-organization that artificial swarm intelligence systems try to achieve, by infusing the components, homogeneous or heterogeneous, of a system with simple rules. The interactions between these components lead to the system-wide behavior.

In principle, this self-organization may cause the behaviors of swarm intelligence systems to be hard to predict (and hence design), since the result of the interactions between possibly thousands of potentially heterogeneous elements could not be readily understandable. However, the ability to deal with this complexity and create organized patterns out of it is also the fundamental advantage of swarm intelligence. In contemporary real life problems, where systems may consist of several thousands of interacting components, traditional techniques may become prohibitively expensive, or a global, centralized control may be unavailable given the absence of global information or the cost of global communication. In addition to these benefits, the emergent-control characteristic of swarm intelligence systems tends to homogenize the components, i.e., there is no *a priori* leader governing the systems, which, when combined with the high number of available components, introduces a form of redundancy that makes a system highly fault tolerant; when some of the components fail, the self-organizing process adaptively adjusts to recreate the correct pattern of behavior using the remaining components.

Finally, robotic systems are a field that can particularly benefit from swarm intelligence, since the manufacturing of several simple units or robots is considered to be far cheaper and more efficient than the elaboration of a single, highly sophisticated robot, assuming the simple robots could cooperatively perform the same task as a complex one[4, 15, 20].

2.2 Collective Movements in Particle Systems

Of particular interests for our study is the work of Reynolds [81, 82, 83, 84], as one of the earliest and most clear examples of the simulation and control of large crowds of mobile agents by designing simple interactions among them. This work introduced *interacting particle systems*, that is, collection of simple entities or oriented particles that move through a continuous space directed by forces exerted upon them by the environment and other neighboring particles, with the interactions among the particles restricted to be of reactive and local nature. Interacting particle systems have since then become a common technique in swarm intelligence [41, 54, 59, 80, 100]. Reynolds' intention was to create a realistic simulation of a flock of birds for purposes of computer animation. Previous attempts consisted in their majority of scripting the behavior of each individual bird, which required of considerable time and effort by part of the animator, besides imposing a high cost in maintenance, since performing small changes to the animations requires re-scripting many or all of the birds. By taking inspiration from birds themselves, that presumably do not collectively plan their paths in advance, Reynolds designed a model where each bird autonomously decides its behavior based in part on the behavior of its neighbors.

This model is itself an extension of particle systems [79], usually employed in computer graphics to simulate fire, gases, liquids, and others "fuzzy" objects. By extending each particle to a more geometrically and behaviorally complex entity, a simulated bird, or *boid*, a flock or other aggregation of animals such as schools and herds can be represented and handled as a single object. The key innovation, however, is to allow these particles to locally interact with each other allowing the collective behavior to arise from such interactions (and not from a leader or external controller). These interactions are meant to imitate the actions that actual birds would take to coordinate their flight. In particular, Reynolds designed three simple rules (depicted in Figure 2.1) to be followed at all times by each boid:

1. Collision avoidance: Stay away from obstacles and nearby boids.

2. Alignment: Match the velocity of nearby boids.

3. Cohesion: Approach the center of the (local, perceived) flock.

These rules, executed in a purely reactive fashion, and combined in order of priority, create the illusion of a centrally coordinated group of entities that move in a united way while quickly reacting to the environment (Figure 2.2). As opposed to traditional approaches, this behavioral simulation where the movement of the flock emerges from the individual movements of each boid, allows for the animation of large flocks with little need for human intervention or a central controller. Given

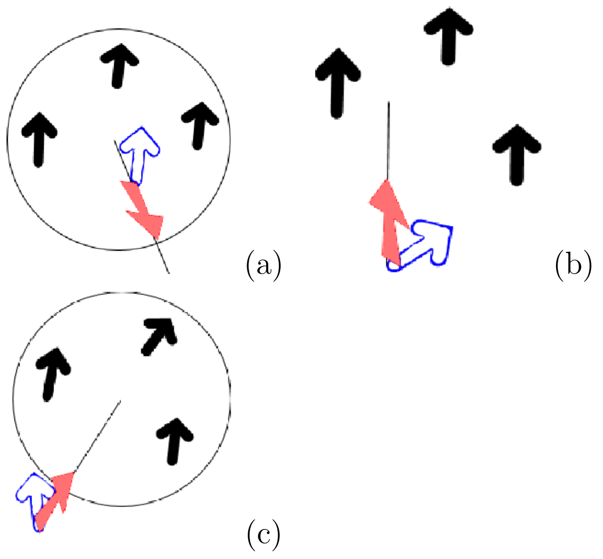


Figure 2.1: Basic forces governing the movement of a boid. A boid is represented as an arrow (the main boid of interest being hollow while neighbors are represented as black, solid arrows). A light arrow indicates the direction of the resulting force over the boid. (a) Collision avoidance: the boid moves away from its perceived neighbors (b) alignment: boid changes direction to match the velocity of neighbors and (c) cohesion: boid moves towards the center of the perceived flock.

that each boid is autonomous and interacts exclusively with nearby objects, the process is also highly parallelizable, and distributed implementations can simulate and display thousands of boids at high frame-per-second rates [84].

On the other hand, by allowing the self-organization of the boids, some of the control is lost, and the animators may find difficulties in achieving pre-determined movements or patterns from the simulation [81]. Unlike the hand-crafted simulations where each element is scripted to the desired behavior, in this case boids will react in manners that result difficult to predict, both individually and system-wide.

Although initially designed for particles, this method of behavioral simulation works independently of the complexity of the dynamics or geometry of the entities. Several examples are available on hearding or similar coordinated, aggregated

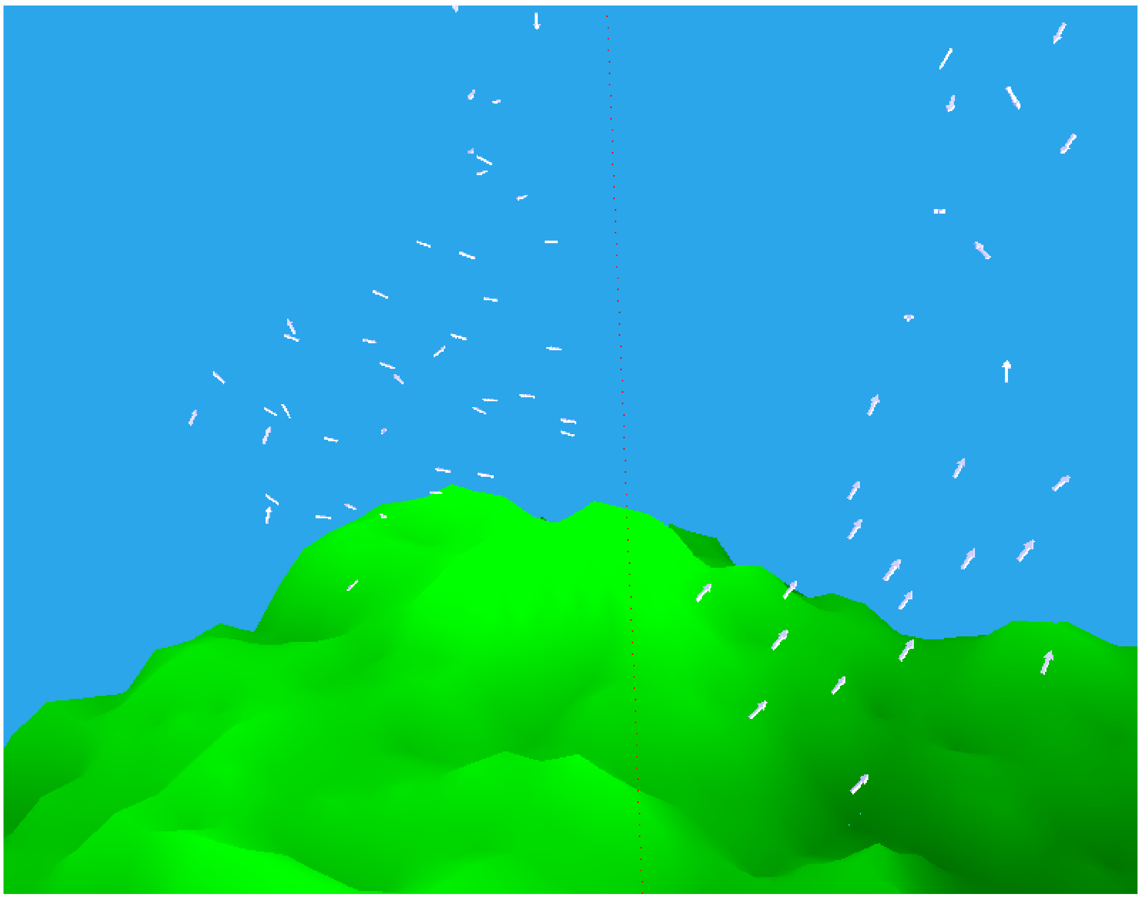


Figure 2.2: Agents moving collectively in a 3D world. Two groups of agents, represented as white arrows, totaling about a hundred, move freely over a 3D world, resembling the movement of schools. The movement of each agent is completely deterministic.

movements in systems with significant dynamics using essentially the same model for behavior control [7, 11, 46, 51, 56, 108].

By varying the behaviors or rules which govern the boids, different types of aggregated motion can be achieved. Later work has extended this basic model with different degrees of realism and complexity. In the realm of simulating aggregations of animals, the physics of the environment and biologically plausibly behaviors have been incorporated. For instance, Tu [100] models the physics, perception mechanisms, means of locomotion and behavior of fish to achieve a realistic graphical simulation. In her model, fish possess a set of predetermined behaviors, such as eating, mating, wandering, schooling, etc., that each fish can switch to according to the strongest intention of the fish as determined by its intention generator, which process stimuli from the environment. This switching between behavior routines provides the fish with a richer, more goal-oriented total behavior. However, notice that fish remain basically stateless and reactive as their goals or actions at a given time have no direct influence in determining subsequent goals.

Helbing et al. introduced a different but similar model [55], also based on particle systems, that combines a set of independent forces varying with the velocity of the particle to achieve trajectories realistically similar to that of human crowds. This model and its variations have been employed for the simulation of human crowds during panic evacuations, traffic jams, and other situations of 2D flow of interacting particles [22, 97].

2.3 Problem Solving

The most obvious applications of models of collective movements based on swarm intelligence and particle systems lie in the simulation of biological groups, either for their study, as in the case of human crowds, or for artistic purposes, such as in schools and flocks. However, it is also possible to apply such models for the control of problem solving systems, where groups of agents, physically mobile or informational, perform a task autonomously but cooperatively, and where the large number of agents or restrictions on access to global information or communication make self-organization the most viable option.

These systems usually consist of collections of unmanned vehicles or teams of cooperating autonomous robots. The focus of this section is on systems that perform some task beyond navigation in formation, as examples of the latter are numerous but less interesting for the purposes of this dissertation [11, 49, 91, 92, 105].

The case of swarming unmanned aerial vehicles (UAVs) presents fertile ground for the application of swarm intelligence given that current systems usually require several human operators for each UAV, making it very ineffective and costly to coordinate multiple UAVs. Swarm intelligence can be used to automatically achieve this coordination, not only on navigation, but also on performing higher-level tasks. One such example is the work of Parunak et al. [72, 73, 74, 86], where multiple autonomous UAVs use virtual, digital pheromones to mark areas of interest in the terrain, for instance already visited areas that need not to be revisited or areas where a target has been sighted, and communicate these pheromones to nearby UAVs, creating a partially shared map of the global area. These same pheromones allow UAVs to recruit other teammates for specific goals, for instance the creation of predetermined formations around objects of interest to achieve the optimal configuration of an array of sensors for purposes of collaborative sensing.

Another interesting example of the control of coordinated mobile agents is presented by self-assembling robots [35, 52, 57]. Self-assembling robots can physically attach to each other, forming a compact cluster that act literally as a new, single robot of a different shape and size but whose components are still controlled individually by each of the member robots. This kind of structure presents challenges both before the assembly, when robots decide to create or join the cluster, and after the assembly, when the cluster of robots needs to move as a single entity. The advantages of such procedures lie in the new properties of the assembled robots; for instance, a group of small robots could self-assemble into a long chain to bridge over an obstacle that no individual robot could step over. Not withstanding the complexity of the problem, simple reactive controllers can be evolved for the individual control of each robot, even real robots with limited sensing and communicating capabilities, that produce through self-organization a coordinated behavior of the cluster, effective for instance to avoid obstacles [98, 99]. More interestingly, the same kind of reactive, autonomous controllers can be evolved to cooperatively transport a heavy prey in a straight line between two points [53], assembling around the prey into groups of pushers or pullers as necessary to exert the required force and control over it while keeping visibility of the goal.

A different application, involving the dissemination of information, is presented by Agassounon [1]. In his work, a team of mobile robots acts as a network of mobile sensors whose task is to detect global conditions on the environment by individually sensing local areas and communicating with nearby robots. In particular, the robots attempt to count the number of seeds, objects of a particular color and size, in an enclosed arena by random wandering around it and counting the encounters with the seeds. Although the teams do not use coordinated movements, they use information exchanges with neighbors as local interactions to increase the accuracy of the global estimate, held by no single robot but as a property of the swarm.

2.3.1 Particle Swarm Optimization

Beyond the realm of mobile agents, the concept of interacting particle systems and coordinated flow has found important applications. Among them, the particle swarm optimization algorithm deserves special mention as the most widely known and successful of such applications. Initially inspired by Reynold's flocking model described above, the particle swarm optimization algorithm [41, 59, 60] extends particle systems to high-dimensional abstract/cognitive spaces based on a social model of the interactions between agents. The inspiration for this model is that people are influenced by the opinions and beliefs (position in a cognitive space) and other properties of their acquaintances. In this sense, agents "moving" through an abstract, high-dimensional cognitive space, are viewed as changing their beliefs according to those of their neighbors.

In particle swarm optimization, a group of agents or particles is initially spread out in the problem space of a function to be optimized. Agents keep track of the best position they have individually found, *pbest*, and also of the best position found by their neighbors, *gbest*. An agent's neighborhood is not defined in terms of the dynamically changing distance in the problem space but rather in terms of social networks defined *a priori* between the agents. At each time-step, agents accelerate toward their best remembered position and also toward the best position of their neighbors. The factor *pbest* drives an agent in the direction of its best memory, while *gbest* can be seen as social norm, guiding all agents to the past location that has been known overall to work best. The velocities pointing toward *pbest* and *gbest* are weighted by independent random numbers, the rationale for randomness being in part that it is hard to decide which one to weight more.

Variations of the algorithm have been obtained by making different definitions of the neighborhood of agents, from 'local' neighbors defined *a priori* as sets of agents conceptually adjacent to each other, to 'global' neighborhoods (every agent is adjacent to every other).

As in the flocking model discussed in the previous section, agents move through the space based on a combination of simple rules which determine at each step the new velocity of the agent according to interactions with other agents. Probably the most important difference lies in the fact that agents move through a N-dimensional, abstract space, and the peculiarities of the movement of the agents (whether they flock or not) are ignored in favor of making agents satisfy a given goal (finding a function minimum).

Particle swarm optimization has been found to be useful in a wide variety of non-linear optimization problems [42, 43, 71], and it is a schema relatively easy to apply given the few parameters which need to be set. However both flocks and particle swarms have in common that the simplicity of the agents and the simple rules which control them are hard to adjust to perform more complex tasks.

2.3.2 Networks and Combinatorial Optimization

Another successful example of the application of swarm intelligence to problem solving comes from imitation of the mechanism used by ants for path-optimization

[13, 26]. When presented with multiple paths between the colony and a food source, ants will eventually settle on the shortest path. They do this without knowledge of the entire path or paths by any individual ant, but by collectively marking the environment with chemical signals, pheromones, that other ants can recognize, creating a form of communication by the modification of the environment, known as stigmergy. The idea of several entities collaborating through stigmergy inspired a meta heuristic for solving problems of combinatorial optimization [39, 40]. The key points of this heuristic could be briefly summarized as having a set of agents that individually take decisions probabilistically, and create marks based on the appropriateness of these decisions that can later be used by the same and other agents to take later decisions.

This heuristic has been successfully applied to problems like the Traveling Salesperson Problem [40], which is a well-known study case for combinatorial optimization, but has been more commonly applied, along with similar algorithms, to routing, load balancing and related problems in networks [29, 38, 39, 87, 88, 93].

2.4 Past Work on Extending Particle Systems

For the study of coordinated movements in teams of agents, the method introduced in this dissertation consists of simplifying and abstracting swarms of agents by treating them as particle systems [79]. As mentioned before, the use of particle systems for behavioral simulation of groups of mobile agents was first introduced in 1987 by Reynolds [81], and has been successfully employed ever since in multitude of applications involving flocking and/or formation control, both in simulated agents and real-world robotics. This success is due to the simplicity of such systems and their capability for representing and generalizing to a wide variety of actual systems. Maybe more importantly, this success is also due to the complex and organized group behaviors that can be achieved, in a way that is both efficient and robust.

However, the same intrinsic simplicity of particle systems can become a disadvantage when modeling more complex problems that required or seem to require higher levels of control, this is, problems where the required global behavior cannot easily be achieved by static combinations of low-level behaviors. For this reason most "pure" particle-system-based control systems have remained in the field of computer animation.

Nevertheless, the attractiveness of such systems has led researchers to attempt to enrich particle systems and behavior-based flocking by expanding them or incorporating them into more flexible, controllable architectures, while preserving their most important features, among them decentralized, local control and self-organization. A common approach is to incorporate behavioral flocking into other successful architectures well-known and commonly used in robotics. Those architectures may range from low-level, emerging-behavior styled systems, in tone with the self-organizing nature of behavioral flocking itself, to more traditional approaches where system designers retain a higher control of the global behavior of the agents.

On the first end of the complexity spectrum, an example of the approaches used is present in the form of dynamics of higher complexity [56]. Of particular interest is the case of the use of neural networks (see for instance Baldassarre [9]). In such systems, neural networks, mapping the sensory input of a mobile agent to the output of the actuators, are typically trained through evolutionary methods. In this case the flocking behavior and the goal-oriented behavior are partially specified in the fitness function, and encoded in the structure and parameters of the network itself. This has the advantage of allowing the rapid, reactive response of neural controllers, often required in real-time robotics, and also of giving the system the freedom to evolve the solution, which in principle may utilize any behavior the network is capable of representing as long as it satisfies the specified goal. However, these systems also share some of the weaknesses of the original behavioral flocking based on particle systems. In particular, although neural controllers are capable of more than purely reactive behavior, the resulting systems remain simple in the nature of the problems they can solve, at least in practice as found in the literature.

On the other end of the complexity spectrum lie systems featuring full-fleshed planners and global control. Among them, the work by Yamashita [109] shows the amount of coordination that can be achieved, for instance when moving objects of arbitrary shape through narrow spaces. This approach works well, in simulation at least, by using global control. Those systems, however, retain almost none of the advantages of the original flocking systems, requiring both global and centralized control, although not necessarily off-line processing.

In the middle between these two extremes can be found several hybrid systems. Those systems often employ multi-layered architectures to combine the long term, goal-oriented control of high-level controllers with the speed, adaptability and robustness in the presence of noise and real-world uncertainty of low-level controllers, and in particular the self-organizing, distributed control of particle systems. Typical examples of this approach are two-level architectures that combine a finite state machine taking higher level decisions about the desired behavior of the agents, and low-level, reactive dynamics that implement the desired behavior. One such example is presented by Reynolds [83], in an application that attempts to simulate the behavior of pigeons in a park, for purposes of animations used in movies or video games. In this case the low-level behaviors are decomposed into basic units (avoid obstacles, move towards neighbor, etc.), that can be combined to achieve more complex behaviors. Even the same group of basic behaviors, combined in different ways, will produce different results, which provides the system with ample flexibility based only in a small set of basic behaviors. The specific combination of behaviors desired is specified in the state machine, in the form of a particular combination for each state. This allows the simulated pigeons to display a range of "natural" behaviors and reactions, such as to flee from pursuers, be drawn towards food, etc. A more application-oriented use of the same ideas can be found, for example, in the work of Parunak et al. [72], who use essentially the same concepts in the context of unmanned autonomous vehicles. The simulated vehicles perform cooperative sensing, arranging themselves into the optimal formation required around the object or area to be sensed. Notice that in both cases, however, there is little room for interaction between agents in different states, or for actions performed in one state to have a direct influence in subsequent states, or even for the sequencing of states to be meaningful in the pursue of other goals (in both cases, the state machines are such that there is a transition between virtually every pair of states). This results in states being isolated units, that greatly simplify what the global behavior of the agents can be when observed over time. However, it is worth remarking that this is not necessarily due to the two-level architecture, but to the nature of the intended applications, that might not require nor benefit from higher complexity.

Those basic behaviors or behavioral units do not need to be combined in a single set according to the state of the agent. As shown by Matarić [67], the resulting higher level behavior can be again combined with others behaviors, in increasingly complex level to form a full hierarchy of behaviors. A simple set of rules can then be used to activate those behaviors needed, in a similar fashion to what is done with deterministic finite state machines.

Higher-level hybrid architectures have also being successfully used in the context of multi-agent systems. For instance see the work of Chaimowics et al. [31], who use low-level behaviors to coordinate the movement of a group of agents, while higher-level controllers, such as a planner, decide the general course of action. The novelty in this case lies in the fact that the system retains its distributed properties by making a single agent decide the plan and communicate it to the rest of the team, but the leader agent can switch roles with following agents at any moment by a collective process of decision making. Once again, similar as the systems featuring global control, the self-organizing nature of flocking and swarming is relegated to the minor role of navigation control while the high-level coordination of agents is effected by more complex, traditional approaches.

In addition to combining the basic swarming or flocking dynamics into more complex architectures, the basic system has also been extended by incorporating other ideas and features that increase its capacity and applicability to particular problems. One useful behavior often desired is for the mobile agents to achieve a given formation as accurately as possible, while dealing with an unpredictable environment and noisy information. An approach commonly used to solve this problem is the dynamic assignment of roles to each member of the flock/convoy [7, 46]. In this way, agents use the low-level dynamics only to keep a relative position with respect to certain, specified neighbors while navigating the environment. A large range of formations can be thus robustly achieved both in simulation and real robots even while navigating open terrain.

Although role assignment in the works discussed above has been used mostly for formation control, higher level uses are also common within the context of coordinated movements. For instance, Vail et al. [102] describes a system where robotic agents use an auction-based role assignment scheme for strategic roles in a soccer game (attacker, defender, etc.). Once the roles have been determined, the robots use potential fields to determine their desired positions and their movement towards them.

Potential fields and gradient following has been used in swarms of mobile agents that perform cooperative sensing [75], for instance for intruder detection, in systems that employ a form of communication inspired in insects, communication through pheromones. Theses agents broadcast signals whose direction and intensity neighbors can perceive in order to share information about the environment. Other extensions to the basic particle-system model include the combination of local and global information (about target destination, path to follow, position of neighbors, position of obstacles, etc.) in different degrees, which has shown that the optimal combination depends on the particular application and usually lies somewhere in the middle between pure local information and total global information [70].

2.5 Discussion

This chapter presented a brief introduction to the main concepts of swarm intelligence and its potential for distributed problem solving. Among the techniques usually employed in swarm intelligence, interacting particle systems stand out in part by their ability to imitate biological systems of coordinately moving entities, but also by their current lack of applications in other contexts and problems. Even though the local interaction between particles is a powerful mechanism to create self-organized behavior, few applications have been found beyond the realm of formation keeping in mobile agents.

Although particle systems have been extended before, previous extensions have not been general enough to exploit particle systems as a problem solving method rather than a navigational control mechanism, and those few that have put emphasis on problem solving have for the most part disregarded the ability of particle systems to create coordinated motion. The extension to particle systems introduced in this dissertation and explained in the next chapter tries to at least partially cover this gap by creating particle systems with greater problem solving capabilities.

Chapter 3

Task Division and Sub-Team Self-Organization

Interacting particle systems provide a powerful mechanism for the modeling of coordinated collections of mobile agents, or particles, due to their self-organizing properties. However, this same self-organization and lack of global, centralized control causes the behavior of such systems to be especially difficult to predict or design. This chapter presents an architecture that tries to guide the self-organization process, allowing the resulting particle system to not only present a richer, more flexible set of behaviors, but also to supply an external entity, the system designer, with the means to influence this self-organization and therefore the global behavior of the system, without sacrificing the intrinsic simplicity of the particles or adding centralized, non-local controls. The core idea of this architecture is derived from hybrid control and consists basically of enhancing the particles with a top-down control mechanism, based on finite state machines, used in combination with the traditional reactive behaviors commonly found in particle systems. Specifically, the hypothesis is that by giving the normally purely reflexive agents found in particle systems a few behavioral states, a simple finite state transition graph that governs state changes, and a simple memory of the locations of significant objects that are encountered, the resulting agent team would have the ability to collectively solve resource locate-and-collect problems. In this scenario, individual behaviors would be implemented by letting each state of an agent be associated with a different goal and with a corresponding set of parameters that influence the individual agent's movements. This effectively couples the collectives' goals to different movement dynamics. Under such conditions, where state changes are triggered by environmental events and the states of other nearby agents in a way that retains the local nature of information processing in particle systems, one would anticipate the emergence of problem-solving abilities by an agent team as a whole. This hypothesis is tested and studied in this and the following chapters.

In this chapter, a concrete practical example is presented as experimental evaluation for the application of the architecture previously described. In order to study the properties of such architecture, combining the self-organizing properties of particle systems with the general problem solving capabilities of discrete, top-down style control, this chapters presents a problem crafted out of the combination of two well known problems in the study of multiple mobile agents: capture the flag and foraging. The resulting problem may possess limited real world applicability *per se*, but it involves issues and properties commonly present in multi-agent systems, and thus it provides an excellent scenario for the study of problem solving particle systems. In particular, this problem demands a level of coordination capable of producing a team of agents to split or join in sub-teams and divide labor among them according to varying conditions.

The series of experiments presented in this chapter not only show the ability of the system to solve a problem previously beyond the scope of the particle systems, but also suggest that coordinated, self-organized movements can greatly contribute to the spread of information about the enviroment and the task being performed, even without explicit communication, and serve as a tool in the creation of self-organization beyond the navigational level.

3.1 Agent Architecture

As seen in the previous chapter, one of the most common architectures employed in the development of hybrid control systems is the layered architecture, whether the systems involve particle/flocking components or not. The layered architectures, initially presented by Brook in 1986 [24] for robotic systems, provide a useful mechanism for the combination of low level, fast, robust behaviors that react directly to the environment, with higher level behaviors that act according to the agents goals. The most common among these is the three layer architecture [25, 48], composed of a bottom layer of basic reactive behavior, a medium layer for the sequencing and prioritizing of those behaviors, and a top layer for long term goal pursuing.

In the case of particle systems and flocking/cooperatively-moving agents, the basic prioritizing and combination of the lowest level behaviors is usually considered to be performed by the bottom layer, with the medium layer effecting the sequencing and/or switching of these behaviors in the medium term. The top layer is often missing, in part because the applications regularly considered do not require a higher level of abstraction and decision making, in part because this higher level usually involves shared global information, which is against the paradigm and

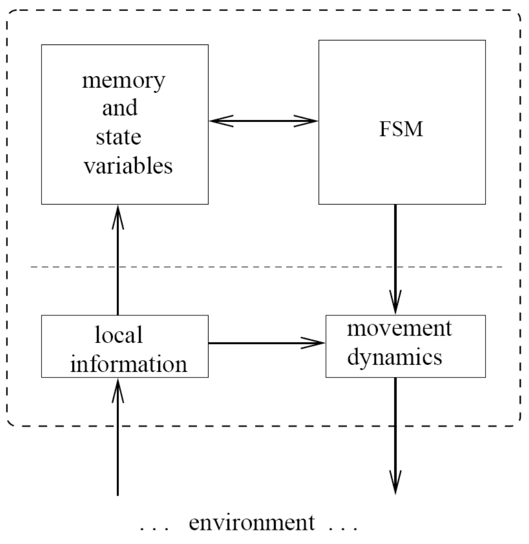


Figure 3.1: High-level view of the agent architecture. The two lowest boxes represent the reactive movement dynamics typically found in past particle systems. The two upper boxes represent the goal-directed mechanisms added to control behavior, including movements. FSM = finite state machine.

sometimes simply not feasible. Therefore, these types of systems often present a two-layer architecture. This is the architecture we adopted as a base for the framework developed and employed in this work, and it will be explained in detail in what follows.

Figure 3.1 shows a scheme of the basic architecture used throughout this dissertation. At the bottom layer of the agent control architecture lies a set of low level dynamics that control the reactive behavior of the agent based on local information. These low level dynamics receive any necessary information from the environment, as gathered by the sensors of the agent, and are assumed to be able to process/synthesize the information into whatever format is required. For simplicity, the information returned by the sensors is already processed into a higher level of abstraction, for instance sensors return the position of a neighboring agent, as opposed to raw sensor data (camera image, infrared sensor activation, etc.). The bottom layer takes input solely from the local environment at a given instant, and outputs a corresponding action, in terms of the basic capabilities of the agent (accelerate in such direction, grab, etc.), based on the immediate goals of the agent and current (local) state of the environment; those basic behaviors are similar to those used in previous works in particle systems and behavioral flocking. The top layer consists of a simple memory and a finite state machine that directs agent behavior in a top-down fashion, modifying the movement dynamics used over time. For example, if the FSM decides that it is time for the agent to go to a particular location, or home, to refuel, it will switch to the state *homing* and provide the bottom layer with the target location of its home destination. The bottom layer will then determine at each step the steering actions needed to properly navigate from the current location to the home. Since the bottom layer is mostly reactive, it can temporarily override the long term goal of going home for a more pressing need, such as avoiding another agent or obstacle. Notice that the homing behavior may be produced by a combination of basic reactive behaviors, and it is the state of the finite state machine who determines these combinations at any given instant.

To be more specific, at any time each agent is in one of several mutually exclusive states. These states and the transitions among them are represented with a finite state machine. The agent's current state determines which set of parameters for the low-level reactive controller currently drives its movements and behavior. As mentioned before, the low-level dynamics that guide the agent through the environment are inspired by earlier work [81, 82]. Movements are governed by a set of individual influences (avoiding an obstacle, staying with the flock, keeping a minimum distance from other agents, etc.) that produce an instantaneous acceleration determined by a desired velocity vector. The individual influences are combined a a non-linear summation of the velocity vectors. By changing with individual influences are combined and their relative weights in the summation, a large variety of movement behaviors can be implemented, each one associated with a different state or goal.

The finite state machine performs these changes of state also based on local information. As the basic behaviors, it receives summaries of the state of the environment neighboring the agent from the sensors of the latter to determine when an event has been triggered that requires the agent to change its behavior. As opposed to the basic behaviors, the state machine uses more general information regarding the general conditions around the agent, that is likely to change less rapidly than the instantaneous information to which those behaviors react. Examples of these kinds of information might be number of team-mates surrounding an agent, the presence of an object in its vicinity, etc. Also, the state machine can switch to a different state according to infrequent or one-time events that may represent the achievement of a goal by the agent, for example reaching a certain destination.

3.1.1 Agent Capabilities

The sensing capabilities of an agent are limited by its neighborhood size. The neighborhood of an agent is a circular segment in front of the agent, defined by a given angle and radius. All objects inside this neighborhood are visible to the agent. As mentioned before, the sensors of the agent perceive the relative position and velocity of an object, and transmit this information in vectorial form to the behaviors. These sensors are also capable of distinguishing between different kind of objects. Thus, an agent can tell if the agent in front of it is a team-mate or a rival, for instance. However, since all agents in a team are homogeneous, agents cannot distinguish between two agents of the same team, or two objects of the same nature.

The radius and angle of the neighborhood are crucial in determining the characteristics of a given behavior. Therefore, to achieve greater flexibility, different radii r_i and angles a_i are given to each behavior, implying that an agent has a set of neighborhoods that define the configuration of its sensors. This is depicted in Figure 3.2. For example, consider a behavior *cohesion* that draws an agent toward its team-mates and a behavior *separation* that moves an agent away from its team-mates. By combining both behaviors while giving a smaller radius of influence to the *separation* behavior, a composed behavior is achieved that keeps agents close to each other while preventing them from getting close enough to collide.

Agents possess a simple memory that allow them to store and recall the relative position of a given point or set of points in space that agents have previously visited. This allows them to return to locations of interest and to keep basic information about those locations.

Agents are also capable of reading the inner-state of team-mates and of interchanging memories, given that the agent providing the information lies in the corresponding neighborhood of the agent receiving it. This provides a simple form of agent-to-agent communication or signaling that can be used to produce higher levels or organization, since it can provide the spreading of information between a group of close-by agents without the need for more costly forms of communication, such as broadcasting. Notice that communication occurs only between members of the same team.

Finally, agents have the capability of grabbing inert objects lying in the environment. The particularities of grabbing have been abstracted into a relatively simple action that requires from the agent simply to move close enough to the object to be grabbed and activate its grip. As long as its grip is activated, the movement of the agent will be transmitted to the object and vice versa, in such a way that both agent and object will move together. The force exerted on the object by the agent is considered to be proportional to the "desired" velocity of the agent, this is, the velocity that would be produced by its behaviors, were it not connected to the object. Objects move according to the total force exerted on them, considering the effect of their mass and the mass of the attached agents. Thus, agents and objects become a single solid for as long as the agents keep their grips activated over the objects. This simplification ignores the problems of aligning a physical grip over an object and the differences between pushing/pulling and other positioning problems.

The agents augmented this way are otherwise regular particles moving in a 2D continuous space, with the exception of possessing both mass and orientation. The mass of the particles is small enough relative to the maximum acceleration produced by the agent's behavior that an agent is capable of changing direction over a small area, although not instantaneously.

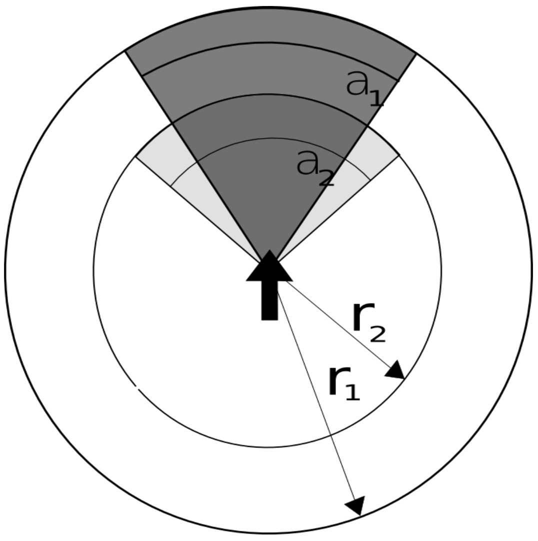


Figure 3.2: Neighborhood of an agent. The agent is represented by an arrow, while the gray area is represents the perceptual area of the agent. Two different neighborhoods are represented, as a single agent may possesses different neighborhoods for different sensors. The first neighborhood is defined by angle a_1 and radius r_1, while the second neighborhood is defined by a_2 and r_2.

3.1.2 Flexibility and Extensibility

Using this architecture, a wide set of different behaviors can be achieved. There are seven basic behaviors, or velocities, that act as building blocks for more complex compound behaviors. These behaviors, summarized in Table 3.1, represent a small catalog of the reactive behavior that agents may require, composed as follows: A *cohesion velocity* $\vec{v}_c$ tends to move the agent toward the center of the flock. Its direction is directly toward the center $\vec{p}_n$ of the agents in its neighborhood, while its magnitude is a fraction of the maximum velocity that increases quadratically with the distance from that center. An *alignment velocity* $\vec{v}_{al}$ tends to move the agent in the same direction that its neighbors are moving. This velocity has the same direction as the average velocity $\vec{v}_n$ of its neighbors, and its magnitude is a fraction of the maximum possible velocity that increases quadratically with the distance from the center $\vec{p}_n$ of the neighbors. An *avoidance velocity* $\vec{v}_{av}$ tends to move the agent away from obstacles. This velocity is directly away from the nearest obstacle location $\vec{p}_o$, while its magnitude is a fraction of the maximum velocity which decreases quadratically with the distance to the obstacle. The avoidance velocity allows agents to block one another while guarding (no other obstacles besides agents were introduced in the experiments reported). A *separation velocity* $\vec{v}_d$ tends to move the agent away from neighbors. This velocity is away from the center of the flock. Its magnitude is a fraction of the maximum velocity which decreases quadratically with the distance to the center of the flock. A *seeking velocity* $\vec{v}_s$ influences an agent to move toward an observed target, specifically a unit of mineral. A *clearance velocity* $\vec{v}_{cl}$ influences an agent to steer toward the side when there is an agent in front of it, in this way clearing its line of sight. This behavior tends to align a group of agents side by side, promoting a broad visual field for the group of agents as a whole. Finally, a *homing velocity* $\vec{v}_h$ is similar to $\vec{v}_s$ in the sense that it drives the agent to a point in the space such as the home of the agent or a remembered deposit. The homing velocity is a vector of magnitude v_{max} pointing directly toward the given fixed point.

The individual velocity influences above are combined as a weighted sum at each time step to update each agent's resulting velocity $\vec{v}$. However, a simple linear weighted addition of the individual velocities has certain negative effects [81], such as directing an agent in intermediate directions that none of the individual influences intended. Therefore the summation process is also prioritized, with the terms being added in a fixed order, and when the sum exceeds a certain threshold v_{max} the term is dropped. Specifically, the velocity of each agent is updated as:

$$\vec{v}^{i}_{new} = \begin{cases} \vec{0} & \text{if } i = 0 \\ \vec{v}^{i-1}_{new} + w_i\vec{v}_i & \text{if } ||\vec{v}^{i-1}_{new} + w_i\vec{v}_i|| < v_{max} \\ \vec{v}^{i-1}_{new} & \text{otherwise} \end{cases}$$

$$\vec{v}_{new} = \vec{v}^{n}_{new}$$

The velocities $\vec{v}_i$ with $i = 1 \ldots n$ are the velocity vectors resulting from the different individual velocity components listed earlier, where the index i corresponds to the priority of each velocity component. A different prioritization and set of weight values is used to produce each specific behavior of the agent. Once the new velocity of the agent has been computed, its new position is updated by adding it to the current position:

$$\vec{p}_{new} \leftarrow \vec{p} + \vec{v}_{new}$$

By prioritizing and weighting each of the seven velocity components differently in computing $\vec{v}_{new}$, or adding new velocity components as needed, several higher-level composed behaviors can be achieved for the specific purposes of the problem being solved. Figure 3.3 illustrates five examples of composed behaviors that are used repeatedly throughout the remaining chapters of this dissertation. Notice that both different equations for the basic behaviors and different non-linear forms of combining them are certainly possible; however, for the purposes of using particle systems as distributed problem-solving systems, only the resulting composed behavior is important, which provides certain flexibility as to how achieve them. Additionally, this architecture makes it easy to plug in other basic or composed behaviors as they are required for specific applications, which makes this part of the system readily applicable to a variety of problems.

3.2 The Competitive Foraging Experiment

As a first test of the benefits of collective rather than individual problem-solving, a resource collection task was used where coordination problems arise naturally. The *competitive foraging experiment* consists of groups of homogeneous agents working in teams to collect and transport back to their teams' bases certain items spread in space. Agents are free to interact with other agents, either in the same team or in competing teams, by forming relatively close packs, blocking the movement of competitors, stealing items collected by other teams, etc. However agents are not allowed explicit, direct communication, except by a simple form of signaling that allows an agent to know the state, a high level abstraction of an agent's short term goals and intentions, of neighboring agents. It is hypothesized that collective movements suffice as a coordination mechanism capable of producing the levels of cooperation necessary in this experiment without resorting to more sophisticated means of communication.

A more detailed description of the experiment follows: One or more teams of homogeneous agents are initially deployed at a random location inside a 2D world with periodic boundary conditions (see Figure 3.4). In this world, units of some

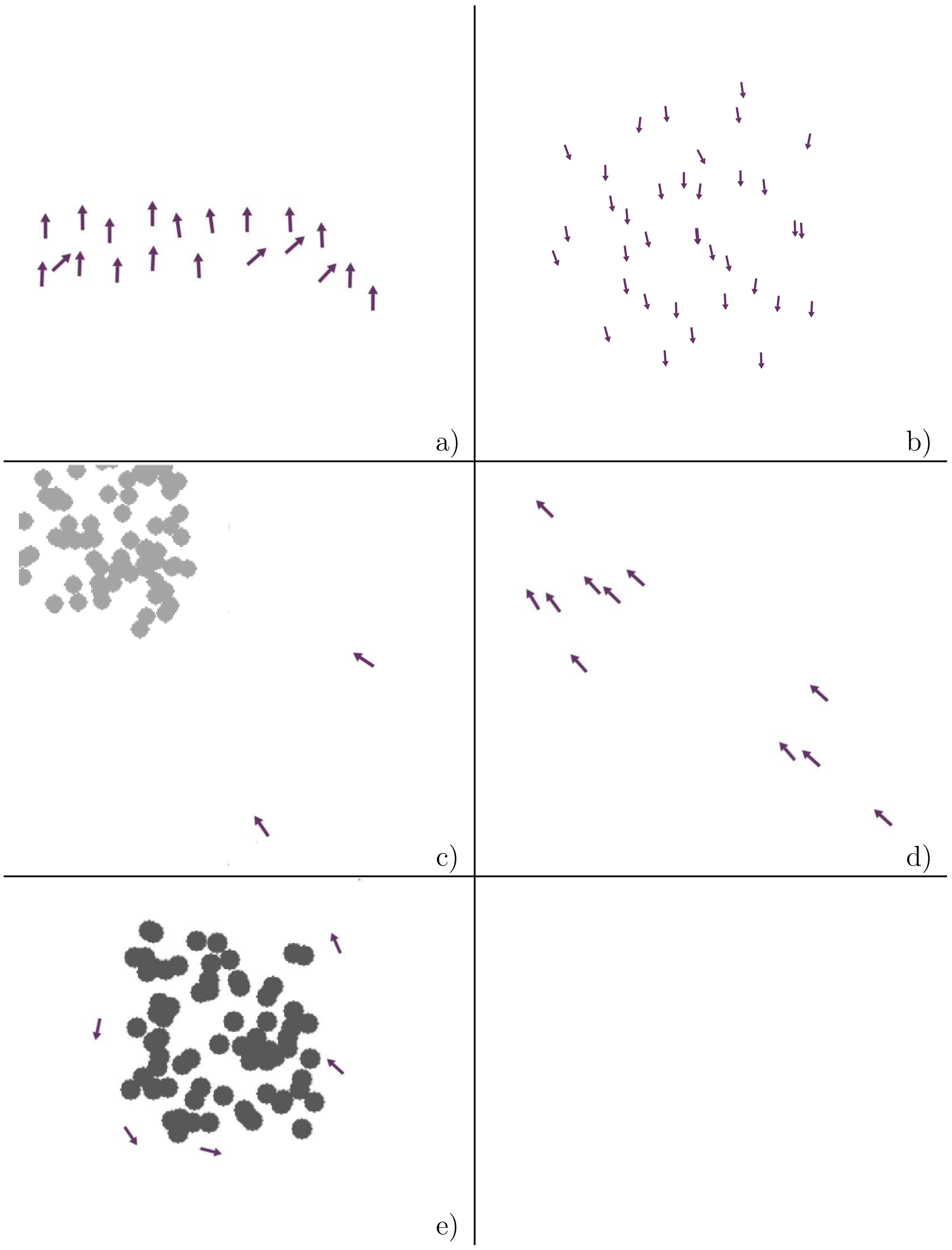


Figure 3.3: Examples of composed behaviors achieved by the combination of basic velocities. a) Spreading: agents roughly move in a line that maximizes the arc swept as they move. b) Flocking: a compact aggregation of agents. c) Seeking: agents move directly towards a given location. d) Caravaning: a column (roughly) of agents that move orderly as a caravan. e) Guarding: the agents patrol an area or location by circling around it.

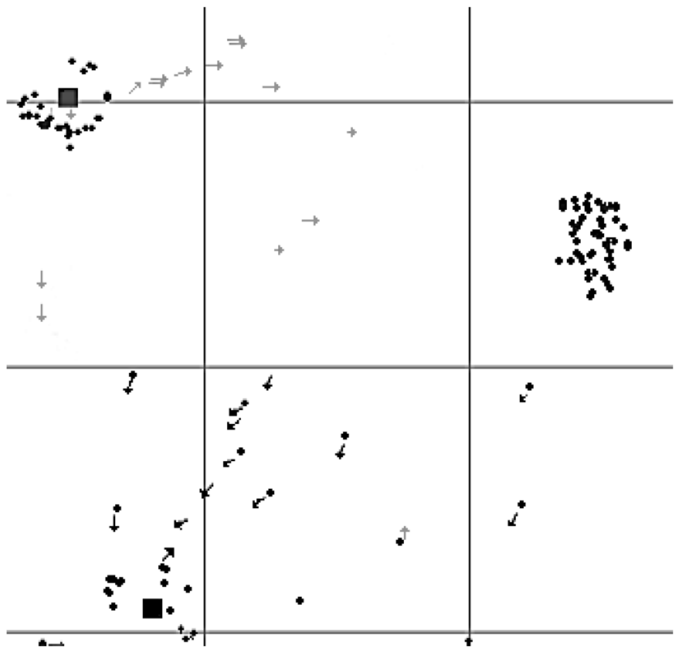


Figure 3.4: A small section of a 3000 x 3000 continuous world is shown with a single “mineral deposit” on the upper right and two different teams (dark and light arrows) exploiting it. Teams’ homes are denoted by small solid squares, agents as arrows and mineral units as spots. The dark team agents are returning home while carrying minerals (spots adjacent to arrows). Most agents of the light team are returning to the deposit after unloading minerals in their home, but some of them are simply exploring.

resource that we call minerals (but which could also represent food, fuel, money, or any other valuable material) are located randomly. A few areas are heavily dense in mineral, thus serving as “deposits”, while the rest of the minerals are scattered throughout the world. The task of the agents is to collect the minerals and deposit them in their designated “home” location, which is common to all members of a single team. This is similar to past tasks used in computer simulations of object collection by social insects [20, 58, 80], but differs in that our task has a designated target collection location, and in that our agents undergo flocking, unlike the independently-moving agents often used in past studies.

The success of a team is measured in terms of the amount of minerals accumulated at its home over time, which advances in small discrete steps. Other teams may be present in the world, and members of one team can potentially hinder rival teams by different means, such as blocking them (agents are not allowed to collide with others), or even stealing from their home. Thus, the task is cooperative within a team, and competitive between teams. Notice that for this reason the amount of mineral accumulated can also decrease over time. Therefore we adopted as absolute measure of performance the amount of mineral collected (and retained) after a long period of time.

The competitive foraging experiment presents a suitable frame for the study of cooperation between mobile agents since it provides opportunities for:

- The use of collective defensive strategies against antagonistic teams, such as guarding a teams' base.
- The use of collective offensive strategies against antagonistic teams, such as coordinated attacks against guarded enemy bases.
- Collective search and retrieval strategies.
- Effective use of division of labor, by requiring a team to perform several non-related task, sometimes all at once, such as searching, retrieving, defending the team's home and attacking others.

In this sense the task is similar to a generalized version of the robotic capture-the-flag game [6], with minerals representing a multitude of flags for a longer, iterated game, as opposed to the one-shot version of capturing a single flag once and for all.

3.3 A Solution Within the Guided Self-Organization Framework

Figure 3.5 shows the FSM employed in this first set of experiments and the corresponding movement behaviors associated with each state. Agents are initially in an idle state. Once they receive a “start signal”, they begin *searching* for resources. When they find some, they choose between *picking up* the minerals or *guarding* the deposit from other teams, depending on whether a group of guarders is already formed. An agent may know this by looking at the state of neighbor agents when it arrives to the deposit. When an agent detects five or more agents guarding in its neighborhood, it decides the deposit is already guarded. Agents recognize a deposit when they detect a certain amount of minerals in their local neighborhood. This implies that homes of other teams are interpreted as deposits, given that they have accumulated enough minerals. If the agent succeeds in picking up a unit of mineral, then it starts *carrying* the minerals home. After arriving home with a unit of mineral and depositing it, if the home is unprotected agents go to *guarding* to protect it, otherwise they return to the last deposit they were exploiting (*returning to deposit*). As before, agents decide that the home is protected when they detect at least five guarders. If the flock arrives to its (unprotected) home, the first five agents will become guarders and the rest will return to the deposit. Since agents can detect guarders only in their neighborhood, if a home or deposit were big enough it could have more than five guarders distributed across the deposit as long as they do not detect all the others at once. When an agent is guarding, it will remain so unless all of the mineral in its neighborhood is taken away (the deposit is exhausted), in

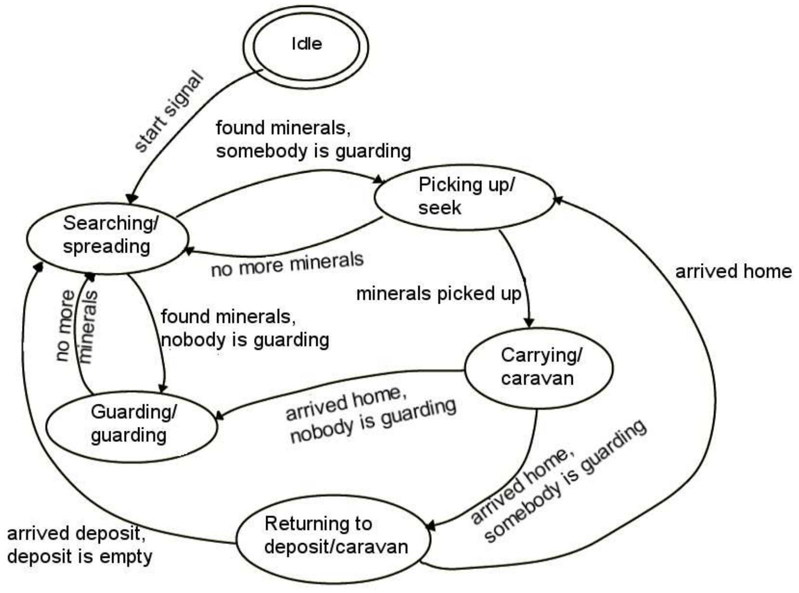

Idle
start signal
found minerals, somebody is guarding
Picking up/ seek
Searching/ spreading
no more minerals
arrived home
minerals picked up
no more minerals
found minerals, nobody is guarding
Carrying/ caravan
Guarding/ guarding
arrived home, nobody is guarding
arrived home, somebody is guarding
arrived deposit, deposit is empty
Returning to deposit/caravan

Figure 3.5: FSM of an agent showing its states and the movement behaviors associated with each state. States are represented by circles labeled by <State/associated controller>, while arrows represent transitions labeled by the triggering event. The initial state is marked by a double circle.

which case the agent returns to *searching*. When agents are *returning to deposit*, it could happen that the deposit is already exhausted, in which case the agent goes into *searching* for another deposit, otherwise it tries *picking up* more minerals.

Agents have a very simple memory that allows them to recall points of interest in the world (Figure 3.6). Each agent has a stack that represents the location of the deposits that the agent has found (with the most recent on top) and the location of the home of the agent; the latter does not change. The precise current goal of the agent is thus represented as a combination of the current state and the contents of this memory. For example, the state could determine that the agent is going back to a deposit to exploit it, while the memory determines which deposit the agent is to exploit. Thus, a simple sequence of implicit goals can be formed, represented as deposits queued for later exploitation.

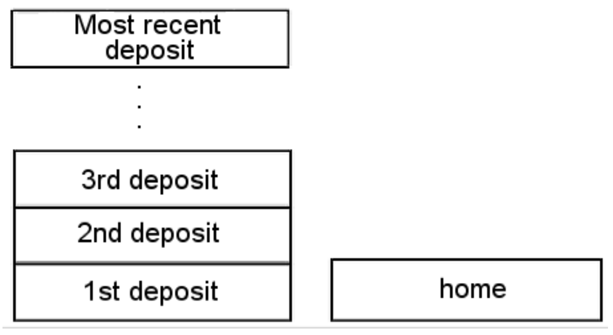


Figure 3.6: Memory of an agent. The stack on the left represents the location of the deposits visited by the agent. The single record at the right represents the location of the agent's home.

3.4 Experimental Set-Up

A basic computational experiment consists of letting different simulated teams compete for the minerals in the same world for a limited time. To compare the effectiveness of collective movement behaviors (flocking) versus independently moving agents, a set of independent agents have been implemented. These agents follow the same model explained above, with the important exception that they do not take into account other agents in their dynamics. In other words, they mainly ignore other agents from the same team. The controllers for these agents replace dependent steering behaviors (cohesion, alignment, separation, clearance) with random wandering. Collision is not replaced for obvious reasons. Otherwise the FSM of the non-flocking (independent) agents is the same as for the flocking agents (Figure 3.5).

The results reported below are the average over 20 runs for each experiment, using 50 agents per team (unless indicated otherwise) in a continuous world of size 3,000 x 3,000 and one team for every type of agent (described below). Agents as well as minerals are both one unit in size compared to the world.

During each run, 12 deposits were randomly located and independently created, each deposit consisting of 80 units of minerals for a total of 960 units of mineral in the world. These parameters were chosen so that it is valuable for a team to go back to a deposit repeatedly. Also, the number of deposits and size of the world make it easy for agents to find the deposits (and other agents' home), but still force agents to compete for the resource (minerals). The maximum velocity of an agent was set to 12 and agents were able to detect minerals up to 200 units away (arbitrary units). Teams' homes were also independently located at random in each run and agents were initially randomly positioned within a radius of 100 units of the team's home. In each run, the simulation lasted 40,000 time-steps (iterations), which was adequate for the system to stabilize, that is, the time needed for the average amount of mineral in teams' home to converge to some value. Notice that the mineral in deposits is not replaced, meaning that eventually all mineral present in the world will be distributed among agents' homes.

Multiple teams of agents were often used in an experimental run, with each team being of a different type. Which teams were involved could vary form experiment to experiment. The six different types are:

- Full-guarding flock. Flocking agents which implement the full FSM of Figure 3.5. These agents guard home and any deposits that they find. The search pattern is a spreading flock.

- Home-guarding flock. Through elimination of transition from searching to guarding, these agents will not guard a deposit, but will still guard their own home. This allows more agents to be involved searching for and exploiting deposits, but also allows other teams to exploit a deposit previously found by another team.

- Non-guarding flock. These agents are the same as above except they do not have a guarding state. All agents are actively involved in either searching for deposits or exploiting deposits.

- Full-guarding, home-guarding and non-guarding independent teams. These three types of agents correspond respectively to the three types of flocking agents above, but do not undertake collective movements; they move independently. They search through random wandering and see other agents (from the same or competing team) only as obstacles to be avoided.

3.4.1 Behaviors

As mentioned before, each state of the agents implies a desired behavior in terms of low-level dynamics. Specifically for these experiments, the behaviors of spreading, seeking, caravaning and guarding are required. These medium-level behaviors were achieved through the combination of basic behaviors shown in Table 3.2.

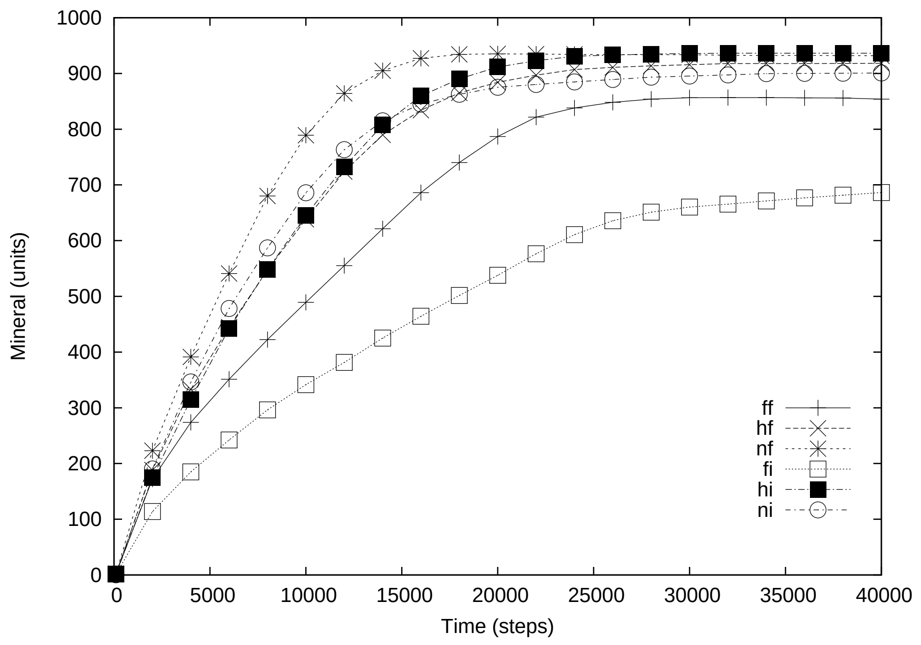


Figure 3.7: Mineral collected over time when each team is present alone. ff: Full-guarding flocking, hf: home-guarding flocking, nf: non-guarding flocking, fi:full-guarding independent, hi:home-guarding independent, ni: non-guarding independent.

3.5 Results

In the following we measure the effectiveness of each team in isolation, when it is simultaneously present with all other teams, and in pairing competitions, to establish the value of collective versus independent movements and of searching versus guarding actions.

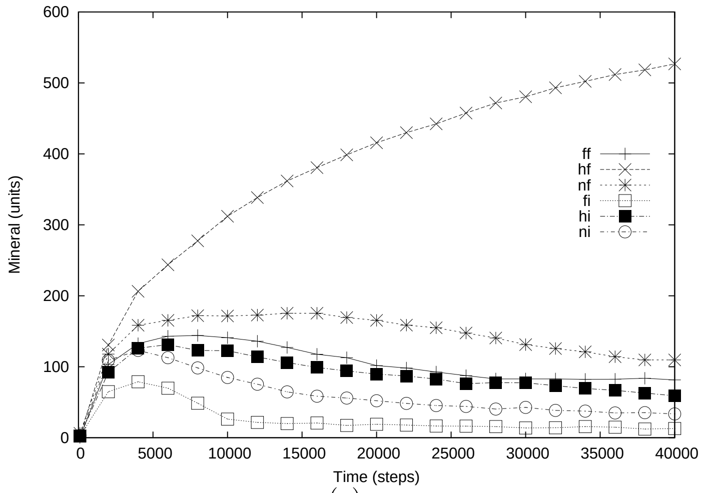
3.5.1 Each Team in Isolation

In this first experiment, a team is present in the world without any other competing teams. The experiment was repeated for each team. Figure 3.7 shows the amount of minerals collected over time, averaged over 20 runs.

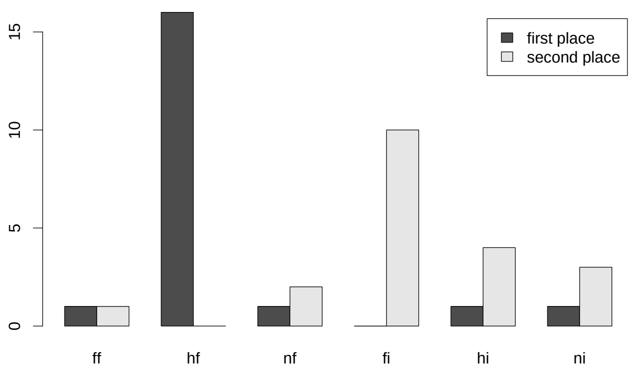
After 20,000 iterations most teams have succeeded in collecting almost all of the minerals available in the world, with the exception of the full-guarding teams that are still slowly collecting resources. This is due to the fact that after the full-guarding teams split off members to guard every deposit found, not enough agents remain to collect the minerals rapidly. The non-guarding flocking team seems to be the fastest in collecting minerals, which is consistent with the fact that all team members are actively searching for and collecting minerals, and that in the absence of other teams, guarding mineral deposits has no value. In this world where agent teams do not need to compete with other teams, there is no clear difference between the other four teams.

3.5.2 All Teams Simultaneously

In this experiment, all six types of agents (one team per type) are present simultaneously in the same simulation. The number of agents per team was decreased to 30 to avoid overcrowding. Figure 3.8(a) shows the amount of minerals in each teams' home over time, averaged over 20 runs. Most striking is that the home-guarding, flocking team's collected resources increase monotonically, with this team clearly outperforming all others. Early in simulations (during the first 5,000 iterations), both this and the non-guarding flocking team collect minerals faster than any other team. After the first few thousand iterations, the explored area of each team is wide enough for teams to find each others' homes. Accordingly, the amount of minerals decreases in subsequent iterations for most teams, especially the non-flocking teams. Notice that during this period the amount of minerals at home



(a)



(b)

Figure 3.8: Results of simulations when all teams compete simultaneously in a single world. (a) Mineral present in teams' home per unit of time. (b) Rank according to amount of minerals collected at time t=40,000. Same notation as in Figure 3.7.

does not decrease as much for the non-guarding flocking team. Presumably this is because this team is fast enough collecting minerals to compensate for any stolen minerals. The amount of minerals in the full-guarding flocking team slowly increases over the time. This is probably due to the fact that this team can prevent its home from being looted, but employs too many members protecting every deposit it finds, which results in not enough members actually exploiting these deposits in an efficient manner. The home-guarding flocking team does not have this problem, thus it can collect about three times as many minerals as any non-flocking team by the end of the simulation period.

Table 3.3 shows the mean amount of collected minerals by each team after 40,000 iterations over 20 runs. The differences are statistically significant at the level of 95% according to a two-way ANOVA, both in sociality (flocking vs independent) and guarding strategy (full-guarding, home-only and none). There is also interaction between the sociality and the guarding strategy, which implies that varying guarding strategies has a bigger impact for flocking agents than for independent agents.

Figure 3.8(b) depicts how often over these 20 runs each team type was in first place (number of runs a team collected more minerals than any other), and in last-place. These data suggest two main hypotheses. First, teams of collectively moving (flocking) agents are more effective at this task than corresponding teams of independently moving agents. This can be seen to be true in Figure 3.8(a) by comparing each pair of teams that use the same guarding protocol (ff versus fi, etc). With collectively moving (flocking) agents, whenever a deposit was discovered by an agent, numerous other agents were immediately nearby and thus pulled in by local inter-agent influences to help collect the discovered minerals (e.g., see Figure 3.9). Second, for both collectively and independently moving agent teams, agents that guarded only their home did better than non-guarding agents, who in turn did better than full-guarding agents. Presumably this is because allocating agents to guard resources, especially multiple deposits, has a large cost: it removes these agents from collecting minerals, and this loss is not adequately compensated for by any protective influences they exert through their blocking actions.

3.5.3 Collective versus Independent Movements

The impact of collective versus independent movements on agent teams can be clarified by varying just that factor between two competing teams. Figure 3.10 shows the mean amount of minerals saved at home over time for pairwise competi-

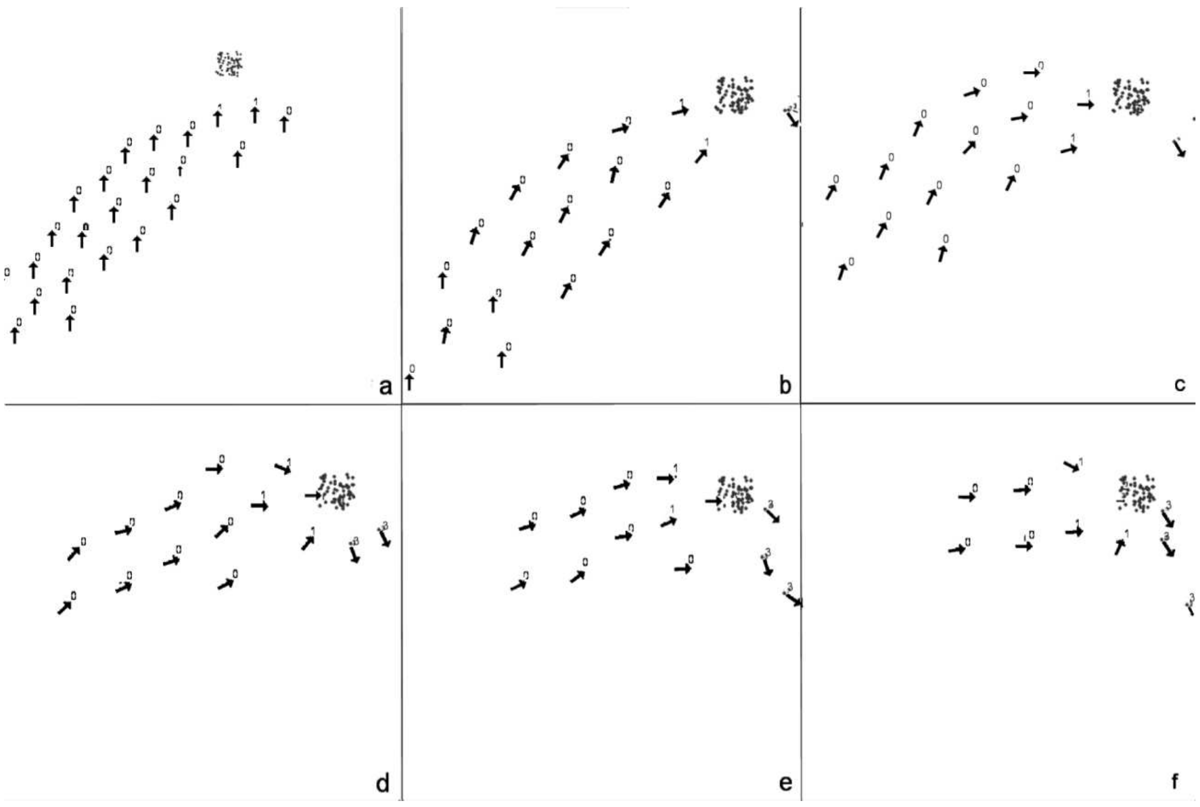


Figure 3.9: Agents in a flock being pulled toward a deposit. The number on top of each agent represents its current state (0 for Searching for a deposit, 1 for Picking up). Only agents in state 1 actually detect the deposit. At a, only two agents have located the deposit, while the rest of the flock moves northward. At b and c, agents that are near the deposit but that do not yet see it turn toward those agents that have seen the deposit and are already going toward it. From d to f, the whole flock gradually turns toward the deposit and collects minerals. Such behavior indicates an advantage of collective movements in recruiting other agents to carrying a resource when it is discovered by just a few.

tions of collectively moving versus independently moving teams of agents. Table 3.4 shows the mean and standard deviation for the amount of minerals collected by the last iteration (40,000). These results are significant at the level of 95% according to a Wilcoxon rank sum test. It is clear that the flocking teams are always faster in accumulating minerals. In general, even by around iteration 7500, flocking teams have collected many more minerals than their non-flocking counterparts. Even more striking, the independently moving teams are not sufficiently effective in protect-

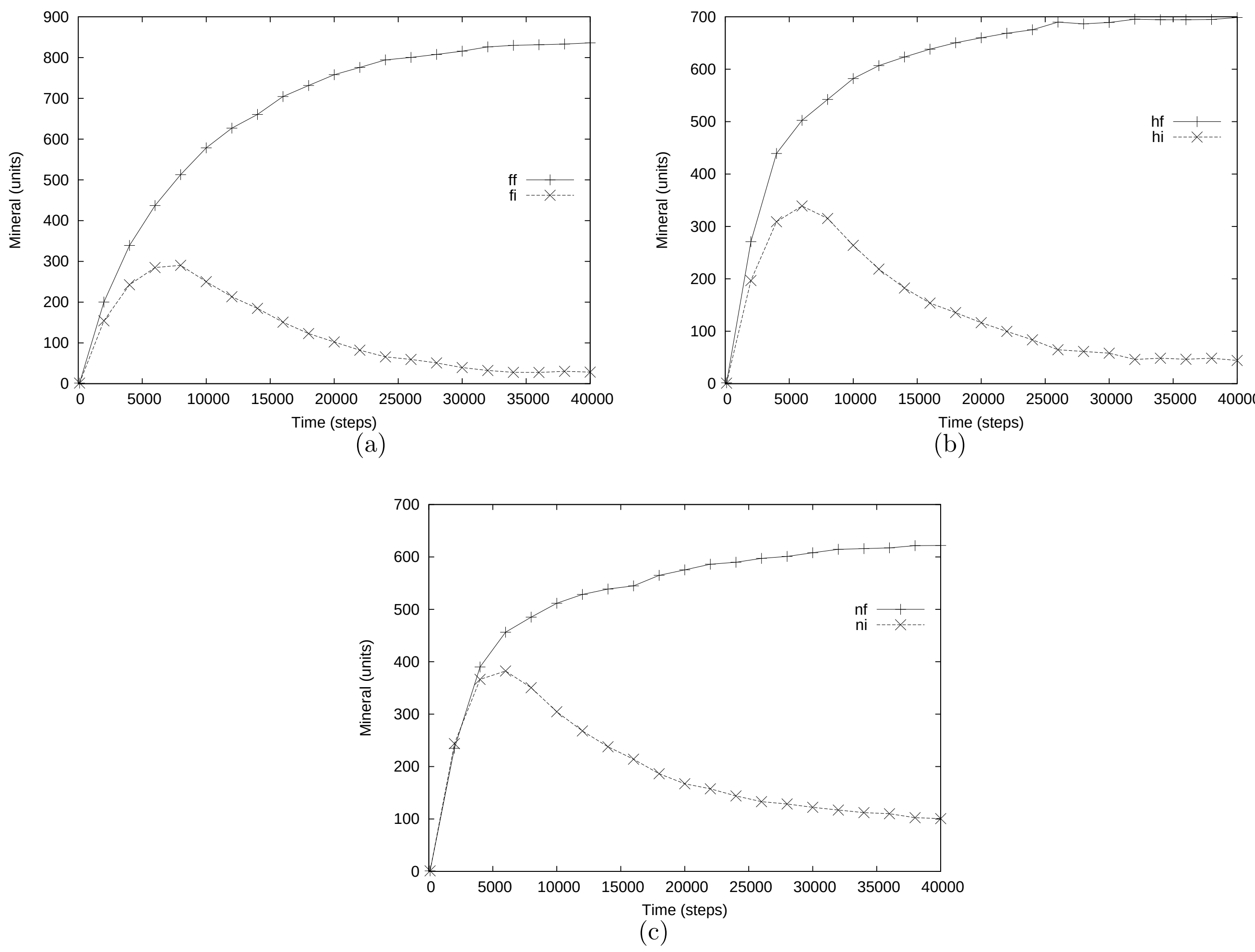


Figure 3.10: Mean mineral collected at home over time by full-guarding teams. (a) ff: full-guarding flocking versus fi: full-guarding independent agents. (b) hf: home-guarding flocking versus hi: home-guarding independent agents. (c)nf: non-guarding flocking versus ni: non-guarding independent agents.

ing its home from being looted by the flocking team, and their collected minerals considerably decrease during the following iterations.

3.5.4 Guarding versus Non-Guarding

Finally, experiments to compare the advantages of guarding versus non-guarding were performed. These experiments involved pairs of guarding versus non-guarding teams which follow similar collective strategies (either flocking or independent movement).

Figures 3.11 shows the mean amount of mineral saved at the team's home over time, while Table 3.5 shows the mean amount and standard deviation of minerals saved at the last iteration. Results are significant at the level of 95%. Early on the simulation (about iteration 5,000) pairwise teams seem to have similar performance, but after this point guarding teams show a clear advantage, as their amount of minerals saved at home keeps increasing, while the amount of minerals in the home of non-guarding teams decreases, probably as it is taken by the opposite team. Again, home-guarding teams perform better than full-guarding teams.

3.6 Discussion

This chapter introduced an architecture to provide an additional level of control over particle systems. This control, built in each particle without requiring global or centralized control, influences the behavior of the particles and hence of the system through their local interactions, altering or guiding the self-organization process to produce a specific system-wide behavior. As shown by the results, it is feasible to extend self-organizing particle systems to exhibit more general problem-solving behavior. The new extended particle system, which includes behavioral states over the usually exclusively reactive agents of particle systems and a simple memory, results in system or team of agents capable of collectively solving search and collect problems. By using a high-level controller that keeps track of the intermediate goals of an agent according the conditions of the environment and to the

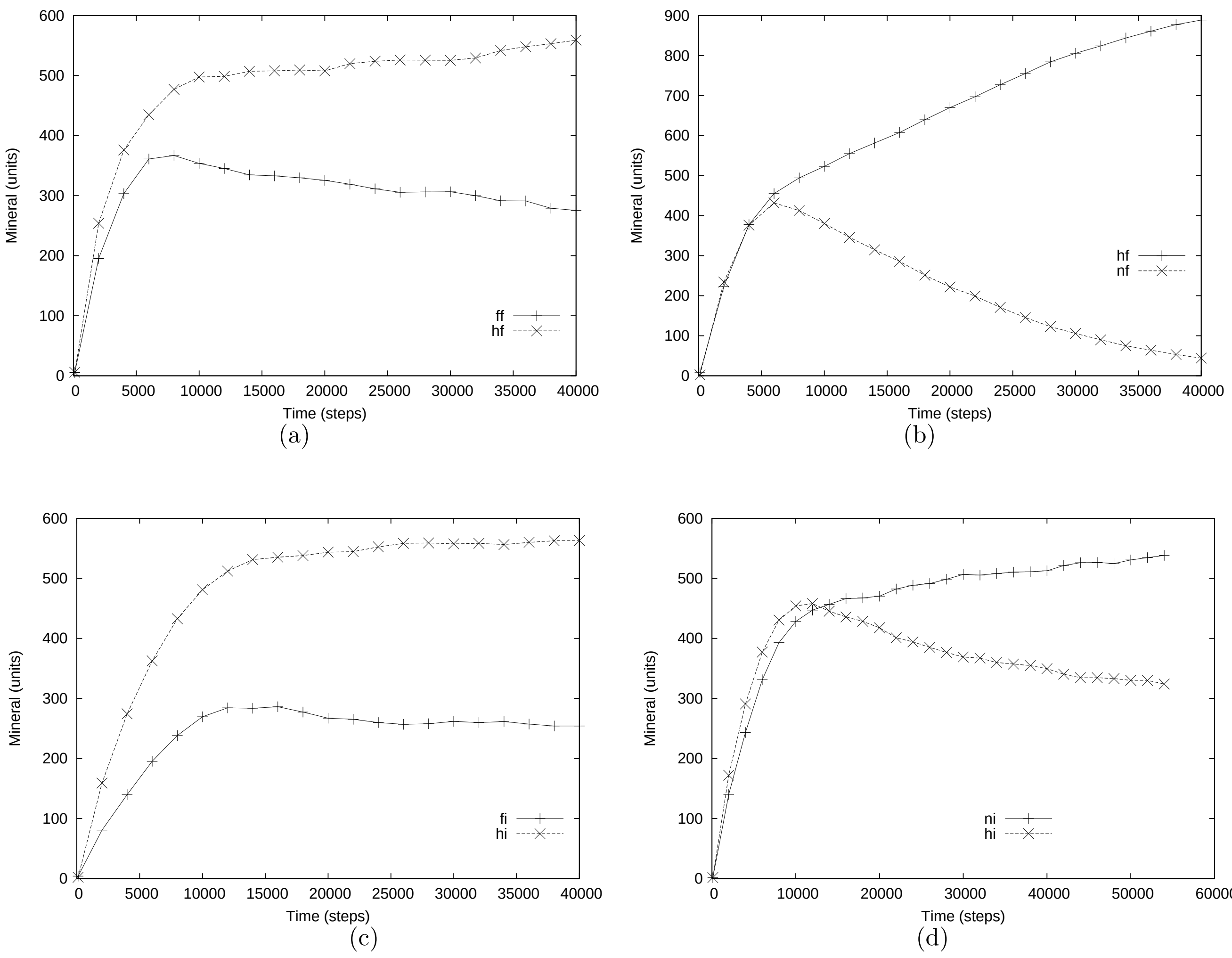


Figure 3.11: Results of simulations for flocking teams in a single world. Mineral present in teams' home per unit of time. (a) ff: full-guarding flocking versus hf:home-guarding flocking. (b) hf: home-guarding flocking versus nf:non-guarding flocking. (c) fi: full-guarding independent versus hi:home-guarding independent. (d) hi: home-guarding independent versus ni:non-guarding independent.

general logic of the problem, problem solving abilities emerge in the team as a whole while retaining the local nature of the information processing.

The simulation results presented in this chapter provide substantial support for the efficacy of this approach. As state changes occurred and spread throughout a collection of agents via local interactions, the group's motion as a whole was influenced and shifted to provide collective problem-solving. Most basically, this was seen in that an agent team could routinely search for, collect, and return discovered resources to a predetermined home location, all the while retaining their joint movement as a “flock” of individuals. Further, it was found in simulations that a team of agents that moved collectively were more effective in solving search and collect problems than very similar agents that moved independently. This was because when one or a few agents on a collectively-moving team discovered a site with plentiful resources, they would automatically pull other team members toward that site, greatly expediting the acquisition of the discovered resource and its transport back to the team’s home base. Thus, a benefit of underlying collective movements of particle systems which, to our knowledge, has not been appreciated in past work, is that they have the potential to automatically recruit additional agents to complete a task when one agent detects the necessity of that task.

Additional computational experiments were undertaken in which multiple teams with somewhat different behaviors simultaneously competed to find and collect the resources that were present. Six different teams were used: three that exhibited flocking (collective) movements, and three others where the individual agents moved independently. In both cases the three teams involved differed in whether agents/particles were allowed to guard (orbit and protect) both discovered resources and their home base of accumulated resources, just their home base, or neither. Regardless of whether the teams competed two at a time or all at once, we found consistently that collectively moving agents were superior to independently moving teams of matched agents in collecting resources. Further, and regardless of whether agents moved as a team or independently, we found that those that were allowed to guard only their home base did best, those that tried to guard both home and discovered resources did worst, and those that guarded nothing were in between. This finding reflects a kind of exploration/exploitation trade off: guarding has a protective value for preserving located and collected resources, but also a cost in that fewer agents are available to continue searching for and collecting new resources. Most importantly, these simulations exhibited group-level decisions not just about which type of movements to make, but also about when it was appropriate to split into groups, with one smaller group remaining to guard resources. They exhibited decentralized cooperation without explicit coordination, such as when a wandering agent would follow another agent that knew the location of uncollected resources, simply because wandering agents tended to follow other agents.

Table 3.1: Movement Equations for the Different Movement Behaviors

Movement Behavior	Equation
Cohesion	$\vec{v}_c = \left(\frac{\|\Delta\vec{p}\|}{r}\right)^2 v_{max}\frac{\Delta\vec{p}}{\|\Delta\vec{p}\|}$
Alignment	$\vec{v}_{al} = \left(\frac{\|\Delta\vec{p}\|}{r}\right)^2 v_{max}\frac{\Delta\vec{v}}{\|\Delta\vec{v}\|}$
Avoidance	$\vec{v}_{av} = -\left(1 - \frac{\|\Delta\vec{p}_o\|}{r}\right)^2 v_{max}\frac{\Delta\vec{p}_o}{\|\Delta\vec{p}_o\|}$
Separation	$\vec{v}_d = -\left(1 - \frac{\|\Delta\vec{p}\|}{r}\right)^2 v_{max}\frac{\Delta\vec{p}}{\|\Delta\vec{p}\|}$
Seeking	$\vec{v}_s = v_{max}\frac{\vec{p}_t-\vec{p}}{\|\vec{p}_t-\vec{p}\|}$
Clearance	$\vec{v}_{cl} = v_{max}\vec{o}$
Homing	$\vec{v}_h = v_{max}\frac{\vec{p}_h-\vec{p}}{\|\vec{p}_h-\vec{p}\|}$

$\vec{p}$:	current position of agent
$\vec{p}_n$:	averaged position of neighbor agents
$\vec{p}_o$:	position of nearest competing agent
$\vec{p}_t$:	position of observed target
$\vec{p}_h$:	position of arbitrary target
$\vec{o}$:	unit vector orthogonal to $\vec{v}$ ($\vec{o} \times \vec{v}$ in $+z$ direction)
$\vec{v}$:	current velocity of agent
$\vec{v}_n$:	average velocity of its neighbor agents
v_{max} :	maximum speed of agent

$$\Delta\vec{p} = \vec{p}_n - \vec{p}, \quad \Delta\vec{v} = \vec{v}_n - \vec{v}, \quad \Delta\vec{p}_o = \vec{p}_o - \vec{p}$$

Table 3.2: Parameters for Movement Behaviors.

Movement Behavior	Velocity Component	Priority	w_i	r	α
Spreading	Avoidance	1	0.66	100	180^o
	Separation	2	0.83	150	180^o
	Clearance	3	0.83	150	115^o
	Alignment	4	0.83	250	115^o
	Cohesion	5	0.83	300	360^o
Seeking	Avoidance	1	0.83	100	180^o
	Seek	2	0.66	250	360^o
	Separation	3	0.25	50	90^o
Caravaning	Avoidance	1	0.83	100	180^o
	Homing	2	0.83	-	-
	Separation	3	0.83	100	180^o
	Clearance	4	0.83	150	60^o
Guarding	Avoidance	1	1	100	180^o
	Separation	2	0.62	50	180^o
	MineralCohesion	3	0.62	150	360^o

Table 3.3: Minerals Collected After 40,000 Iterations, All Teams Competing*

	Guarding		
Sociality	Full	Home	None
Flocking	81 (99)	526 (292)	109 (175)
Independent	12 (23)	59 (108)	33 (58)

*Entries are: mean (standard deviation).

Table 3.4: Mineral Collected After 40,000 Iterations, Pairwise Simulations

Simulation	Means	Standard deviations
ff vs. fi	836 vs. 28	188 vs. 68
hf vs. hi	699 vs. 44	220 vs. 85
nf vs. ni	621 vs. 100	300 vs. 176

Table 3.5: Amount of saved mineral at home in pairwise simulations

Simulation	Means	Standard deviations
ff vs. hf	275 vs. 559	212 vs. 252
hf vs. nf	888 vs. 44	108 vs. 49
fi vs. hi	254 vs. 563	226 vs. 256
hi vs. ni	524 vs. 346	280 vs. 271

Chapter 4

Information Sharing and Spreading

The extended particle system based on guided self-organization has shown in the previous chapter to be able to create self-organization useful in problem solving through its application to a relatively simple problem of search and collection. In this chapter, the properties of the system are further studied by employing in two versions of a problem also well known multi-agent systems, collective transport. Interest in collective transport comes from the fact that it requires cooperation, as opposed to the problem presented in Chapter 3 which in principle could be solved by independent agents, albeit less effectively, which makes it a prime example for the study of cooperative systems. A second reason for interest in this problem is that it usually arises in actual robotic problems, given the convenience of using multiple simple robots assuming these could cooperate as opposed to using a single, sophisticated and more expensive robot. Self-organizing biological systems, in particular ants, are probably the best examples available for the solution of this problem, both in the natural world and artificial systems. However, ants achieve this level of efficiency despite their relatively limited cognitive capabilities. This once again suggests that collective transport present an exceptional opportunity for the study of distributed problem solving through a mainly, but not exclusively, reactive system of self-organizing particles. In this case, the problem also includes the navigation around obstacles. As opposed to the problem presented in the previous chapter, which in principle could be solved, albeit inefficiently, without any cooperation nor coordination, collective transport strictly requires the work of two or more agents, and this work has to be coordinated to be effective, as no single agent has complete control of the object being carried and therefore ignoring other agents' actions results in agents that are unlikely to successfully steer an object without it bumping into obstacles.

The first version of the collective problem that follows presents a simple case of carrying a single (heavy) object by a multitude of agents and require local coordination in order to navigate the object through partially perceived obstacles. The second version consists of a more sophisticated extension that requires the agents to distribute a group of objects among distant locations obeying certain demands not originally known to the agents. This requires the agents to collectively develop global knowledge of the world based on their local interactions and their local perceptions.

The results of the experiments presented in this chapter suggest that collective movements and their self-organizing properties are capable of allowing multi-agent systems to share information, effectively creating a cooperative sensing system that can perform the collective transport task even when using only reactive behaviors. They also show, however, that local communication, once incorporated to the extended particle system, can greatly improve the spread of information over large areas, allowing the system to develop a global strategy to solve a problem through the local interactions of the agents.

4.1 The Collective Transport Experiment

The previous chapter showed some of the advantages of coordinated motions in tasks where cooperation could be useful. Another more clear area of application for collective movements lies in tasks that absolutely require two or more agents to cooperate as a team. In particular, as opposed to the coordination among a large team in terms of division of labor and formation of sub-teams, this chapter studies local coordination when all members of the team need to perform the same task. Again, it is hypothesized that aggregate motions are an effective tool to achieve the required coordination in this kind of problem.

To test this hypothesis, a well studied task in systems of multiple mobile agents has been selected: collective transport [31, 53, 58, 62, 96, 109]. In this task, a set of agents is required to move a single object between two locations while avoiding obstacles. Since our focus is strictly in the cooperation aspect of the task while transporting the object, agents have been provided with knowledge of the desired destination to avoid unnecessary wandering and the complexities of “sensing” the destination. The object to be transported will only move if the total force exerted on it surpass a certain threshold, which could be thought of as a friction force produced by the surface of the world. This threshold was set in a way such that only the combined force of two agents pushing at top strength in the same direction could move the object; therefore the collaboration of at least two agents is required. Notice, however, that while approaching obstacles or steering the agents will not try to move at top speed, which makes a team of only two agents likely to be

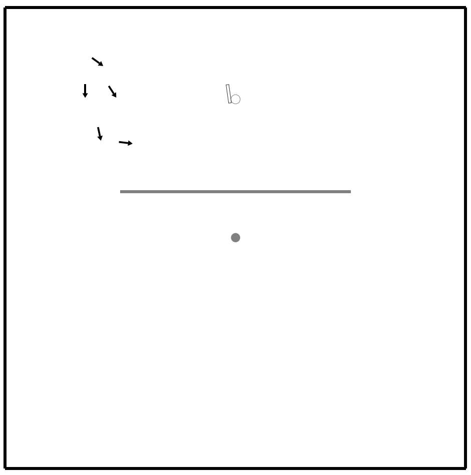


Figure 4.1: Layout of a simple world. Agents are presented as arrows, destinations and sources as filled and hollow circles, respectively. A unit of product (hollow bar), is present near the source. Four walls surround the world to keep the agents constrained inside.

ineffective. Figure 4.1 shows a simple example of this scenario. The objects to be moved are represented as elongated, rod-shaped pieces that follow all the principles of dynamics.

4.2 Application of Guided Self-Organization

4.2.1 High-Level Controller

Figure 4.2 shows the FSM that guides the agents during this task. Basically, agents search for objects, using a flocking pattern that keeps the team in a relatively compact state. When one of the agents locates the object, it moves toward it and tries to grab it. Agents that have not detected the object may be dragged toward the object by the influence of the others, as seen in Chapter 3. A state 'moving away' helps the coordination of the team by making agents avoid objects that are already being carried by a large number of agents (as perceived in the local neighborhood of each agent). This behavior stimulates a team to automatically split into smaller sub-teams as needed to transport a varying number of objects. Additionally, when an agent drops the object, it emits a signal perceivable by its neighbors during a short period of time. This signal indicates to other agents to also drop the object. This produces a coordinated dropping when the agent in front of the team reaches the destination, as opposed to progressive droppings that would result in the agent farther away from the destination continuing to carry and push even when most of the object is already in place. When the team is stagnating, either because they have hit an obstacle or for any other reason, one of the agents will drop the object, but is still attracted by the object itself and by the presence of its teammates, it will likely re-grab it at a different point. The signal that it emits at the moment of dropping will cause its teammates to perform a similar action. The end result is that the whole group of agents will reallocate around the object, which usually is enough to recover from the stagnation and put the object in movement again. This strategy is inspired by the realignment and relocation strategies observed in ants [45, 94, 95].

4.2.2 Perception

Objects and agents in previous chapters consisted exclusively of particles. However for the collective transport problem objects could in principle be in the

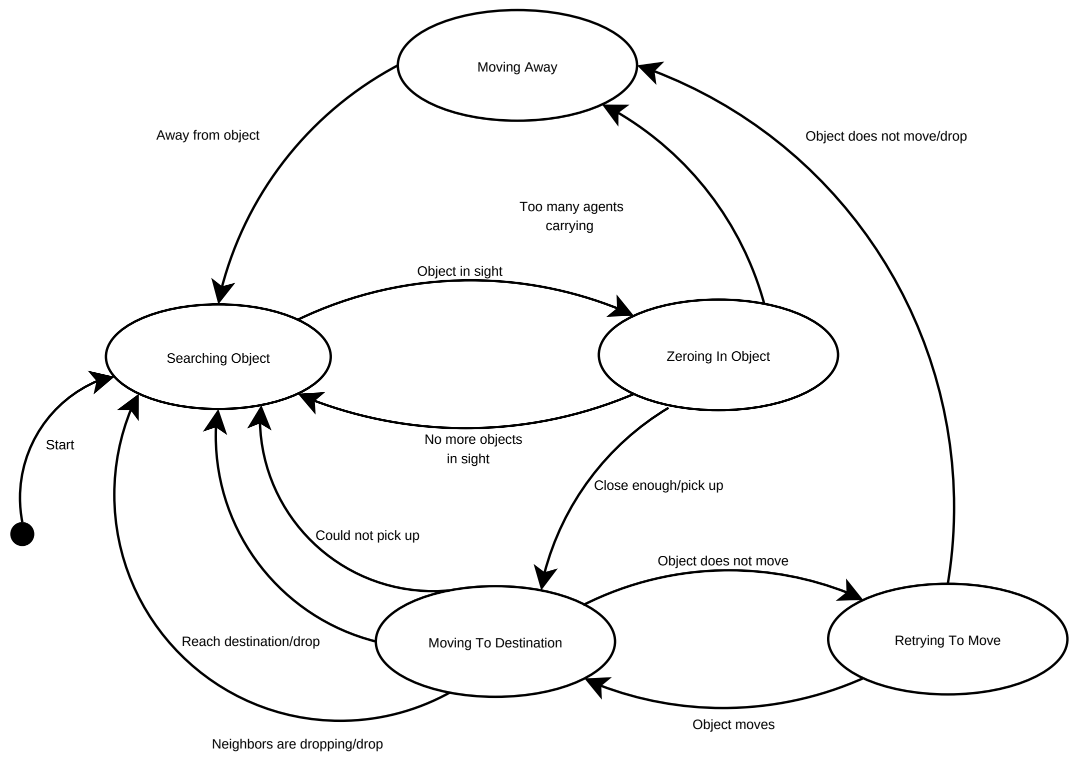


Figure 4.2: FSM of agents. States are shown as labeled circles while transitions are depicted as arrows. Each transition is labeled as *Event/action* where *event* triggers the transition while *action* is an optional action to be performed by the agent only once during the transition. The initial state is marked as a filled circle.

shape of any arbitrary 2D closed polygon. For simplicity only rectangles are used; nevertheless agents are required to be provided with a more sophisticated, although still very simple, "vision" system. In this system, agents perceive the position of the closest point of the object that lies inside the agent's neighborhood. In case more than one object or obstacle is present, agents will only perceive the closest one. This incidentally provides a simple although inaccurate form of occlusion. Figure 4.3 shows an example of the perception system.

Thus, the world as perceived by agents still consists entirely of particles. It is worth noting that agents do not memorize previously seen obstacles, which makes the avoidance problem more complicated since agents do not perceive the shape or

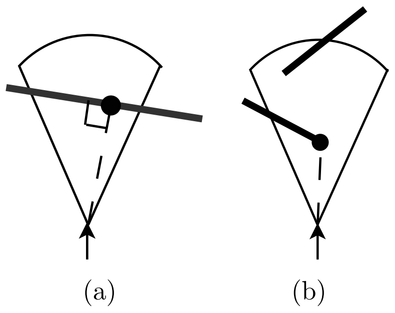


Figure 4.3: Two examples of an agent's neighborhood. Agents detect objects in front of them as a particle whose location is the closest point of the object. (a) shows the perception of the agent when a single object is in its neighborhood. (b) depicts a case of occlusion as two objects are in the neighborhood, but only the closest one is detected.

size of an obstacle except for a point they see of it at any instant.

4.2.3 Alignment and Allocation of Agents While Moving an Object

The collective transport problem as described above presents several opportunities to model collective behaviors, including movements. While transporting the product, agents not only need to agree on the destination, they also have to agree on the path to it. Since they are located at different positions around the object, they might perceive obstacles differently, and therefore pull in different directions. Although it may seem trivial to solve this problem by forcing alignment (same direction) in all agents in the neighborhood, there exist actual advantages in not doing so. For example, Figure 4.4 shows a situation that simultaneously exemplifies the importance of the allocation of the agents along the object and the contribution of pulling in different directions. The first row shows five agents distributed about equally along the object while pulling it around an obstacle. After (**b**), the agents on the front of the object have cleared the obstacle and start pulling towards the target destination (hollow circle); however the agents in the rear keep pushing forward (parallel to the obstacle and away from the destination) as they have not completely cleared the obstacle, causing the object to rotate. This results in the object moving in a diagonal line to the right and downwards (**c**), while slowly turning the corner. The second row shows a case when all agents have positioned themselves close to the front of the object. Since they are so close they will perceive the obstacle in the same way and move simultaneously in the same direction. After all of them have cleared the obstacle (**f**), they will start moving directly towards the target, even though part of the object is still behind the wall. When all agents change direction simultaneously towards the target, the object turns in excess (**g**) and hits the wall (**h**). Thus, by properly distributing themselves along the object, covering both the front and back end of it, agents achieve a simple form of cooperation. This even distribution of the agents along the object was achieved by the combination of the *cohesion* and *separation* behaviors.

4.2.4 Adaptive Weights

Another issue brought up by the dynamically changing environment of an agent that occurs due to the actions of other agents is how these changes affect the way the different behaviors influencing the agent are combined. As mentioned before, the resultant velocity of an agent is computed as a non-linear prioritized weighted summation of the individual velocities $w_i\vec{v}_i$, where $\vec{v}_i$ is the velocity of an independent behavior and w_i is a weight between 0 and 1. Although this strategy

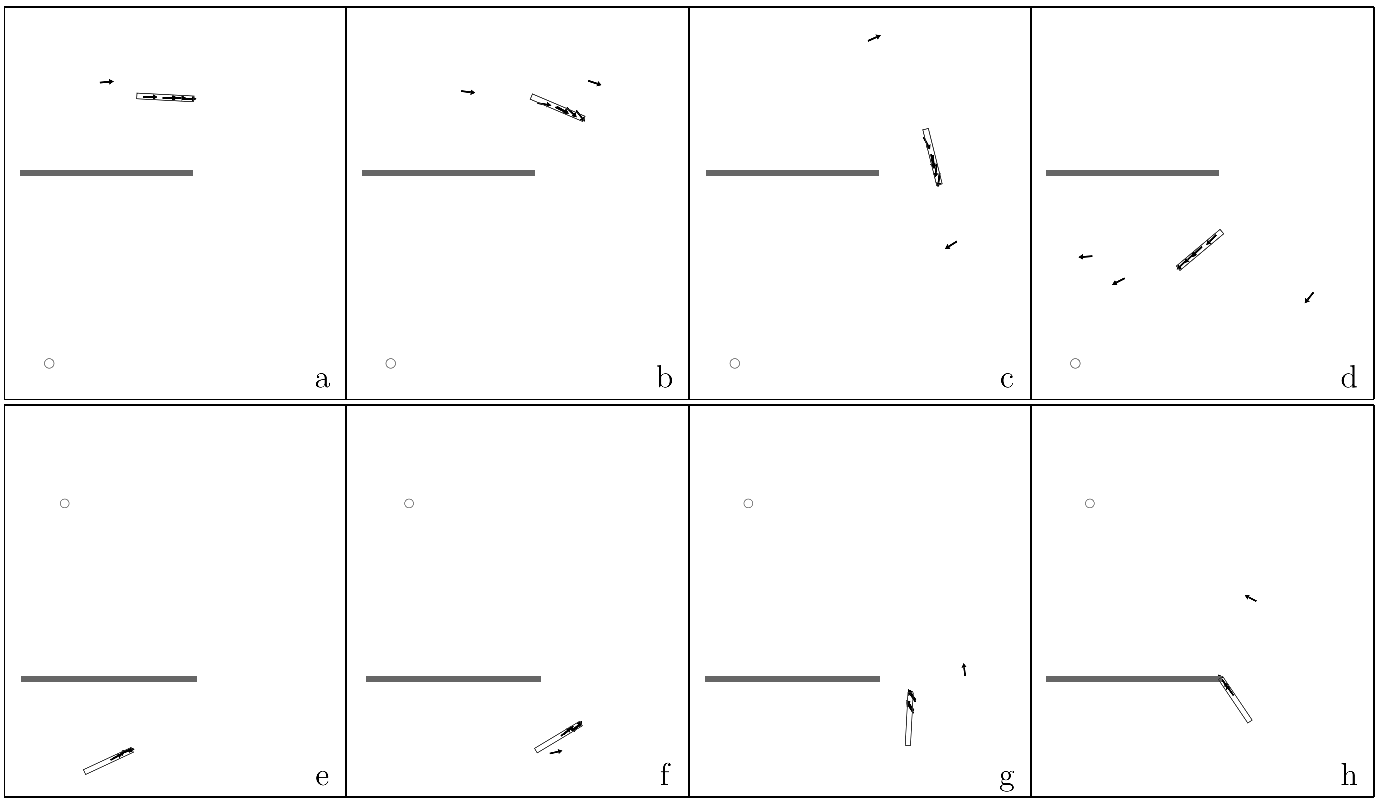


Figure 4.4: Agents moving an object around an obstacle. The object is represented as a hollow bar, while the target destination is a small circle. The agents pushing the bar are represented as small arrows superimposed on the bar. The first row (a-d) and the second row (e-h) show two different time sequences (see text).

works well for the search-and-retrieve task, it presents several problems in the more complex setting described in this section. Take, for example, the *seeking* behavior, which moves an agent toward a unit of product and is computed as:

$$\vec{v}_{seek} = v_{max} \cdot \frac{\vec{p_t} - \vec{p}}{dist} \cdot \sqrt{\frac{dist + c \cdot r}{r}}$$

$$dist = ||\vec{p_t} - \vec{p}||$$

where v_{max}: max speed of the agent, $\vec{p_t}$: position of the target, $\vec{p}$: position of the agent, r: radius of the neighborhood of the agent, and $0 < c << 1$. In plain words, this velocity is a vector in the direction of the target which magnitude is the maximum speed multiplied by a factor in (0,1] that increases rapidly with the distance to the target. The seeking velocity, when computed in this fashion, causes the agent to slow down rapidly as it approaches the target, thus enabling the agent to reach a target with the necessary precision without overshooting it. However, when the target object itself can be moving away from the agent (when it is being pulled by other agents), this can potentially lead to a state where the agent stays at a constant distance from the target, neither increasing nor decreasing its velocity. A simple solution to this problem is to use adaptive weights. By changing the behaviors from purely reactive to behaviors that consider a small time window (one time step), it is possible to dynamically adapt the weights using the following formula:

$$w_i(t) = \begin{cases} w_i(t-1) \cdot (1+\delta) & \text{if } \frac{dist(t-1)-dist(t)}{dist(t-1)} < \epsilon \\ w_i(t-1) & otherwise \end{cases}$$

where $w_i(t)$ and $dist(t)$ are the weight of the *seeking* velocity and the distance to the target at time t, respectively, and $0 < \delta < 1, 0 < \epsilon << 1$. The above equation simply states that the weight is increased when the distance to the target is increasing or when it is decreasing at a very slow rate. The weight is reset when a new target is acquired. Although this strategy works in the case of *seeking* and similar cases, notice however that is not applicable in every situation. For instance, sometimes the agent needs to move away from its destination in order to clear an obstacle.

4.3 Experiments and Results

A simple scenario was designed to test the the influence of aggregated motions on the task of collective transport. Figure 4.1 shows the layout of a small world. In this world, a team of 5 agents is deployed in random positions inside a small area in one corner of the world, close to a small bar. This bar is to be transported to a destination at the other side of a long wall-like obstacle.

Agents were given 2000 simulated time steps to complete the task. Usually under 1000 steps were sufficient for the agents to find the object and move it to its intended destination. The maximum speed of the agents was 10 units per time-step (arbitrary units), with the world being a square of size 1600x1600 and the agent range of perception of radius 200 and 180 degrees of amplitude. As before, the advantage of the flocking approach was assessed by comparing the performance with that of a team of agents of identical architecture and capabilities with the important exception of not using collective movements. The agents in the *independent* team used the same FSM as the flocking team, but did not consider the position or velocities of their neighbors. Table 4.1 shows the success rate of a team of agents using aggregated motions over 20 trials, versus the equivalent team of independently moving agents. These results clearly indicate that agents using aggregate motions have a definitive advantage over independently moving agents. Notice that aggregate motions are used in three different phases or states of the task: searching, zeroing in on target, and moving to destination (once the object has been grasped). The flocking team uses collective movements in all three phases. However, after suppressing

Table 4.1: Results of collective transport for teams using aggregated motions vs. independent motions.

Sociality	Success rate
Collective Movements	85%
Indep. Movements	40%

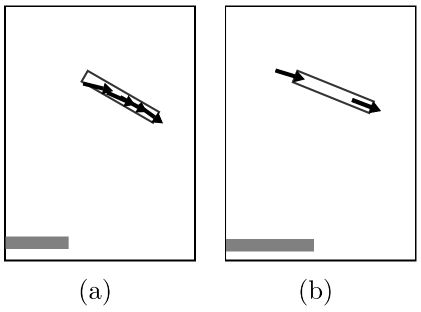


Figure 4.5: Typical distribution of agents along a piece of product. (a) Collectively moving agents (b) Independently moving agents (there are four independent agents present, two on front and two on the back).

collective movements during the moving-to-destination state, while retaining them in all others, similar results are obtained. This suggests that the importance of the collective movements comes from the distribution of the agents along the object during the initial phase of grabbing the object.

Figure 4.5 shows two examples of the typical position of the agents while carrying an object. In the case of agents using aggregate motions (Figure 4.5(a)), the agents have allocated themselves uniformly along the object, while the independently moving agents (Figure 4.5(b)) have a tendency to pile up at the same points. The most regular, organized distribution of collectively moving agents makes them more likely to successfully steer around an obstacle since, given that all agents have a different perception of the obstacle, they will turn at different moments with different radii, making the object turn more slowly and in a more controlled fashion. On the other hand, when agents tend to share the same position in the object, as in the

Table 4.2: Results of collective transport for a team carrying a heavy object vs. a single agent moving a light object.

Weight	Success rate
Heavy	80%
Light	20%

case of independently moving agents, agents will try to steer as soon as the obstacle is behind them or out of their neighborhoods, even when most of the object has not passed the obstacle yet. Additionally, the object will tend to rotate more rapidly when it is grabbed at only two points as opposed to four. The combination of these two factors make the independent team more likely to over-steer and crash into the obstacle, while the flocking team tends to move in a smoothly, controlled fashion, taking advantage of the distributed sensors of each agent.

To further study this hypothesis and the influence of the geometry and distribution of the agents along the object, a second set of experiments has been designed, similar to the previous one. In the first case, five agents are deployed as before to move a slightly heavier object. In the second case, one agent is deployed to move a five times lighter object. This light object may be moved by a single agent, and it will move as fast as the heavy object when the latter is pushed by the force of five agents; however, both objects have the same length (shown in proportion to the world and obstacles in Figure 4.1). Table 4.2 shows the results for this set of experiments over 20 trials. Although only one agent was strictly required for completing the task in the case of the light object, the single agent failed to do so 4 times more often than the 5 agent team moving the heavy object. In most cases, the single agent crashed the object against the wall, due to the over-steering phenomenon described before. This shows that a properly coordinated team will perform the task more successfully not only because the higher number of agents will provide more brute force, but also because the team can use a sort of cooperative sensing that will provide the team as a whole with more information than a single agent can acquire by itself. Notice that in Table 4.2 the success rate of the team has decreased from 85% to 80%. This difference can be accounted for by the fact that the object in these experiments is heavier. Results not relevant to the current discussion show that the difficulty of the task increases with the weight of the objects to be carried.

4.4 The Complete Collective Transport Experiment

The previous section described the capacity of aggregated motions to produce local tactics in self-organized teams. However, I propose that these movements are also able to produce coordination of the team at a global level based only on the local interactions among neighboring agents. To study this hypothesis, the collective transport task above has been extended into a problem that requires the coordination of the agents at a global level. In the complete collective transport problem, multiple sources and destinations are present in the world, and a larger set of agents may potentially split into sub-teams. The goal of the task is to distribute a certain number of objects, or products, to *every* destination in the least time possible. As mentioned before, this implies that agents may use a cross-team strategy to better allocate the resources of the team and assign sub-tasks to different sub-teams. However, since a whole team would usually spread over an area larger than the neighborhood of an agent and there is no central control nor complex communication, this strategy needs to emerge from the local interactions among the agents and to be communicated to the team in a similar fashion.

Agents follow the same general hierarchical architecture described before. A simple extension to facilitate the development of this strategy was made to the agents: a simple memory was added so that agents could remember the amount of product already deposited in each destination (to the best knowledge of the agent). This memory works basically as a table, with the positions of the destinations preloaded in the rows of the table at the beginning of the simulation. Along with the position of a destination, agents store (individually) the amount of product present in a given destination and a time-stamp of the moment when the information was collected. This time-stamp is used as a measure of the accuracy of the information, with the most recent information being considered more accurate. Agents acquire this information by directly counting the amount of product in the neighborhood when they visit the destination, or by exchanging memories with neighboring agents. This is the most complex form of communication in the system, and consists of transmitting the content of the memory (amount of product in each destination plus age of this memory) to nearby agents. When exchanging memories, the more accurate memories are preserved on a row by row basis, including their timestamps, and adopted by other nearby agents. Based on this memory, and using the same FSM and the same reflexive behaviors that were used for the collective transport task, agents complete this task by using a simple strategy: after picking up the product, the first agent to reach the object chooses as destination the one with the lowest amount of product already deposited, and agents that arrive at subsequent times will adopt the same destination. This way, agents try to maintain a roughly equal amount of product at all destinations by delivering to the destination most in need first.

4.4.1 Experimental Set-up

In order to study the interactions between agents in the described scenario, and to determine the efficacy of the adopted strategy, a series of simulations were performed. Aspects useful in determining the properties and advantages of collective movements include three factors: (1) Total number of agents in the team. (2) Communication, in this case there is at one level the agents just described before that share memories as they encounter each other. At the other level, there are non-communicating agents. The resulting strategy in the latter case restricts the cooperation between agents and is useful as a control, although the agents on these simulations still undergo collective movements. (3) Finally, we are interested in the effect of collective movements in the resultant global behavior of the agents. Therefore, comparisons are made between teams of agents that undergo flocking during the different states, and teams of agents that simply ignore each other positions and velocities thus moving independently.

As a quantitative measure of behavior and performance, the average time required for a team of agents to deposit 10 objects at every destination is used. Notice that in principle there is a simple strategy that requires no coordination at all, consisting of depositing the objects at a destination chosen at random among all destinations with uniform probability. By making some destinations have more difficult access than others, and therefore increasing the rate of failed attempts at depositing, or simply increasing the time required for a successful deposit, this simple strategy is discarded. A successful team will necessarily have to carry the objects to the destinations where they are more needed, in order to satisfy the requirements of 10 objects at every destination in minimum time. Extra objects deposited at a destination are simply discarded, and they only contribute to slow down the team working as wasted effort.

A simulation consists of deploying a group of homogeneous agents within a given area chosen at random in the world shown in Figure 4.6. It is worth noting that the destination at the center of the world is particularly difficult to access because of being at the cross point of two narrow hallways. This added difficultly was corroborated empirically in trial runs. An agent's maximum speed was set to 10 units per time steps, while the world is a square of side 3000 (arbitrary units). The maximum neighborhood of agents is 300 units, although they also know the position of the destinations (but not of the sources). Product pieces are all 60 units in length and only 5 agents can attach to a product at any given time. The force applied by an agent is proportional to its current desired velocity, and the force applied by a single agent pushing at speed 1 is referred to as one unit of force. The force required to move a piece of product is 12, which implies that at least two agents are required to move it since agents' maximum speed is 10. Notice that the velocity of the object is proportional to the net force applied on it, implying that although only agents are required to move it, it will move faster if several agents push in the same direction. Simulations end when all four destinations have at least 10 pieces of product or after 100,000 time-steps.

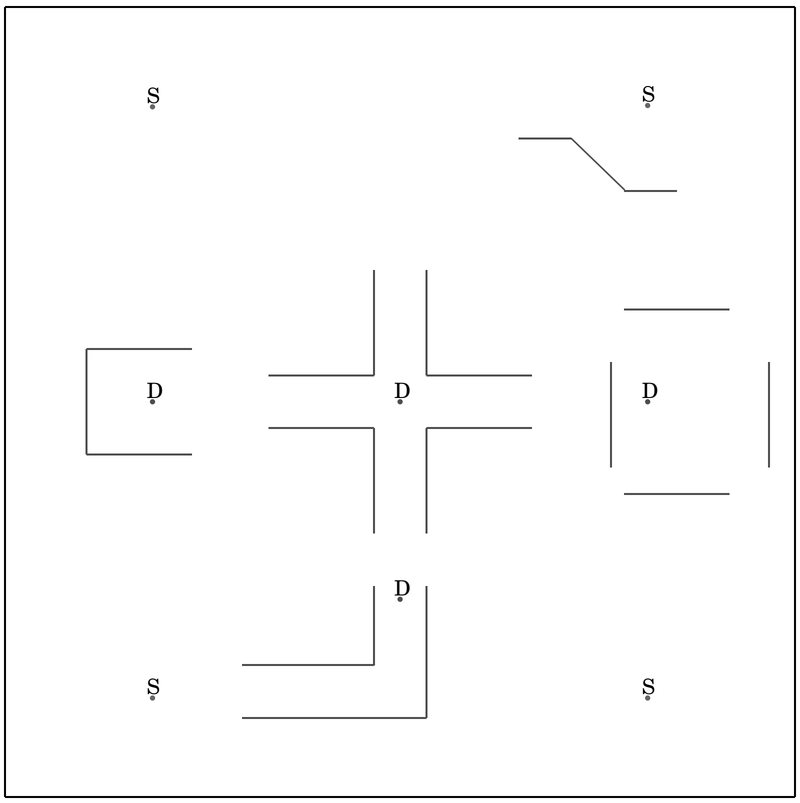


Figure 4.6: Layout of the obstacles, sources and destinations in a collective transport world. Sources are spots marked with an 'S' while destinations are marked with a 'D'.Impassable barriers are indicated by straight lines.

A second set of simulations studied the influence of the size of the product pieces on the performance of the agents and difficulty of the task. For these simulations the conditions the number of agents was kept constant for all runs while the size of the product was varied. The condition of only 5 agents being able to attach at any time to a given unit of product was relaxed to allow more agents to carry longer (and heavier) pieces. The results of each experiment reported in the next section are the average over 20 runs.

4.4.2 Results

Figure 4.7 shows the time required to complete the task versus the number of agents. In most instances, 10 agents are not able to finish the task in the alloted time. Although only two agents are strictly needed to move a product, a unit of product moved by a team formed by a small number of agents is prone to excessive turning while transporting an object, and to collide with obstacles, as shown in Figure 4.4. This in turn decreases the rate of success of transporting a single product which contributes to smaller teams taking more time to complete the task. When teams are formed by 25 or more agents, enough agents are already traveling together to overcome this problem, so adding more agents to the team does not have an impact as significant as it has for small teams.

Clearly, for sufficiently large teams (over 40 agents), the agents that both communicate and use aggregated motions outperform the other teams (the differences between collectively-moving, communicating team and each of the other teams for teams of 65 agents are significant at the 95% level according to Wilcoxon rank sum tests). However, teams that use communication but not aggregate movements achieve a performance comparable to non-communicating, aggregatively moving agents for sufficiently large teams. This suggests that even teams that do not

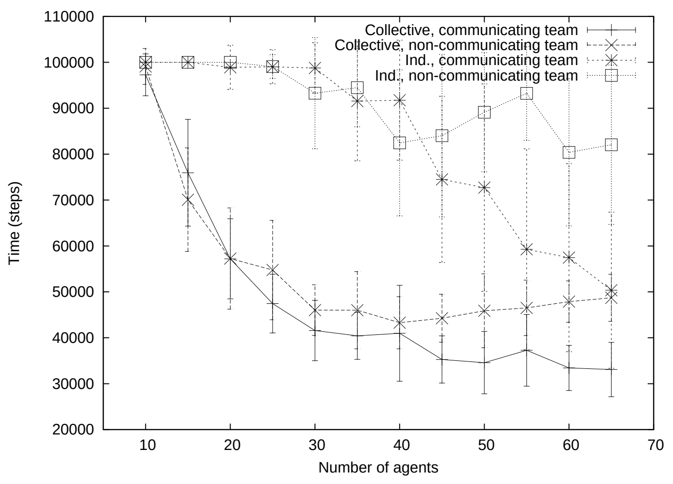


Figure 4.7: Time required by a team of agents to deposit at least 10 units of product in every destination. Time is shown with standard deviations.

use aggregated motions benefit from a high concentration of agents.

Interestingly, in the case of agents using aggregate motions, the performance of the team as compared to its size reaches a plateau after about 50 agents, after which it could be considered that the world is saturated and the performance will not improve by adding more agents to the team. However, in the case of aggregating agents that do not communicate, the performance not only does not improve, it actually worsens. The lack of communication in this team causes the agents to hinder the work of each other: agents that possess inaccurate information about the world, and where the product is needed, tend to carry it to the inappropriate destination, taking it away from other agents.

This can be better observed in Figure 4.8, which shows the maximum difference in the amount of product deposited at two different destinations at any point in time (averaged over 20 trials) for a team of 60 agents using aggregate motions. Ideally, a perfect strategy would keep this difference at a maximum of 1, implying that product is always delivered to the destination that needs it the most. In practice, it is difficult for teams to take into account units of products that are in transit. In the case of the communicating team, this difference increases rapidly in the first steps and stays at around five for the rest of the simulation. This shows that the team as a whole stays informed of the state of the various destinations and individual agents can properly decide where to deliver the product. In simulations, however, it is observed that different sub-teams will rush product to the same destination at the same time, causing the difference of about five units of product to one destination over another, implying that although the information about the general (global) state of the world is transmitted efficiently, agents do not possess information about the current actions of other sub-teams. In the case of agents that do not communicate, the imbalance in the amount of product delivered steadily increases at the beginning of the simulation and stays as high 20. This implies that a destination is overflowed with at least 30 units of product although only 10 are needed. This clearly shows the inefficient allocation of resources by the non-communicating team due to the outdated information held by each agent.

Figure 4.9 shows the results for the last set of simulations. In these simulations, 60 agents that share memories and flock as described before are required to perform the same task. The experiment was repeated for different sizes of products. Figure 4.9(a) shows the time agents took to finish the task. As expected, the time increases with the size of the pieces of product, not only because this pieces are moved more slowly (due to the increase weight), but also because the problem becomes harder with the increased difficulty of steering the product around obsta-

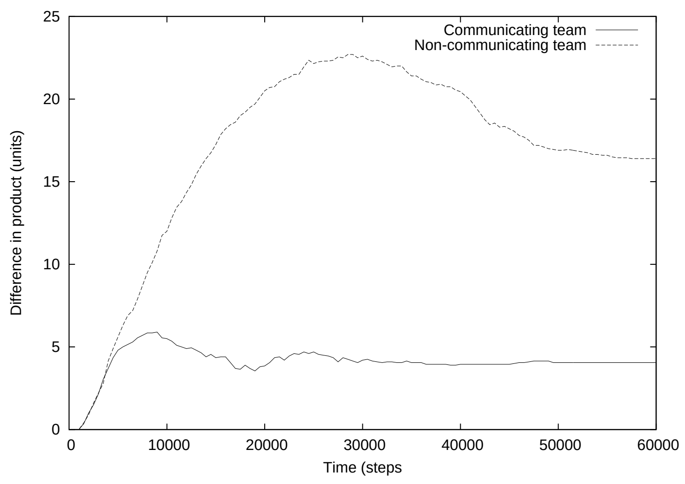
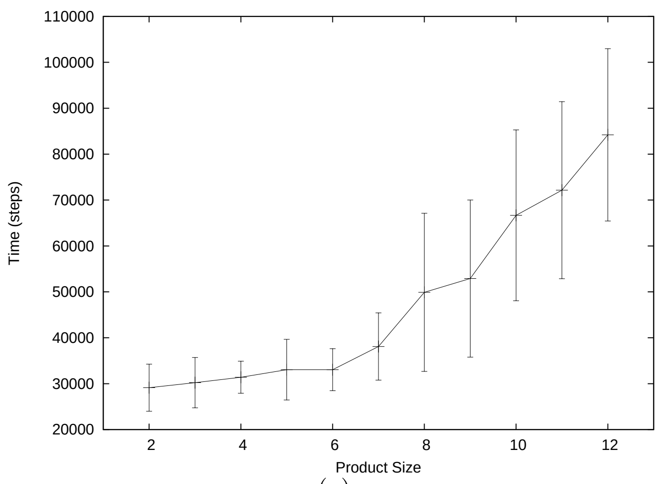
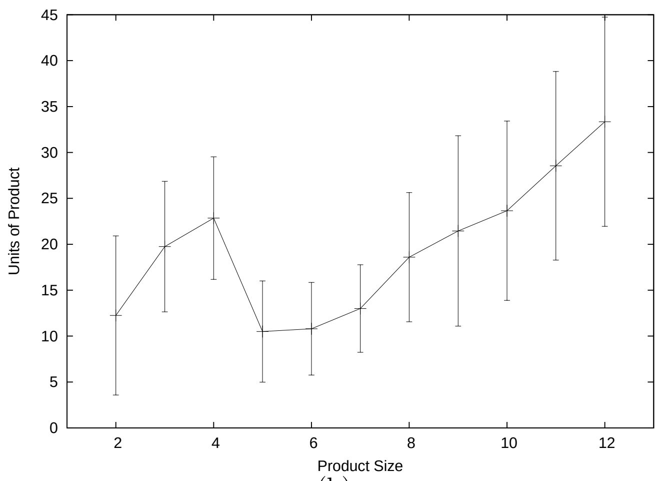


Figure 4.8: Results of a simulation for teams of 60 agents. The maximum difference between the amount of product deposited in any two sources at each time-step is shown.

cles. Similar results can be observed when varying the density of the pieces while keeping the size constant. Figure 4.9(b) shows the number of pieces that were not allocated in a destination by the end of the simulation. This number represents the pieces that agents tried to move but eventually dropped in some place other that the intended location, either because they collided with an obstacle or could not move them for some other reason. The fact that the number of undelivered pieces increases with their size confirms that the problem becomes harder for larger pieces. Most interestingly, it can be observed that the number of undelivered pieces decreases sharply for size 5 and increases slowly again for larger sizes. The number of agents that can attach themselves to a piece of product is limited by its size given the tendency of the agents to avoid collisions with other agents. Therefore larger pieces can be carried by a larger number of agents whose distribution along the product also gives them essentially different visual fields. These facts allow the kind of coordination explained in section 4.2.3, making medium-sized pieces easier



(a)



(b)

Figure 4.9: Results of simulation for teams of 60 agents with different sizes of product. (a) Time required to complete the task (shown with standard deviation) vs product size. (b) Number of undelivered pieces by the end of the simulation (shown with standard deviation) vs product size.

to transport than short pieces.

4.5 Discussion

The previous chapter showed the feasibility of using the extended particle systems controlled by a multi-level architecture as a problem-solving mechanism. To further study the properties and capabilities of the proposed model, this chapter presents a second set of experiments where agents solve different problem. The collective transport problem extends the difficulty of the first problem not only by adding obstacles, but also by requiring every object to be transported by more than one agent. The collectively moving agents proved able to complete this new task, confirming that such agents can solve more challenging problems. Further, these simulations showed that simple collective movements can produce cooperation between self-organized sub-teams, as seen for example when a group of agents coordinated to make a bar-shaped object turn as it went around a corner.

Several approaches have previously been employed in the solution of the basic collective transport problem [31, 30, 62, 66, 69, 76]. The key difference in the experiments shown in the first part of this chapter is that agents of the extended particle systems are capable of solving a basic collective transport problem using only partial information of the world and no explicit communication, merely by the use of reactive behavior. That is, no form of coordination or planning except the essential reactive movements that allow the agents to move in aggregations is used, yet the interactions among them are capable of implicitly transmitting information, effectively transforming a team of agents into an array of mobile sensors that cooperate based on this shared information to navigate the world while collectively avoiding obstacles and carrying an object.

The second part of this chapter presented a more challenging problem constructed as an extension to the first one. The complete collective transport experiment requires agents to transmit information over large areas so as to achieve a coordinated global strategy by self-organization. As in the previous problem and also the previous chapter, the experiments show that the extended particle system is capable of producing the necessary team and sub-team global division of labor exclusively through local interactions. However, the experiments also show that agents who possessed capabilities of local communication performed significantly better at the task, and were in general better able to achieve an appropriate knowledge of the conditions of the world, which leads them to developed a more successfully shared strategy. Since this communication occurs at the local level, these results that simple, explicit local communication among neighboring agents, in cases when it is feasible to be added to the agents' capabilities, can greatly impact the distribution of information and self-organization process.

Chapter 5

Distributed Learning

In this chapter, I extend the model of collectively-moving agents previously presented, and incorporate distributed learning into the system. In previous chapters we have seen that particle systems with a top-down controller at the particle level are capable of relatively sophisticated examples or coordination, such as self-organization in task division and the development of global strategies. To further study the intrinsic capacities and benefits of such systems, we turn to the question combining learning with collective movements. In particular, we are interested in distributed learning, where the independent contributions and adaptations of each agent or particle are combined through the self-organizing collective behavior to achieve a global solution to a problem that is beyond the scope of any of the individual agents. For this objective, a special optimization problem is developed, inspired by both flow and scheduling in networks and collective transport. This problem combines the issues of dynamics in the physical (simulated) world, and the logic, abstract aspects of graph theory. A formal model of the problem is presented, and a simple learning algorithm, inspired by techniques commonly used in swarm intelligence, is developed specifically for the model of particle systems described earlier.

The results of this chapter show that agents can successfully learn to solve the task with a comparatively superior performance, implying that this particular type of learning is suitable for particle systems. More importantly, agents using distributed learning outperformed agents using a similar, global learning strategy, indicating that distributed learning, when combined with the self-organizing nature of particle systems, presents a powerful tool to solve complex problems. Additionally, the results show that collective movements, or flocking, improve the performance of the agents, both those using distributed learning and those that do not, providing more evidence that collective movements offer important benefits for problem solving, even in contexts where there is no explicit navigational component to the problem, or where these components are of secondary importance.

5.1 The Logistics Experiment

As a case study for this chapter, I have designed what hereafter will be referred to as the *logistics experiment*. Imagine a factory with several stations, each one manufacturing a possibly different commodity. Each job station requires one or several commodities as input to produce its own output. A physical distance separates the stations, implying that commodities must be transported between them. This is the job of the agents, specifically teams of mobile agents that move around the factory floor collecting commodities from some stations and depositing them in others. Among the stations we distinguish a set of special stations that do not require any input as *sources*, representing the external input to the system, and a set of special stations whose output is not required by any other station as *sinks*.

The sinks output the final products manufactured at the factory and represent the external output of the system.

The problem consists of maximizing the amount of commodities produced by the sink stations over time by using the agents to distribute the commodities among the stations according to the capacities and requirements of every station, and the demand and availability of commodities.

This problem presents several features that make it particularly suitable for the study of distributed learning:

- Dynamic conditions in the environment, such as changes in the rate of production in some stations, imply agents need to continuously learn and evaluate their strategy.
- Restricting agents to use only local information about the state of the environment, the learned solution needs to be distributed among the agents, since no agent has knowledge of the whole extent of the problem, and the global solution is obtained by combined behavior of all agents.
- Agents cannot immediately see the impact of their actions in the solution of the problem, providing a delayed-reward learning scenario.

As in previous chapters, agents move in a continuous 2D world that, in our experiments, is simulated to resemble the physical world. The connections between stations do not represent actual constraints in the physical world, and therefore the problem of the movement of the agents and the collective transport of objects remains as it has been treated in Chapter 4.

5.1.1 Possible Applications

The logistics experiment in its abstract form, that is, excluding the dynamics and other issues of movement and coordination in a continuous world and focusing only in its properties as an optimization problem in graphs, falls into the general category of sequencing and scheduling problems [64]. In particular, it can be seen as a problem similar to the generalized job shop scheduling problem, where the commodities are jobs that need to be transported between shops (nodes). Traditionally, jobs require going through all of the shops, each job in a specific order or partial order. In this case, there is a redundancy of shops, in the sense that some of the shops are equivalent and some of the shops are completely superfluous for some jobs. The job shop problem is known to be NP-complete in several of its forms [47, 50, 101], and multiple heuristic and local search techniques have been used in an effort to find good solutions, such as genetic algorithms [32], tabu search [12, 36], simulated annealing [104], ant systems [34], etc.

5.2 Modeling

The relationship between the network of stations in the logistics problem can be naturally modeled as a labeled directed graph, where nodes, representing the stations, are labeled with the byproducts manufactured by the station, and the production rate of the station, i.e., the probability that the station will output its product given that it has received the required sub-products. Edges are labeled with the distance between nodes. An edge from i to j is present in the graph if the byproduct of i is transportable to j and j may require product from i. In this context, *sources* are nodes with no incoming edges and *sinks* are nodes with no outgoing edges. Although this graph is not explicitly present in the world in which the agents move, the agents implicitly know about the location and dependency relation between nodes, which have been given to the agents and stored in their memories.

Given this definition of the environment, several diverse strategies are possible for the agents to distribute the commodities. However, for the purpose of this work, we will define a strategy as a dynamic probabilistic assignment from agents to edges, in such a way that each agent a that is not currently transporting commodities, will choose an edge $< i, j >$ with probability $p^a_{i,j}$ and transport an item from i to j if one is available (produced by i and not already moved far from it). It is possible for an item to need to be transported by more than one agent, which affects the time required for the item to be transported. Notice that by assigning an agent to an edge, the agent will try to transport commodities between the stations at the extremes of the edge in the proper direction; however, the agent is not restricted to move along the edge as the latter is not physically present in the environment.

Therefore, a solution to the problem consists of a set of probabilities $p^a_{i,j}(t)$, with $\sum_{i,j} p^a_{i,j} = 1$, such that the agent a will serve the edge $< i, j >$ with probability $p^a_{i,j}(t)$ at time t, for all adjacent nodes i and j, where t is an instant in time from 0 to t_{max}. An optimal solution consists of a set of such values that maximizes the sum of the outputs of the *sink* nodes over time.

Considering this model at the macroscopic level, i.e., summarizing the effects of the agents as that of a single agent for a short period of time, we can study the probabilistic result of a given solution, defined by the strategy described above and the specification of all the parameters in the model, among which we are specially interested in the probabilities $p^a_{i,j}$ for all agents a.

Formally, a **logistics problem** consists of a tuple $< G, s, S, M, l, p, r, z, e, c >$ where:

$G =< V, E >$	A directed acyclic graph. Where V is a set of nodes and $E \subset V \times V$ is a set of edges.
$s \subset V$	The set of source nodes.
$S \subset V$	The set of sinks, with $s \cap S = \emptyset$.
I	The set of intermediate nodes implicitly defined as $V - (s \cup S)$
M	A set of classes of commodities to be produced, with $\|M\| < \infty$.
$l : E \to \mathbb{R}^+$	A function representing the distances between adjacent nodes.
$o : V \to M$	A function representing the classes of commodities produced by each node.
$r : V \to \mathcal{P}(M)$	Classes of commodities required by each node, with $r(n) = \emptyset \;\; \forall n \in s$.
$z : V \to (0, 1]$	Productivity rate of each node, defined as the probability that a node will produce a commodity at a given time assuming its requirements are satisfied.
$e : \mathbb{R}^+ \times \mathbb{R} \to [0, 1]$	Efficiency of a team of agents over a given distance, that is, the

	fraction of units of a commodity that a number x of agent can transport over a distance l per unit of time is $e(l, x)$.
$c \in \mathbb{N}^+$	Total number of agents.

In the graph G, each node represents a station, while the edges represent the dependencies among them. The productivity rate models the amount of commodities a station can produce per unit of time, assuming all of its requirements are met. The efficiency of a team of agents consists of the fraction of commodities that a set of agents of a given size can transport per unit of time. This is assumed to be a continuous function but not necessarily linear nor monotonic, since a large team of agents present at the same place at the same time could actually result in agents hindering each other, negatively affecting its performance.

Assuming that at a given instant t in time each agent a will visit the edge $< i, j >$ with probability $p^a_{i,j}(t)$, and that the time required for agents to travel between agents while not carrying items is dismissible, the expected number of agents at edge $< i, j >$ at time t is

$$E(N_{i,j}) = \sum_a p^a_{i,j}$$

with the fraction of all agents $x^t_{i,j}$ at that edge being

$$x^t_{i,j} \doteq \frac{1}{c} \sum_a p^a_{i,j}$$

Intuitively, the amount of the output of a source node is simply its productivity rate, thus $v_j = z(j)$, where v_j is the output of node j. In the case of intermediate or sink nodes, the output depends on the amount of required commodities received by the node. If we call $\sigma_{i,j}$ the amount of commodities transported from node i to j, and m is one of the classes of commodities required by j, we have that $\sum_{o(i)=m} \sigma_{i,j}$ is the amount of commodities of class m received by j at a given time. The minimum over of this quantity over all classes of commodities required by j is the amount of requisites of j that are completely satisfied. Taking into account j owns productivity $z(j)$ and putting it all together, the output of node j at time t is defined as

$$v_j^t = \begin{cases} z(j) & if\ j \in s \\ \min_{m \in r(j)} \{ \sum_{i:<i,j> \in E \wedge o(i)=m} \sigma_{i,j}^t \} z(j) & otherwise \end{cases}$$

The amount of commodities transported from i to j is a function of the output of i, v_i, the efficiency of the teams of agents assigned to that edge over the distance between i and j, $e(l_{i,j}, x_{i,j}^t c)$, and the fraction of this commodities that are carried away from i towards other nodes, $1 - \frac{e(l_{i,j}, x_{i,j}^t c)}{\sum_{<i,k> \in E} e(l_{i,k}, x_{i,k}^t c)}$. Assuming for simplicity that this relationship is linear we have

$$\sigma_{i,j}^t = v_i^t e(l_{i,j}, x_{i,j}^t c) \frac{e(l_{i,j}, x_{i,j}^t c)}{\sum_{<i,k> \in E} e(l_{i,k}, x_{i,k}^t c)}$$

Therefore the logistics problem consists of finding the set of values $0 \leq x_{i,j}^t \leq 1$ that maximize the sum of all $v_j^t \quad j \in S$, that is, find

$$arg \max_{x_{i,j}^t} \sum_{0 \leq t \leq t_{max}, j \in S} v_j^t$$

This sum will be referred as the **total output** of the network.

For simplicity, the objective function is defined as the total output produced by all sink stations over time. However, this function considers all sink stations to be equally important and interchangeable, since this maximum can be achieved by making zero the total production of some stations in order to increase the production in others. Although it is not explicitly stated in the objective function and therefore not required, other things being equal, solutions that do not dismiss or over-favor some sink nodes are preferred. This criterion will be taken into account as a secondary, subjective measure of quality of the solutions. Other objective functions are clearly possible, for instance weighted sum of the production, either by time or node, balance of the flow, number of edges/nodes actively served/exploited, etc.

Some remarks about the objective function:

- The function is non linear, independently of whether e is linear.
- A feasible solution does not require the flow to be balanced, this is, the sum of inputs to a node to be equal to its output. However, the optimal solution is balanced, after considering the production rate of each node.
- It is not necessarily true that a set of values $x_{i,j}$ that maximizes the output at time t will also maximize the total output. However for practical purposes we will sometimes consider a static version of the problem as a snapshot of the network, where the values $x_{i,j}$ remain "frozen" in time.

Additionally, notice that the solution is local and distributed among all agents in the sense that no agent needs to make a full path from a *source* node to a *sink* node, nor does it need to know what other agents do, although it is affected by

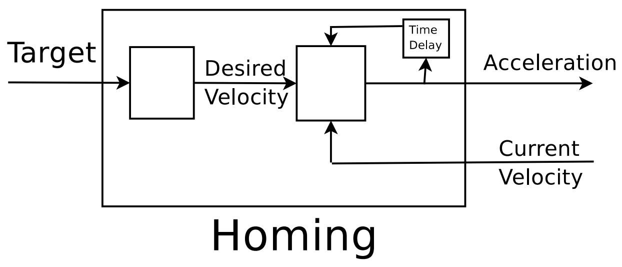


Figure 5.1: Example of a closed-loop behavior: Homing. A relative target location is input to the system. Using the orientation and distance of this location, a desired velocity is then computed (leftmost inner box). The desired velocity is then combined with the current velocity of the agent and the acceleration at the previous time-step to output the desired acceleration of the agent in the next time-step.

that. Second, an optimal solution does not necessarily represent a shortest path in the graph, nor any path in the graph for that matter, but an optimal dynamical flow, meaning the flow at any given time and possibly the schedule of changes in the flow, which both balances the flow of byproducts in the graph and exploits the most productive paths.

5.3 Solution Using Guided Self-Organizing Particle Systems

As further refinement to the ideas presented in Section 4.2.4, an extension is done on the basic reactive behaviors or forces that control an agent at the bottom layer, with the intent of letting the basic behaviors deal with more of the additional complexity of the problem at the dynamic level, in such a way that the top layer can be used to deal with the higher logic of the problem.

This extension consists of substituting some of the purely reactive behaviors by simple closed-loop controllers. Figure 5.1 shows such a controller for the case of the homing behavior. As an undetermined number of agents move an object of undetermined mass to a given target location, the purely reflexive behavior, which moves the agent in the general direction of the target, might not provide a sufficient level of control over the movement of the agent in the varying circumstances, causing the agents to stall even when they could collectively exert the necessary force to move the object, or to move exceedingly fast, spin out of control, etc. These aberrant conditions are caused by the fact that agents do not communicate during the transport of an object, making it impossible for them to compute the net force being applied on the object, or even whether they are all moving it to the same target. A closed-loop controller provides the necessary feedback to an agent to self-regulate its movement in a more controlled fashion without adding more inter-agent communication. This simple controller will continuously increase the force on the direction of the desired velocity as long as the current velocity of the agent does not reach that desired velocity, and continuously decrease the force (or increase it in the opposite direction) when the current velocity exceeds the desired velocity.

5.3.1 Agents' Strategy

At the agents' level, the strategy previously described of probabilistically assigning agents to edges is implemented by the scheme depicted in Figure 5.2. The starting position of the agents has little impact over time and therefore is set at random. Agents start in the *choose* state, in which each agent immediately chooses an edge to serve according to its own probability table. This table, of the format $(p^a)_{i,j}$ contains an entry for every edge $< i, j >$ in the graph, and its value at position

$p^a_{i,j}$ represents the probability that the agent will serve the edge. Each table owns a table independent from other agents, and the probabilities are normalized for each agent. The way in which these $p^a_{i,j}$ values are found is specified in the next section. Once an agent has chosen an edge, that is, a pick up and a drop point, the agent travels to the pick up point, or tail of the edge. After arriving to this point, it is possible that no items are available for pick up, in which case the agent will choose a new edge. Otherwise, the agent will proceed to pick up the item. Notice that any item present at the station may have already been picked up by other agents, since an item is considered available for as long as it is in the neighborhood of its producing station. After this step the agent will try to advance toward the drop point, or head of the edge, while carrying the item with it. If stagnation is detected, that is, the agent does not advance for a period of time, the item will be dropped and the agent will choose a new edge to work on. Stagnation can be caused by several reasons, among them are the fact that different agents may attempt to carry the same item toward different destinations, or the item could be too heavy to be carried by a small number of agents, etc.

Notice that at two states in this process agents have some liberty of movement, those are the "move to pick up" and the "move to drop point" states. Although the goal is clearly specified, and the environment is presumed free of obstacles, agents still have a choice on how to get there. Specifically, there potentially exists a difference when agents engage in flocking behavior during the state "move to pick up point", meaning that agents that have independently decided to go to same pick up point at about the same time, and execute some form of flocking when in the

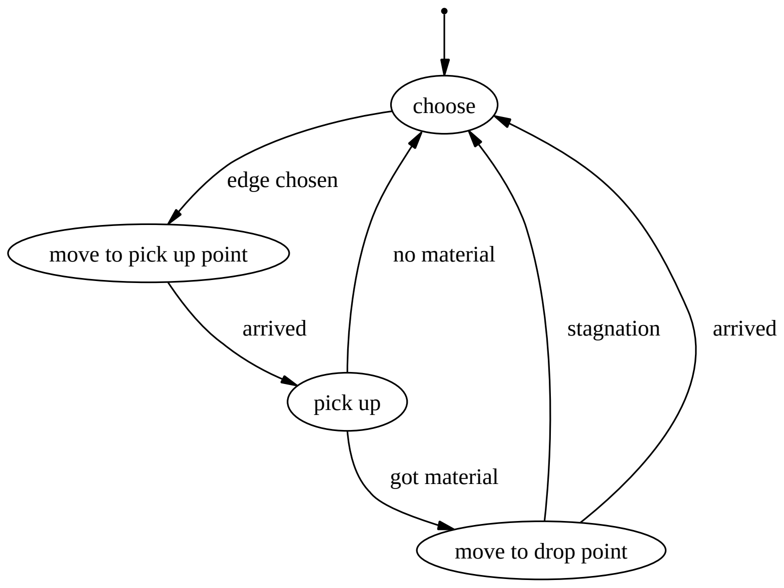


Figure 5.2: FSM of an agent for the logistics problem. States are represented as globes while transitions are represented as arrows labeled by the event that triggers them. The initial state is marked by a dot.

neighborhood of each other, obtain a different performance on the task. This will be discussed in more detail in Sections 5.5 and 5.6.

The solution deals with several details and “nuances” that are not present in the mathematical model of the logistics problem, such as the time required for idle agents (agents not carrying items) to move between nodes, the path between nodes, coordination between agents present at the same station, the presence of faulty agents, etc. Therefore, this solution is more suitable for a concrete, closer to the real world implementation than for the model of the problem that only takes into account the average effect of the agents.

5.3.2 Learning Algorithm

In the previous section, the scheme for an agent-based solution was presented while omitting the details for the process of finding the probabilities $p^a_{i,j}$ of each agent serving a specific edge, which defines the solution as stated in Section 5.2. In this section we will present this process.

This process consists of each agent dynamically adjusting its own $p_{i,j}$ values based on local information available to it, effectively learning the solution on-line. The learning algorithm is partly inspired by previous algorithms based on insect societies, specially ants [39, 40, 93]. These algorithms are based on the idea of agents working on different sections of a graph and communicating by laying pheromones, or digital markers, to convey information about the solution. In these aforementioned algorithms the pheromones are assumed to be produced by the agents. In our case, however, we require these digital markers to be produced by the stations, transported by the agents and deposited at different stations. Therefore, it resembles the behavior of bees more than ants and the way such behavior is reinforced by the presence of pollen rather than ants' pheromones [26, 27]. Otherwise there is little resemblance with other processes carried out by bees, for instance agents do not use explicit forms of message passing or communication as bees are known to do [23, 106, 89].

Thus, each agent a possesses a table or list $p^a_{i,j}$ of probabilities associated with each edge $< i, j >$ of the graph (such that these probabilities add to 1), with these probabilities representing the likelihood of the agent serving a particular edge at a given time. Agents also have a certain amount of pollen, $pollen^a$, that can possibly be zero and change over time.

Pollen in this context represents the demand for a certain commodity at a given node. It is produced by the sink nodes periodically, that is, each time a fix number of time-steps pass. It can also be stored in intermediate nodes, carried up to them by agents. However, a node may have different demands for different commodities at a given time, therefore a node possesses several pollen storages, one for each commodity that it may require. This is represented as $pollen^{i,k}$, meaning amount of pollen stored at node i for commodity type k. Notice that in this notation, $pollen^{i,k}$ is pollen stored at node i, while $pollen^a$ is pollen transported by agent a.

Given these provisions, the main idea of the learning algorithm, which is shown in Figure 5.3, works as follows: Initially, all probabilities $p^a_{i,j}$ are initialized to $\frac{1}{|E|}$, that is, all edges are equally probable. An agent a chooses an edge $< i, j >$ to serve, transporting items (one at a time) from i to j. Upon dropping the item of type k (the type of output of i, $o(i)$) at j, the demand for that item at j decreases, which is accounted for by the agent picking up pollen from j, or in practical terms, $pollen^{i,k}$ decreases by one unit and $pollen^a$ increases by the same amount. As node j is being provided by node i, the demand on i must increase, in other words the demand must be propagated backwards from j to i. This is accomplished by the same agent a, depositing the pollen back in node i. However, each item produced by i may require a whole set of commodities $r(i)$; therefore, a unit of pollen is subtracted from $pollen^a$ but a unit of pollen is added to $pollen^{i,l}$ for each l required by i. It could be seen as the agent not only transporting pollen but also inducing the production of pollen by i. However the purpose of this algorithm is not be biologically plausible so these explanations are not necessary.

Given this situation, $pollen^{j,k}$ represents, at the moment that agent a is serving $< i, j >$, the dependency of j from i, so the agent adjust the probability of serving this edge again by an amount proportional to $pollen^{j,k}$, or:

$$p^a_{i,j} \leftarrow p^a_{i,j} + \alpha \cdot pollen^{j,k}$$

where *alpha* is a proportionality constant, typically between 0 and 1. All probabilities are then re-normalized to keep the requirement that they add to 1.

Notice however that the success of this strategy depends on the agent revisiting node i, which in principle it is not required to do, since the agent chooses an edge to serve stochastically after each drop.

For this reason, the algorithm is modified by adding a tendency for an agent to re-serve the last edge just served. This is, after a drop, the agent will serve the same edge again with probability ρ, and choose a new edge with probability $1 - \rho$. This also introduces other effects, such as the fact that agents will be less likely to jump in each iteration between opposite ends of the graph, and also that agents will tend to specialize, revisiting the same edges often. The artifact of forcefully revisiting an edge is nonetheless not required, as will be seen in Section 5.6.

Two additional actions are still required for the algorithm to work: (1) Probabilities $p^a_{i,j}$ should decrease continuously. This allows agents to gradually 'forget' past actions which in turn allows them to learn or explore new actions. This is

Learn $p^a_{i,j}$ on-line

```
INITIALIZATION
for each agent $a$
    $\forall (i,j) \in E \quad p^a_{i,j} \leftarrow \frac{1}{|E|}$
    choose an edge to serve, $a^{serve}$ to be $e_{i,j}$ with probability $p^a_{i,j}$
    $pollen^a \leftarrow 0$
next
for each node $i$
    $\forall k \in r(i) \quad pollen^{i,k} \leftarrow 0$
next
LEARNING
repeat
    for each agent $a$ currently not carrying commodities
        // Keep serving currently selected edge with probability $\rho$
        // or randomly choose a new edge
        with probability $1 - \rho$
            choose edge $e_{i,j}$ with probability $p^a_{i,j}$
            $a^{serve} \leftarrow e_{i,j}$
        $p^a_{i,j} \leftarrow \delta \cdot p^a_{i,j}$
        //If carrying pollen, drop it at node
        if $pollen^a > 0$
            $pollen^a \leftarrow pollen^a - 1$
            for each $k \in r(i)$
                $pollen^{i,k} \leftarrow pollen^{i,k} + 1$
            next
        end if
        if there are commodities at node $i$
            //Carry one item to $j$, transfer pollen from $j$ to agent
            //and update $p^a_{i,j}$ based on pollen remaining at $j$
            carry one item of type $k$ to $j$
            $pollen^{j,k} \leftarrow max(0, pollen^{j,k} - 1)$
            $pollen^a \leftarrow pollen^a - 1$
            $p^a_{i,j} \leftarrow p^a_{i,j} + \alpha \cdot pollen^{j,k}$
            normalize $p^a_{i,j}$ for all $i, j$
        end if
    next
    //Increase pollen in all sinks
    for each node $i$
        if $i \in S$
            $\forall k \in r(i) \quad pollen^{i,k} \leftarrow pollen^{i,k} + 1$
        end if
    next
until termination condition
```

Figure 5.3: Learning algorithm for agents in the logistics problem.

accomplished by decreasing $p^a_{i,j}$ each time the edge $< i, j >$,

$$p^a_{i,j} \leftarrow \delta \cdot p^a_{i,j}$$

where δ is a forgetting factor, between 0 and 1. (2) For similar reasons, pollen in nodes should periodically decrease, allowing the agents to quickly adapt to changes in the conditions of the demand of commodities by the node. In order to accomplish this, pollen possesses a predetermined life time after which it will disappear, meaning that nodes discard pollen that has remained in them for too long. In practice, each 'unit' of pollen is associated with a counter, initially of value zero, that increases at every time-step, and the unit of pollen is discarded when the counter reaches a predetermined threshold. This also prevents ever increasing amounts of pollen to accumulate in a single node which would cause aberrant effects on the learning process.

5.4 Heuristic Search

As a control measure, a well known standard heuristic search optimization technique is used, namely simulated annealing [61], to solve the macroscopic problem described in Section 5.2, and then apply the resulting $x_{i,j}$ values to the simulated microscopic system to find the total output of the network under the given solution. To reduce the solution space and simplify the search process, that is the space of all $x^t_{i,j}$, the problem is simplified to a static version of it, where the probabilities $p_{i,j}$ are fixed along the whole time interval $[t_0, t_{max}]$. Under these circumstances, maximizing the total output becomes equivalent to maximizing the instantaneous output at an arbitrary instant in time. Additionally, all agents are required to use the same value $p_{i,j}$. In summary:

$$x_{i,j} = x_{i,j}^{t_1} = x_{i,j}^{t_2} \quad \forall t_1, t_2 : t_0 \leq t_1, t_2 \leq t_{max}$$

$$p_{i,j}^{a} = x_{i,j} \quad \forall a \in Agents$$

Given these conditions, the problem becomes easier to solve for a heuristic local search method. However, this is due in part to the reduction in the size of the search space, which is not guaranteed to preserve the optimal solution. This is, an optimal solution for the static problem may be a suboptimal solution of the dynamic problem. Additionally, at the macroscopic level there is no difference between specialization and no specialization, or whether all agents use the same probabilities $p_{i,j}$. However, when this solution is translated to the microscopic level, specialization may be a factor that affect the quality of the solution. Nevertheless, the fact that heuristic search optimization methods use full knowledge of the global value of the objective function may compensate for these disadvantages. Thus, although solving the static problem is not an entirely fair comparison, it provides a base control as reference for evaluating the quality of the solutions found by the agent-based method.

From the array of heuristic optimization methods available [68, 78], simulated annealing was chosen [61, 103] for its simplicity and ease of obtaining acceptable results without the need for extensive parameter adjustments in comparison with other techniques [3].

In order to use this method, it is necessary to define two important aspects of the heuristic: (1) representation of a candidate solution and (2) local variation of a solution. A solution is represented as the matrix of values $x_{i,j}$, with $x_{i,j} \geq 0$ and $\sum_{0 \leq i,j \leq |V|} x_{i,j} = 1$. For all edges $< i, j >$ not present in E, the value $x_{i,j}$ is equal to 0. A local step or variation of a candidate solution consists of disturbing each value $x_{i,j}$ with probability p, where a disturbance is defined as adding a random number with zero-mean Gaussian distribution, and then normalizing values to comply with the restrains previously mentioned.

With these definitions, the search proceeds according to the simulated annealing algorithm, that is: an initial solution is generated at random and assigned as the current solution and its value v computed. Then, at each iteration, the current solution is disturbed and its value v' computed. The new candidate solution is accepted as the current solution with probability whenever it is improvement, otherwise is accepted with probability

$$p(v, v') = e^{\frac{v-v'}{T}}$$

where T is a parameter initially set to a high value and then decreased gradually. The value v of a solution is defined as $K/\text{total output}$, where K is a proportionality constant.

This form of finding a solution through simulated annealing will be referred to as *static learning* since it uses a static version of the problem, and once the values

$x_{i,j}$ are found they remain fixed, as opposed to the learning algorithm described in the previous section, that will be referred to as *dynamic learning.*

5.5 Experimental Set-up

The system previously described was tested in simulations. Each run of a simulation consists of randomly placing a team of agents in a squared two-dimensional world that contains all of the stations in the problem. The simulation is then run for a predetermined period of time and the total output of the network is recorded. As in previous chapters, the simulations advances in discrete time steps.

The world is padded with open space surrounding the stations in the boundaries. The agents have prior knowledge of the position of the nodes and the connections among them, but are free to move through any continuous position inside the world. The minimum distance between two nodes, whether adjacent or not, is two times the size of the perceptual neighborhood of an agent and at least ten times the distance at agent can cover in one time step at maximum speed. All distances are measured in arbitrary units. The nodes or stations are considered to occupy a single point in space. They instantaneously absorb an item when this item is both in the vicinity of the node and moving slower than a threshold speed. In a similar way, commodities produced by a node are instantaneously placed at the position of the node. Building upon the ideas presented in Chapter 4, items are represented as 2D boxes that can be transported by one or multiple agents. These boxes have a uniform density and, accordingly, a weight proportional to their size. The speed at which boxes are moved, and the ease of controlling them, is dependent on several factors, including shape, weight, number of agents transporting them, and point of contact between the agents and the boxes. However, for simplicity, the condition of collision avoidance is relaxed, allowing boxes to interpenetrate and go through each other. This is done in order to reduce the complexity of dealing with traffic at the nodes, both incoming and outgoing. This does not completely eliminate the problems, since there still remain situations, particularly in the case of outgoing traffic, when a box may be simultaneously picked up by several agents, possibly with different goal destinations.

For the experiments presented in the next section, several instances of the problem, represented by different graphs, were tested. The first ten instances, shown roughly in order of difficulty as judged by the number of nodes and other qualitative factors, are depicted in Figure 5.4. These simple cases were designed by hand and have the advantage of being small enough to allow a reasonable understanding of their properties and the properties of the solutions found just by mere inspection. Additionally, these cases are designed to present specific properties in terms of the type of optimal solution that may be found. These issues will be addressed in Section 5.6 when discussing the results.

An additional set of randomly generated graphs was also used, presented in Figure 5.5, again in order of difficulty according to number of nodes. Theses graphs are layered networks with a directed path from every non-sink node to a sink node. The number of *intermediate* nodes (neither source nor sink) is 16 or 36 for each graph. In each case a node may require up to 5 different commodities in order to generate its product, and a single type of commodity may be provided to it by several nodes. All instances are guaranteed to have a valid solution, in the sense that the requirements in every non-source node may be satisfied from its incoming connections.

5.5.1 Factors

Following the methodology of previous chapters, the relative performance of the collective-behavior approach was studied by comparing it to the behavior obtained by using the same architecture while excluding some of its properties, notably properties such as flocking or collective-moving behaviors. Thus, for the experiments in this chapter we identified the following relevant factors:

- **Sociality**. As was mentioned before, during the states "moving to pick up" and "moving to drop", although agents possess a defined goal or destination, they still have the liberty to adopt different tactics to arrive there. Of particular interest is the case of adopting collective-movement, or flocking, behaviors. In the situation of using flocking, agents will engage in a collective movement with nearby agents if those are present. Agents are still free to break apart from other agents. This can be caused by the agents moving to different destinations, but can also be the result of the emergent nature of flocking as described here; for instance, agents may leave a flock to join other flocks or when the flock suddenly distances itself from the agent. Agents do not need to be in the same state to perform this flocking. For instance, an agent moving to

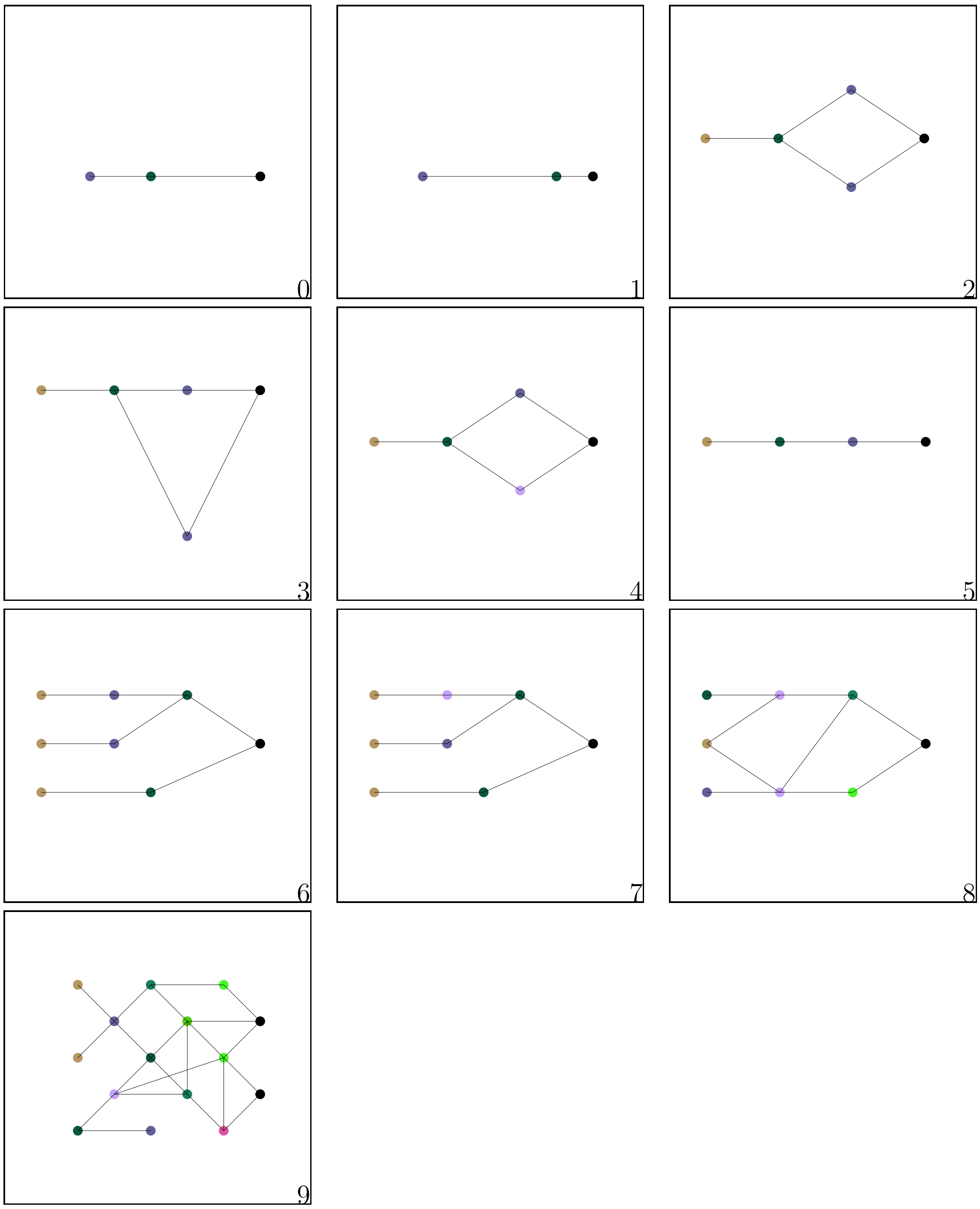


Figure 5.4: The graphs of 10 simple problems designed by hand. Graphs are numbered subjectively according to their estimated difficulty based upon the number of nodes and other factors they involve. Nodes on the left side of a graph are sources, nodes on the right side are sinks, with all others being intermediate nodes.

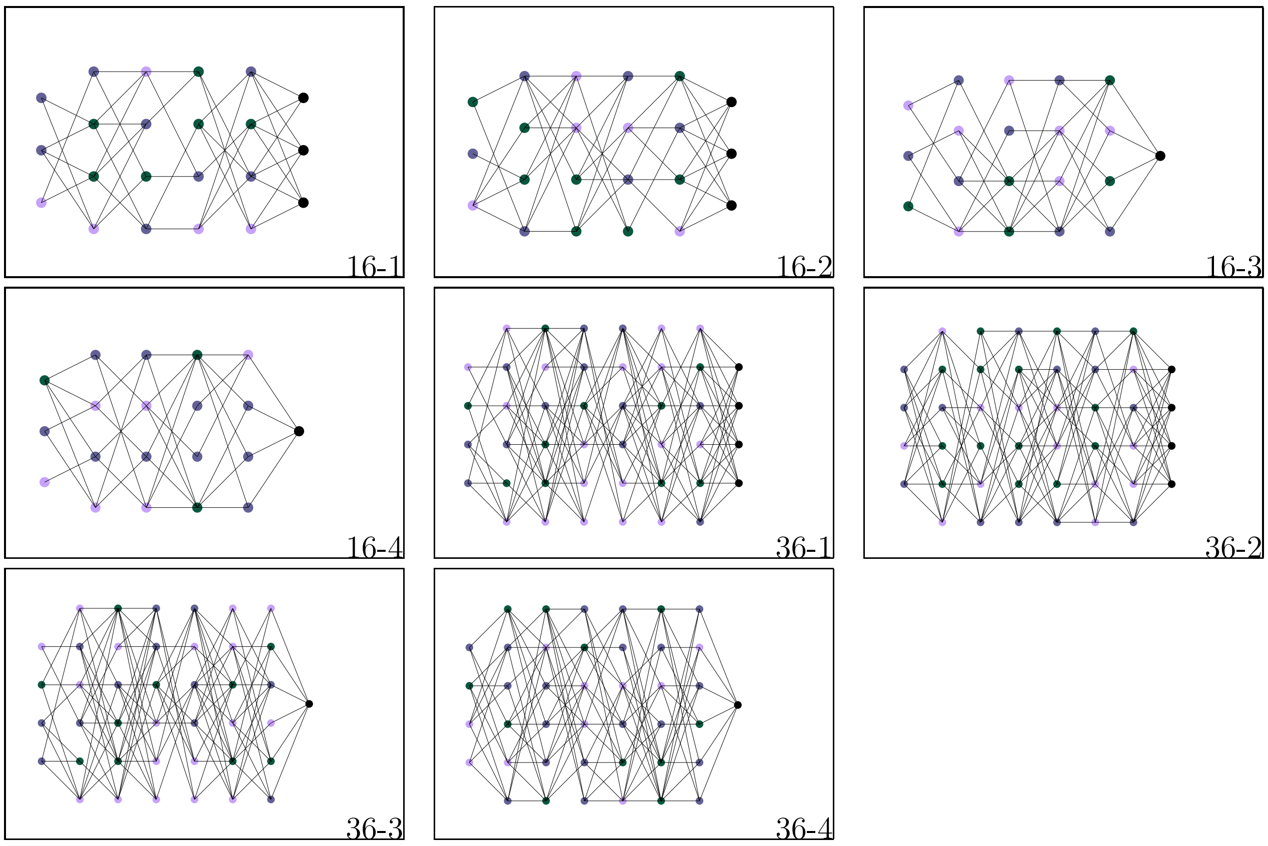


Figure 5.5: Graphs of randomly generated problems. Graphs are numbered N-M according to the number of intermediate nodes N and an arbitrary label M. All nodes on the left side of a graph are sources, nodes on the right side are sinks, while all others are intermediate nodes.

a pick up point may tend to follow, or align itself, to agents that are carrying an item. This flocking behavior is subordinated to the homing behavior that drives the agent toward its destination, meaning it has a lower priority and eventually the agent will arrive at its target point regardless of the movement of its neighbors. Therefore, two possibilities are available for the sociality of an agent: *flocking*, as just described, and *independent*, where agents move without ever letting the movements of their neighbors affect their own.

- **Adaptability**. To study the efficacy and efficiency of the learning strategy described above, the method to acquire and adapt the probabilities $p_{i,j}$ of visiting a node that ultimately determines the behavior of a team of agents as a whole in this task, we compared the learning algorithm against the simplest strategy of visiting each edge with equal probability, keeping the probabilities constant during the entire length of the simulation. Since this strategy does not present any kind of adaptation in time nor adjust to the characteristics of the particular instance of the problem, it will be referred to as *non-learning*, as opposed to the aforementioned adaptive strategy, referred as *learning*. Additionally, according to the discussion in Section 5.4, one other strategy is tested, consisting of finding a solution to the static problem employing standard heuristic search, and then using the values found this way as the probabilities for each edge to be visited during the simulation of the dynamic problem. This strategy is referred to as *SA*.

In summary, the main factors to be considered are: sociality, with levels of flocking versus independent; and adaptability, with levels of leaning, non-learning, and SA. Additionally, some of the experiments described in Section 5.6 study the effect of the number of agents in each team.

5.5.2 Parameters

For all of the experiments below, boxes or commodities are considered the same shape, size and density. The time of a simulation was set to 40,000 time-steps for the small (hand-generated) cases. The time for the large cases was 80,000 and 160,000 for graphs of 16 and 36 intermediate nodes, respectively. When interpreting this time, consider that a non-carrying agent requires at least 10 time-steps to travel the distance between two nodes at maximum speed. The size of the world was 1200 units for the small cases and 1400 and 1800 for the large cases. Agents' max speed was set to 10 units, and maximum perceptual neighborhood to 100, all these units being arbitrary. The production rate of source nodes is constant at 0.005 items per unit of time, or one item every 200 steps. The production rate on intermediate nodes is 0.1 for small cases and a random but constant number in the range $[0.001, 0.1]$ for the large cases. Regarding the number of agents in each team, it is clear that this greatly affects the result of simulations. Preliminary experiments showed, as expected, that for too small or too large teams, there is no difference between the possible strategies or solutions, because the number of agents is either too small to have significant performance, or sufficiently large to saturate the environment to the point where any strategy that visits every edge with probability larger than zero will perform equally well. A team size where a consistent difference occurs for the strategies explored was empirically found and set constant; this value is 3 for the smaller cases (0-5), 20 for the remaining hand-crafted cases (6-9), and 160 and 500 for the 16-X and 36-X cases respectively. Finally, additional parameters for the algorithm presented in Figure 5.3 are $\rho = 0.9$, $\delta = 0.99$, $\alpha = 0.5$ and max. life of pollen = 3000 steps. The agents move under the forces of four basic behaviors, as presented in Table 5.2.

Table 5.2: Parameters for Movement Behaviors.

Movement Behavior	Velocity Component	Priority	r	α
Flocking	Alignment	1	150	180^o
	Distance	2	75	180^o
	Cohesion	3	200	180^o
	Homing	4	-	-
Carrying	Distance	1	75	180^o
	Homing	2	-	-

5.6 Results

Figure 5.6 shows the total output for each of the different strategies for the hand-crafted graphs (also summarize in Table 5.3). As a first observation, notice that for the cases 0-5 there are no significant differences (at the confidence level of 95%) between learning and non-learning. Non-flocking teams perform slightly better than their flocking counterparts. However, the number of agents in a team (three), might be too small to draw any real benefit from flocking, while spreading agents across the world might result in a more effective strategy. Preliminary empirical trials show that for a larger number of agents, even as small as five, the teams manage to achieve a total output close to the maximum possible (150, given the parameters of the experiment), regardless of the sociality or adaptability, indicating that the graphs are simple enough that any strategy that causes the agents to serve each edge with frequency larger than zero will be equally effective. For smaller teams, however, small differences appear between sociality, and in some cases between adaptability. In the case of the problem 3, for instance, the non-flocking/non-learning teams show a slight adventage over the SA teams. In general, for problems 0-5, all strategies and combinations of sociality/adaptability are roughly equally effective, reflecting

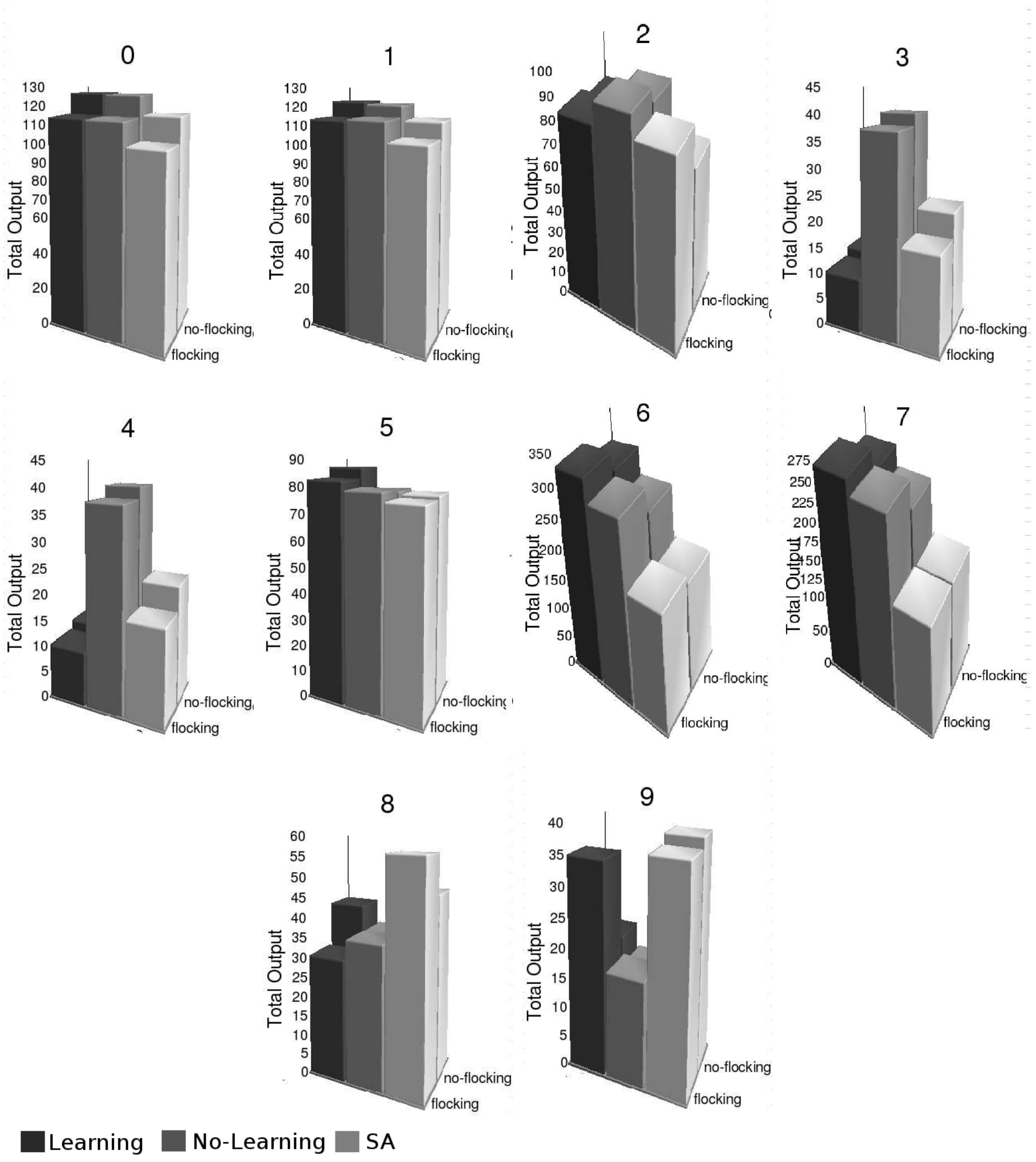


Figure 5.6: Total output for hand-designed problems after a given period of time for all 6 strategies. Results shown are the average over 20 runs.

the simplicity of the problem.

The same is not true for more difficult instances, graphs with multiple sources, as shown by cases 6-9. For cases 6, 7 and 9, there is an advantage, statistically

Table 5.3: Total output after a given period time for all 6 strategies. Values are the average over 20 runs for each case/strategy, standard deviations are given in parenthesis. Abbreviations are f. for flocking and l. for learning.

Case	Strategy					
	f./l.	f./no-l.	no-f./l.	no-f./no-l.	f./SA	no-f./SA
0	114.2 (12.6)	114.0 (7.2)	126.6 (7.7)	124.1 (6.1)	101.8 (11.5)	115.3 (9.7)
1	113.6 (15.7)	114.6 (12.1)	122.3 (15.4)	121.1 (8.6)	104.5 (10.3)	112.9 (8.6)
2	83.3 (33.3)	92.9 (6.3)	85.5 (27)	79.4 (12.6)	83.4 (7.2)	83.8 (6.7)
3	65.6 (27.5)	73.3 (11.0)	79.3 (20.6)	86.3 (5.7)	66.0 (11.6)	64.7 (6.7)
4	10.1 (10.4)	37.9 (7.3)	11.7 (10.2)	40.3 (8.4)	17.8 (9.7)	22.5(14.0)
5	82.9 (26.9)	79.5 (14.6)	86.8 (21.9)	78.1 (8.9)	76.3 (10.5)	77.4 (12.6)
6	330.4 (13.5)	294.6 (17.3)	299.0 (9.3)	255.0 (11.0)	188.5 (4.9)	172.4 (3.3)
7	269.7 (10.7)	251.1 (12.9)	240.1 (12.9)	212.7 (24.5)	145.8 (0.9)	144.1 (2.6)
8	30.6 (11.4)	35.8 (6.1)	41.8 (18.9)	35.6 (4.9)	56.7 (4.6)	46.3 (4.4)
9	35.1 (16.9)	17.5 (3.7)	20.0 (8.1)	15.8 (2.9)	36.9 (4.9)	38.8 (2.9)
16-1	83.4 (17.2)	51.1 (7.0)	79.7 (14.0)	49.7 (9.0)	25.1 (4.3)	20.4 (3.9)
16-2	18.3 (5.7)	12.9 (2.7)	16.3 (5.7)	12.1 (3.4)	18.9 (2.2)	15.8 (2.8)
16-3	77.9 (13.7)	33.1 (3.3)	54.1 (17.5)	33.1 (5.1)	20.7 (5.5)	18.3 (4.5)
16-4	33.0 (13.4)	28.5 (4.5)	29.0 (11.0)	27.0 (3.3)	21.0 (4.6)	21.4 (4.0)
36-1	8.3 (2.5)	5.5 (2.0)	8.8 (4.0)	5.5 (1.9)	9.7 (2.0)	4.9 (1.7)
36-2	14.7 (5.0)	2.4 (1.2)	12.0 (3.1)	2.0 (1.2)	7.9 (2.8)	5.0 (1.8)
36-3	12.7 (3.4)	10.2 (2.2)	12.3 (3.6)	9.8 (1.1)	1.6 (1.0)	0.8 (0.8)
36-4	1.4 (1.0)	0.7 (0.7)	0.3 (0.5)	0.7 (0.8)	0.7 (0.8)	0.5 (0.5)

significant at the 95% level according to a Wilcoxon rank-sum test, for the flocking/learning team versus all others, with the difference being more prominent for cases 6 and 9. In case 8, once again no differences are observed between the learning versus non-learning strategies, but the SA strategy outperforms them.

From these 10 cases, it seems that the learning strategy provides a considerable advantage for instances of the problem hard enough to benefit from it. At the same time, for these same instances, flocking proves also to be effective in facilitating performance of the task. The static learning strategies, SA, perform equal or worse than the other strategies in all but two cases, cases 8 and 9, suggesting that the changing conditions of the environment and other dynamic aspects of the problem

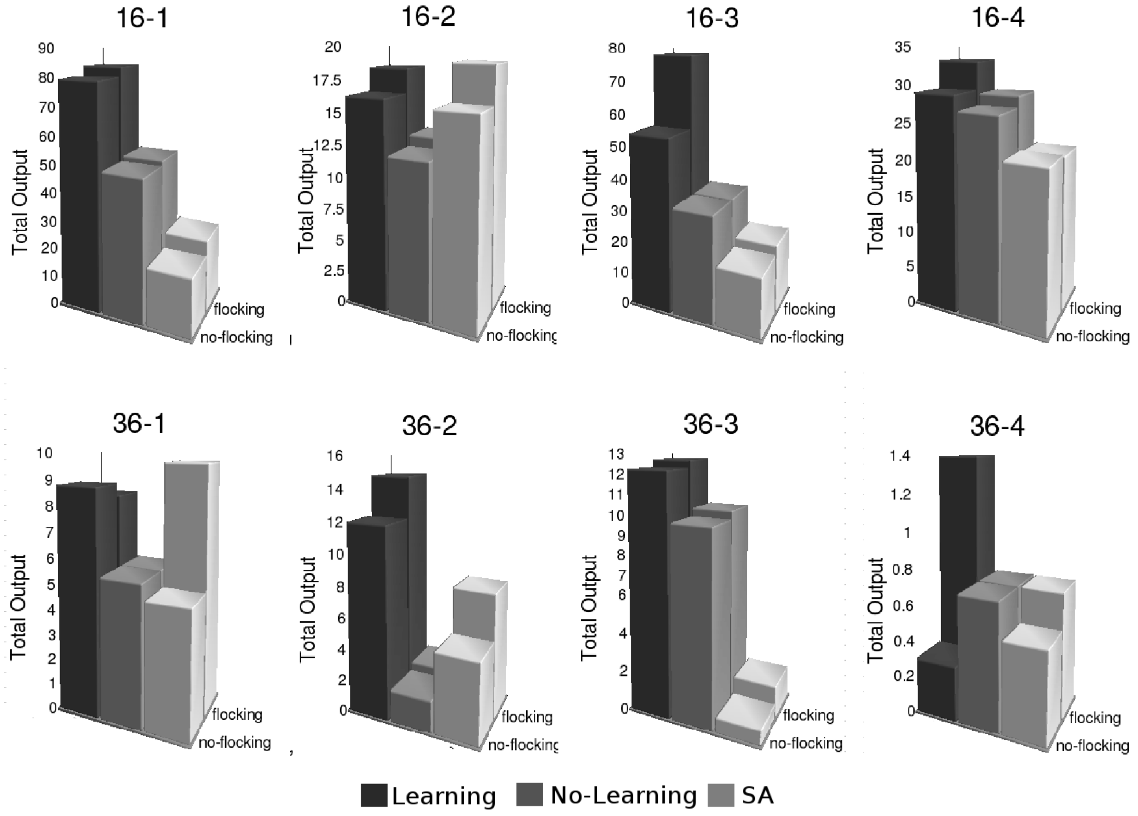


Figure 5.7: Total output for randomly generated cases using each of the 6 strategies for a given period of time. Results shown are the average over 20 runs.

render this type of approach impractical, even though it has the advantage of global knowledge during the initial phase of learning.

There is no evidence of interaction between sociality and adaptability, according to a two-way analysis of variance, in any of the 10 cases. This suggests that, at least for the hand-crafted problems, learning and flocking have no effect on each other and can be used independently to improve the performance of any strategy.

This tendency is more clearly shown in the randomly generated cases, as depicted in Figure 5.7 (also summarized in Table 5.3). For 6 out of 8 graphs tested, the learning agents proved more effective than their non-learning or static learning counterparts, with the difference always being statistically significant, and in some cases dramatic, as in graphs 16-1, 16-3 and 36-2. In the same way, flocking agents also showed a clear advantage, outperforming the non-flocking teams in each case. However, learning seems to be the most important factor in the performance of the team, as non-flocking/learning teams were superior to flocking/non-learning. This results in the flocking/learning team being the overall winner across all the cases tested except one, with the non-flocking/learning team as the second best in terms of total output. However, as the proportion of the advantage varies largely according to the case, it is not clear how the amount of the advantage can be generalized or estimated before hand. In fact, this variation does not seem to be related with the size of the problem, making it impossible to conclude whether the learning agents will have a greater or smaller advantage on larger problems. Nevertheless, it is clear from the randomly generated cases and the hand-crafted cases that the learning algorithm proves most effective for medium and large size problems.

As was for the case on the hand-crated instances, the static learning algorithm, in the form of SA performed the worst of all the strategies in most cases, with flocking/SA agents having no statistically significant differences from flocking/learning agents in only two cases (16-1 and 36-1). As with other adaptation strategies, flocking agents perform equal or better then non-flocking agents, although the difference is smaller than it is in the case of learning agents.

Once again, the disadvantage of the SA teams suggests that static, global learning presents some limitations as all agents are forced to share the same policy, as opposed to dynamic, local learning which allows agents to specialize in small areas of the problem and to dynamically change policies, working different areas of the problem at different times, which can be useful as commodities might propagate through the graph at different rates in different moments of the simulation.

Except for case 16-3, there is no significant interaction between sociality and adaptability, according to a two-way ANOVA. In case 16-3, there is a positive interaction as flocking and learning increase each others effect.

5.6.1 Entropy

As mentioned before, one of the quality measures of the solution, that is not an explicit goal of the algorithms presented here, is the distribution of commodity products among all the sink nodes, that is, not only maximizing the total output, but also minimizing the difference between the output of each sink. To formally measure this value, Shannon's definition of entropy is used [90]. Consider that at a given point in time during the experiment, an item will be produced by any of the sinks, and we tried to predict which sink will produce it. Entropy in this case represents the randomness or unpredictability of which node will produce an item, being at its highest when all sinks will produce an item with equal probability, that is, when the total production is perfectly distributed among all sinks.

Shannon's entropy is defined as:

$$H = -\sum_{i=1}^{n} p(i) \log_2 p(i)$$

where n is the number of sinks and $p(i)$ the probability that a produced item will

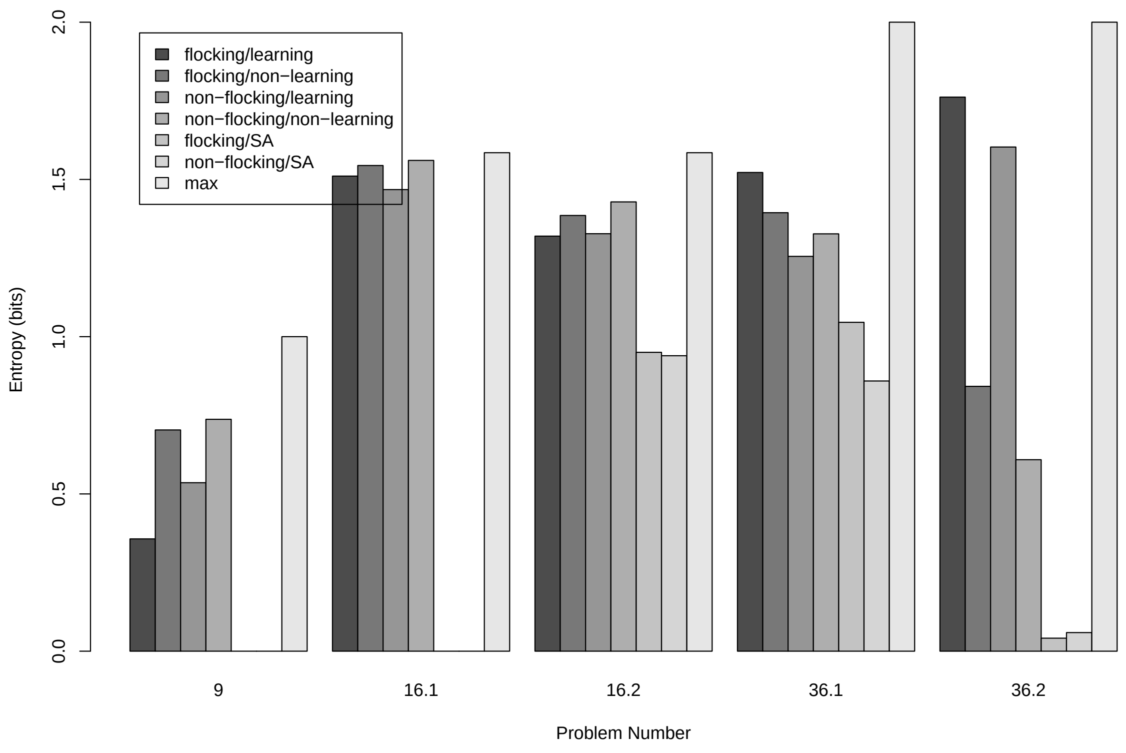


Figure 5.8: Final entropy in sinks for problems with multiple sinks, shown with maximum theoretical entropy. Values shown are the average over 20 runs.

be produced by the ith sink. The probabilities $p(i)$ are estimating using output of each sink by the end of the experiment, assuming for simplicity that this probability remains constant at all times. Although this assumption is unlikely to hold, it is of little importance as we are concerned only with the entropy over long periods of time, and more specifically with the final entropy, or entropy at the end of the experiment.

Figure 5.8 shows the final entropy for all the cases studied with multiple sinks. It shows the entropy for each combination of the strategies according to sociability/adaptability, and also the maximum possible entropy, when the output is perfectly evenly distributed among all sinks.

From these results it is clear that the static learning methods tested tend to produce poor results in terms of the entropy in the sinks. This is due to the fact that the algorithm prefers to maximize the output by allocating all agents to a single "distribution path", or the path required to supply a single sink node. As shown in the previous set of results, this strategy does not result in optimal or superior performance relative to those produced by dynamic learning, and has the effect of essentially ignoring all but one of the sink nodes, which causes an entropy of zero in some of the cases. Regarding the non-learning and dynamic learning strategies, the results are mixed, with both learning and non-learning producing similar values of entropy. It is important to notice that, in principle, it would be expected for the non-learning strategies to produce the higher entropy, since a uniform distribution of the agents could produce a similar distribution of the output. However this is not the case, as shown by problem 36-1 and especially problem 36-2, where learning agents obtain a distribution of the output of considerably higher entropy. This is an indication that the learning strategy tends to distribute the agents in a way that gives roughly equal importance to all sinks. The usefulness of the flocking behavior in this respect seems to switch from case to case, making it difficult to determine its actual role, although it has an influence in the result.

As seen in the results presented before, the dynamic methods posses a definitive advantage over the static learning. One limitation of the static learning method that may lead to this difference is, as explained above, that the dynamic methods allow changing the strategy or p^a_{ij} values of each agent over time, causing the team to reallocate resources differently at different stages of the problem. A second difference between dynamic and static learning is the fact that the static methods force all agents to follow the same assignment of edges by setting the p_{ij}^a values identically among all agents. In contrast, dynamic learning allows each agent to have a potentially different assignment of edges, which would result in the specialization of agents that service certain parts of the graph with higher probability than other agents, possibly forming sub-teams. This in fact is the case in the simulations run. Figure 5.9 shows the results for a random run of the flocking/learning team on the problem 9 using 40 agents. The figure depicts the values p_{ij}^a for each agent on a given time of the simulation. It can be observed that 12 clusters have formed (evidence for this number of clusters was calculated using the average silhouette value of the pam clustering method). These clusters basically separate the 40 agents in 12 sub-teams, each sub-team taking care of an average of 3 edges with a probability higher than 0.8, and the other edges with very low probability. Notice that this specialization occurs spontaneously as a consequence of the self-organizing nature of the system. This may provide several advantages, for instance: by allowing sub-teams to allocate themselves to small parts of a graph, a large graph can be subdivided into smaller pieces that make the problem more manageable for the agents as a whole. Also, by restricting themselves to a subset of edges, agents limit the distance traveled in comparison with agents that need to cover the whole graph.

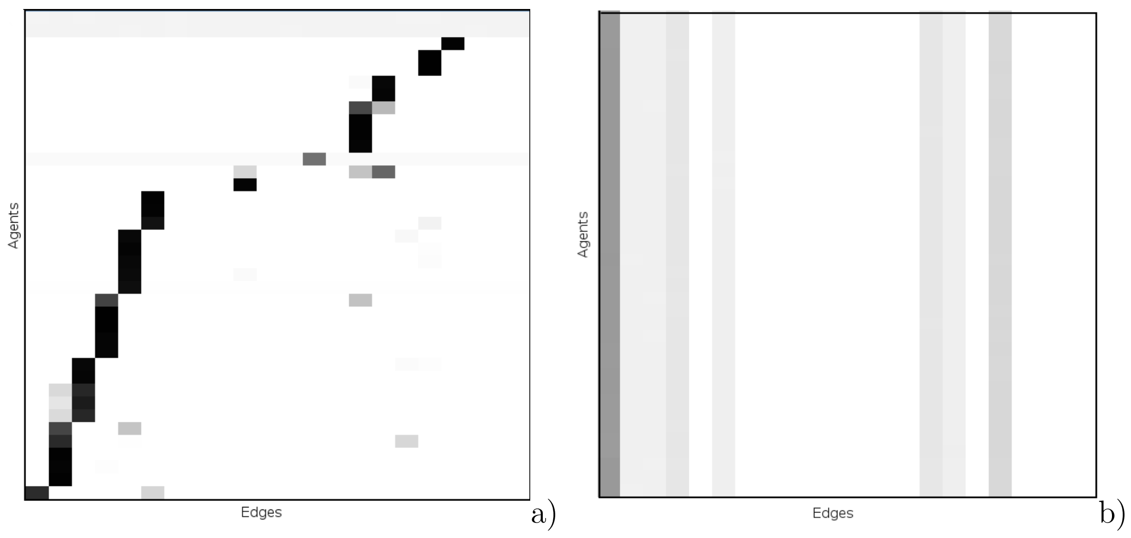


Figure 5.9: Probability of each agent serving an edge on a single run on graph 9 after 40,000 time-steps of flocking and learning (a) or flocking and SA (b). Probabilities p^a_{ij} are shown as gray tones in a matrix, with white being zero and black representing one, where each row represents an agent and each column represents an edge, giving a 40x22 matrix. Edges and agents has been arbitrarily enumerated to facilitate comprehension of the figure. Before any learning is performed, all probabilities have the same value, which would initially make these plots uniformly gray.

5.7 Discussion

This chapter presented a clear example of the application of collective movements and particle systems involving learning. In particular, we have shown how the extended particles discussed throughout this dissertation can be directly applied to problems of distributed learning. Working under the hypotheses that collective movements or flocking can effectively contribute to solve problems that require both local, navigation coordination and global strategies in an abstract space, and taking as a case study a problem that combines the dynamic difficulties of cooperative transport and the higher-level problem-solving features of logistics, we developed a simple model that can be built directly into the architecture presented in previous chapters, and presented a series of experiments that show several benefits of this approach.

For this, a simple distributed on-line learning algorithm, partly inspired by swarm intelligence techniques, was developed and incorporated in a straight forward manner into the existing agents, showing in this way the flexibility and extensibility of the approach. The resulting system was then compared through a series of experiments with a similar system using instead static, global, off-line learning through well known, general combinatorial optimization algorithms. Although in principle it could be expected that the off-line system would produce a better performance, given its use of global knowledge, the results show the opposite, the on-line distributed learning performs significantly better in most non-trivial cases.

It is not claimed that the particular learning algorithm used in this chapter is optimal or near-optimal for this particular problem, either in the on-line or off-line case. Instead, our focus is to show how learning can be readily incorporated into simple collectively moving agents, producing a form of distributed learning that enables the agents to solve problems that require a higher level of coordination to achieve a global solution using exclusively local interactions. Additionally, although no significant interaction was found between learning and flocking, collective movements proved, as in previous chapters, to improve the general performance of the system, in some cases by a wide margin, showing once again that collective movements are not exclusively useful to coordinate movements in basic navigational tasks, but that they can also be exploited to produce coordination in more general problem-solving contexts.

An important property of the learning model when applied to the particle system paradigm, as shown by the experiments for all non trivial cases, is specialization. If serving a particular edge in the logistics problem is considered as an individual task, agents will automatically and autonomously self-assign to different tasks, and during the course of learning will acquire a high propensity to perform the same task of a set of tasks while leaving the rest to other agents, thus becoming in a sense specialized. Although it is expected from the agents to show propensity to repeat some tasks, since this is encoded in the learning algorithm, it is noteworthy that a team of agents do this while distributing the tasks among its member so to guarantee that all esential tasks are acomplished, and that certain tasks get assigned higher numbers of agents according the requirements of the task. This specialization of the agents implies that the solution found by the system is truly distributed among the agents, with each agent holding only a small piece of information about the whole solution, which can only be reconstructed through the interactions of all the agents. This distribution of the information contributes to the scalability of the system, and also to its tolerance to failure: since the solution is formed by the combination of the information held by several agents, a small number of 'wrong" agents, containing outdated or otherwise incorrect information, will not significantly affect the solution.

Finally, additional improvements could be made to better exploit the specialization of the agents. Although there is no evidence that the edges served for a cluster or sub-team of agents are topologically related, the learning process could be modified to increase the probability of these clusters forming on sets of nearby edges, minimizing the distance agents need to travel between edges and increasing the efficiency of the system. Also, the fact each agent would serve certain edges with low probability (although the same edge could be served with high probability by other agents), implies that agents do not need to keep a table with the probability for each and every edge. Edges with low probabilities could be dropped entirely from the memory of the agent, reducing the amount of memory required for each agent and possibly simplifying or accelerating the learning.

These results, in addition to the experiments presented previously, show feasibility and some of the benefits of combining distributed learning and collective movements to solve learning problems that require coordination far beyond navigational tasks, and to achieve a global solution by using local information and interactions.

Chapter 6

Discussion

Previous efforts to exploit the many advantages of particle systems have faced the problem of these systems being hard to control or predict. It has been expressed that often they “seem to have a life of their own” [81]. This is not surprising since many of their features have been inspired by biological systems and incorporated specifically for the purpose of simulating life. Perhaps paradoxically, most of the systems found in the literature in particle systems are also of great simplicity, maybe due to the fact that particle systems were originally created for the modeling and simulation of entities with relatively “simple” behaviors, such as fire or smoke. These two characteristics have greatly contributed to limiting the extent to which particle systems are utilized as a problem solving technique. With the exception of numerical optimization, most successful applications have been restricted to the fields of computer graphics and video games, with the goal of animating a group of characters that move, imitating a flock, soldiers in formation, etc. However the multiple benefits of the approach still make it an attractive tool in the construction of multi-agent systems.

In this dissertation, I have proposed that particle systems can be extended, without loosing their key features, into systems with general problem solving capabilities. The resulting systems can perform tasks of interest beyond the fields of computer animation of collective movements and formation keeping in robotics, and effectively use aggregated movements and other features of swarm intelligence as essential components of the system and its ability to perform the task. As this kind of “guided self-organization” approach has previously remained largely unexplored, one of the goals of this work was to find a suitable set of tasks to exemplify and evaluate such system, as it is clear that not all multi-agent problems actually benefit from extra coordination or aggregated movements. The main goal of this thesis was to develop a mechanism to guide the self organization of particle systems, and utilize collective movements as a mean to achieve non-strictly-navigational coordination in problems that require or benefit from cooperation, exploiting this self-organization to allow teams of simple agents to solve tasks in a coordinate fashion using only local information and a minimum of communication, and to study the conditions under which such collections of agents moving in aggregations present any advantage over regular, independent agents.

These aggregated movements and collective behaviors are caused by local interactions, produced among the agents in the system by the combination of a few simple behaviors, namely cohesion, separation, avoidance and alignment, enhanced with behaviors specific to certain tasks such as homing, seeking, grabbing, dropping and the exchange of simple memories with local neighbors.

Through simple extensions to traditional particle systems, I have presented a method for guiding or controlling self-organizing systems, allowing them not only to perform more complex, goal-oriented tasks that may include pursuing a long-term goal that requires the successful completion of shorter goals or tasks, but also utilizes the aggregated movements of agents as a mechanism of local communication that can substitute for explicit communication in some cases, and as a mechanism for achieving coordination, both at the local level, when a small team of neighboring agents need to cooperate to jointly perform a task, and at the global level, when several sub-teams of agents need to self-assign different tasks and develop a shared global strategy, based only on the local information possessed by each agent. This is achieved by allowing and employing the self-organization of the system via local interactions, and guiding this process through the use of top-down influences, while retaining its original self-organizing nature by not allowing at any moment global or centralized control.

Although the modularity and extensibility of these systems allows this approach to be adapted with minor modifications to several problems, it remains difficult to adapt the system, given the unpredictability of self-organizing particle systems. This is, although the introduced methods present a way to guide the self-organizing process, this method still involves design decisions and parameter tuning that can prove difficult or tedious to perform in the general case for the (human) designer of the system.

6.1 Extending Particle Systems

Since their introduction in 1987 [81], the model of behavioral flocking, or collective movements in particle systems, proved to capture the complexity of several biological systems in a manner that was both simple and efficient. Ever since, researchers have been inspired by this model as a clear example of self-organization. By following simple rules that require only local interactions between neighboring agents, large groups of agents can coordinate in a robust manner that results in the whole crowd of agents moving or behaving in a coherent, synchronized way, acting loosely as a single organism that is capable of quickly reacting to the environment, splitting around obstacles, smoothly joining other groups of agents, in a manner that is insensitive to the number of agents present. The model has been afterwards employed in several other applications, ranging from artistic creations in computer animation to unmanned military vehicles that travel in formation. The model has also been extended following the inspiration of social human interactions, resulting in a simple, well-known numerical optimization meta-heuristic [41, 59]. However, beyond this abstract application to numerical optimization, other applications have so far remained in the field of formation keeping, whether in military scenarios, animal and insect societies or even human crowds. In this thesis, I have proposed that this model of aggregated movements cannot only create flocking-like behavior, but that this flocking behavior or coordinated movement itself can create higher levels of organization that are useful in general problem solving.

In this dissertation, I have explored the question of whether self-organizing particle systems can be extended to exhibit more general behaviors. Specifically, the hypothesis was that by giving the normally purely reflexive agents found in particle systems a few behavioral states, a simple finite state transition graph that governs state changes, and a simple memory of locations of significant objects that are encountered, the resulting agent team would have the ability to collectively solve problems requiring cooperation, competition, sharing of information and development of global strategies by agents that only interact locally, and the pursuit of several goals in a logical, coherent manner. In this scenario, individual behaviors are implemented by letting each state of an agent be associated with both a different goal and with a corresponding set of parameters that influence the individual agent's movements. This effectively couples the collectives' goals to different movement dynamics. Under such conditions, where state changes are triggered by environmental events and the states of other nearby agents in a way that retains the local nature of information processing in particle systems, one would anticipate the emergence of problem-solving abilities by an agent team as a whole.

As shown in this thesis through several systematic evaluations, collective movements can quickly and robustly transmit information over large areas without the need for explicit communication, allow agents to dynamically split or join sub-teams for different tasks according to the environmental situation, produce division of labor and spread knowledge of the environment through large areas and periods of time. They allow agents to develop global strategies even though any single agent possesses only local, limited information, and they also allow teams of agents to implicitly share information that allows for a sort of cooperative sensing that makes the agents act in coordination as by a concerted strategy even though each agent is acting in a purely reactive fashion.

All these features, considerably useful in multi-agent systems, were obtained by a relatively simple extension to the architecture of interacting particle systems that has not been used before in distributed problem solving in this fashion, and that allows incorporation of collective movements for the first time as a tool and essential component that permits a more general problem solving system. This represents a step in the path towards guiding self-organizing processes into goal-oriented, multi-step procedures that performs a given task. This is a critical point in making use of swarm intelligence and self-organizing systems as a general purpose methodology, or a new model of computation for distributed adaptive problem solving, capable of solving problems whose complexity makes them intractable to traditional approaches, or where the cost of traditional methods makes them prohibitive as compared to the relatively cheap, simple components of swarm intelligence systems.

It is important to notice that collective movements and distributed problem solving in general are not applicable or useful in every situation, and part of the difficulty of employing such techniques consists of identifying which characteristics of a problem are suitable for and exploitable by collective behaviors. From the problems studied here it seems clear that some of these characteristics include the property of benefiting from parallel work, division of labor, behavioral specialization and team recruitment. A case that does not satisfy these properties, for instance, is the search and collection of a single item positioned at a random location or of several items independently located. Empirical evidence not reported here tends to confirm the intuition that agents spreading apart from each other will cover more ground and therefore find the item faster than agents moving in aggregations. This pattern of movement can be easily achieve simply by random wandering that does not require of any cooperation by the agents. Other property present in all the problems described here, and seemingly necessary or at least useful, is the opportunity for reiterated interactions among the agents, which allows for the self-emergence to occur and produce, for instance, useful division of labor. Finally, dynamic environments or problems are specially prune to benefit from systems of emergence behavior such as interacting particle systems.

The techniques here presented still require considerable tinkering and human "intuition" to develop a self-organizing solution to a given problem, and the set of problems used so far is admittedly limited. Although neither the applicability of self-organization into a more general class of problems has been shown, nor universal guidelines have been developed on how to use collective movements for a given problem, the tasks presented include common, well-known problems often studied in the field of cooperative mobile agents, as are capture the flag, foraging, collective transport, routing and logistics, and are extensively noted in the literature as containing some of the most common issues and challenges for this type of systems. The approach in this dissertation has shown to successfully apply to these problems, with minimal variations on the architecture of the system, suggesting the generality and ease of adaptation of it to a larger family of problems.

6.2 Contributions

- I developed a flexible and extensible architecture for the application of particle systems using collective movements to a wider variety of problems. This architecture, based on a few basic behaviors and a finite state machine, allows an external entity or designer to reconfigure the system's behavior to achieve very different global behaviors, and basically different systems, simply by reconfiguring the basic behaviors. The finite state machine allows an intuitive mechanism to encode the logic of the problem or task for the system to tackle, and thus to guide the self-organizing process. This provides for a simple approach to solving a diverse range of problems through minimal changes to the architecture. Further, the system presented is easily extensible by incorporating a few basic behaviors for specific tasks or requirements. We showed how this can be attained by using essentially the same system on three different problems simply by specifying a different FSM and adding or removing basic behaviors of different levels of sophistication as needed.

- Through the use of hierarchical architectures, a decentralized control mechanism is introduced for self-organizing systems that facilitate its design for solving specific problems while retaining its self-organizing feature. This mechanism, based on state-transition control and the introduction of memory/goal into reactive particle systems, allows guiding the behavior in top-down fashion, which alleviates the difficulties in designing self-organizing systems brought by the complexity of predicting the behavior of such systems, and allows a human designer to incorporate prior knowledge of the problem by designing it into a high-level controller that follows the logic of the problem, and breaking the complex behavior of the agents into independent, simpler behaviors adequate for a single sub-goal.

- Through simulations in a series of different problems, it was shown that collective movements can be successfully exploited as a problem-solving tool that allows a team of agents to improve its performance even in non-exclusively navigational tasks, such as search-and-retrieve and collective transport. In the past, collective movements were used only as a means to simulate biological aggregations or to achieve formations for navigational purposes. In this thesis, I have shown that collective movements can be advantageous in other contexts, helping agents achieve implicit communication and coordination while solving a problem where the navigational component is of secondary importance. Although aggregation *per se* proved to be useful in, for instance, facilitating explicit communication among agents or causing agents to increase their presence in an area of interest, collective movements can produce coordinated behaviors besides aggregation. This dissertation illustrated how this behaviors can be exploited in a series of tasks/applications, such as cooperative transport, cooperative sensing, cooperative guarding, etc.

- I also integrated adaptation and learning into interacting particle systems, creating an adaptive system capable of distributed learning that uses the self-organizing properties of particle systems to distribute the task of acquiring information about the problem and its solution and then combining the segments of information stored explicitly or behaviorally in each agent into a single, coherent solution. The specialization that emerges from this process allows the problem to be subdivided into smaller sub-components, not obvious at first inspection nor predetermined by an external entity, but found by the self-organizing process, and this not only make the system parallelizable, as with most particle systems, but also allows the system to handle large, complex problems that would otherwise require high computational cost.

6.3 Limitations and Future Work

Throughout this dissertation, an extension to particle systems has been presented and demonstrated on varied examples. This extension has the main purpose of facilitating the application of self-organizing behavior through particle systems to general problem solving. This goal is achieved by introducing a higher-level control mechanism that gives to the particle system more flexibility and to an external entity, e.g. the human designer, more control or guidance over the self-organizing process itself. However, after the work on the three applications presented, it is clear that there is more work to be done. The high-level controller succeeds in adding the desired flexibility to the system, and partially succeeds in allowing easing the process of introducing higher-level logic to the behavior of the agents, but the process of designing these complex adaptive systems remains a task hard to understand, that requires considerable amount of iterative improvement based upon trial and error, and the only way to accurately predict the global behavior of the system seems to be to simulate it, which adds to the difficulty of designing a particular desired behavior. In addition to this, the self-organizing nature of the low-level behaviors increases the complexity of the situation. The introduction of high-level controllers allows breaking the low-level behavior into simpler, independent behaviors for different situations, which greatly simplifies the process of designing them. However, as in the case of the high-level controller, the process remains highly experimental and laborious for a human designer.

Because of the self-organizing nature of these systems, it is intrinsically difficuly to completely eliminate the complexity of predicting or designing their behavior. However, a different route is possible. Emerging computing, or the use of machine learning for automatically designing or improving a system's design, offers a promising alternative to coping with the complexity of self-organizing systems. Already this is done in the case of low-level behaviors through the use of evolving neural networks, that allows a system designer to specify the desired features of the basic behavior and the system is capable of developing the desired controller [8, 35, 99]. This should be also possible in the case of the high-level controller, and in fact, in the case of the whole extended particle system. The interaction between the low-level controller and the high-level controller, and the capacity of the system to assimilate errors and variations through its self-organization, contributes to the feasibility and efficacy of a method of gradual variation and improvement for the design and tuning of the system, such as evolutionary computation. This coupling of systems of emerging behavior designed by emerging methods could importantly reduce the difficulty of creating such systems, and possibly be the only feasible solution for highly extensive, sophisticated systems.

As a final comment, all three evaluated tasks presented in this dissertation involve mobile agents. However, this does not imply that all possible applications of extended particle systems are limited to this field. These examples show that the extended particle systems display features necessary in distributed problem solving in general, although their use of collective movements makes their most obvious applications in problem solving related to robotics or robotic-like systems. Nevertheless, the requirement of agents or particles moving through space does not limit particle systems to robotic applications, as has been shown elsewhere [39, 41, 59, 63]. The extended particle systems presented here are capable of complex, coherent behavior, which, if applied to problems in abstract spaces, such as problems of discrete optimization, for example, could have an important influence on the application of self-organization as a true general problem-solving mechanism, and by extension in the understanding and engineering of complex systems.

BIBLIOGRAPHY

[1] W. Agassounon. Distributed information retrieval and dissemination in swarm-based networks of mobile, autonomous agents. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 152–159, 2003.

[2] R. D. Andrea and M. Babish. The RoboFlag testbed. In *Proceedings of the American Control Conference*, pages 656– 660, 2003.

[3] P. J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*, pages 601–610, London, UK, 1998. Springer-Verlag.

[4] R. Arkin and T. Balch. Cooperative multiagent robotic systems. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 277–296. MIT/AAAI Press, Cambridge, MA, 1998.

[5] S. Aron, J.-L. Deneubourg, S. Goss, and J. Pasteels. Functional self-organization illustrated by inter-nest traffic in the Argentine ant iridomyrmex humilis. In W. Alt and G. Hoffman, editors, *Biologial Motion*, pages 533–547. Springer-Verlag, 1990.

[6] M. Atkin, D. Westbrook, and P. Cohen. Capture the flag: Military simulation meets computer games. *Proceedings of the AAAI Spring Symposium on AI and Computer Games*, pages 1–5, 1999.

[7] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[8] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviours. In C. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour*, pages 11–22, Monte Verità, Ascona, Switzerland, Sept. 8-13, 2002. University of Zurich.

[9] G. Baldassarre, D. Parisi, and S. Nolfi. Coordination and behaviour integration in cooperating simulated robots. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats VIII. Proceedings of the 8th International Conference on Simulation of Adaptive Behavior*, pages 385–394. MIT Press, Cambridge, MA, 2004.

[10] A. Barabási. *Linked: The New Science of Networks.* Perseus Publishing, Cambridge, MA, 2002.

[11] J. Baras, X. Tan, and P. Hovareshti. Decentralized control of autonomous vehicles. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 1532–1537, 2003.

[12] J. W. Barnes and J. B. Chambers. Solving the job shop scheduling problem with tabu search. *IIE transactions*, 27(2):257–263, 1995.

[13] R. Beckers, J.-L. Deneubourg, and S. Goss. Trails and u-turns in the selection of a path by the and lasius niger. *Journal of Theoretical Biology*, 159:397–415, 1992.

[14] G. Beni. The concept of cellular robotic system. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 57–62, 1988.

[15] G. Beni. *Swarm Robotics*, chapter From Swarm Intelligence to Swarm Robotics, pages 1–9. Springer Berlin, Heidelberg, 2005.

[16] G. Beni and S. Hackwood. Stationary waves in cyclic swarms. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 234–242, 1992.

[17] G. Beni and J. Wang. Swarm intelligence. In *Proceedings of the Seventh Annual Meeting of the Robotic Society of Japan*, pages 425–428, 1989.

[18] G. Beni and J. Wang. Theoretical problems for the realization of distributed robotic systems. In *Proceeding of the IEEE International Conference on Robotics and Automation*, pages 1914–1920 vol.3, 1991.

[19] E. Bonabeau and F. Cogne. Self-organization in social insects. *Trends in Ecology and Evolution*, 12:188–193, 1997.

[20] E. Bonabeau, M. Dorigo, and G. Theraulz. *Swarm Intelligence: From natural to artificial systems.* Oxford Univ. Press, 1999.

[21] E. Bonabeau, G. Theraulaz, , and J.-L. Deneubourg. Quantitative study of the fixed threshold model for the regulation of division of labor in insect societies. In *Proceedings of the Royal Society of London, Series B-Biological Sciences*, volume 263, pages 1565–1569, 1996.

[22] A. Braun, S. Musse, L. Oliveira, and B. Borman. Modeling individual behaviors in crowd simulation. In *IEEE International Conference on Computer Animation and Social Agents*, pages 143–148, 2003.

[23] M. D. Breed, G. E. Robinson, and R. E. Page. Division of labor during honey bee colony defense. *Behavioral Ecology and Sociobiology*, 27:395–401, 1990.

[24] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[25] R. A. Brooks. Intelligence without representation. *Artificial intelligence*, 47(1):139–159, 1991.

[26] J. L. Burton and N. R. Franks. The foraging ecology of the army ant eciton rapax : An ergonomic enigma? *Ecological Entomology*, 10:131–141, 1985.

[27] S. Camazine and J. Sneyd. A model of collective nectar source selection by honey bees: Self-organization through simple rules. *Journal of Theoretical Biology*, 149(4):547–571, 1991.

[28] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, March 1997.

[29] G. D. Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[30] L. Chaimowicz, M. F. M. Campos, and V. Kumar. Dynamic role assignment for cooperative robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[31] L. Chaimowicz, V. Kumar, and M. F. Campos. Framework for coordinating multiple robots in cooperative manipulation tasks. In G. T. McKee and P. S. Schenker, editors, *Proceedings SPIE Sensor Fusion and Decentralized Control in Robotic Systems IV*, volume 4571, pages 120–127, oct 2001.

[32] R. Cheng, M. Gen, and Y. Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms. *Computers and Industrial Engineering*, 30(4):983–997, 1996.

[33] R. J. Collins and D. R. Jefferson. AntFarm: Towards simulated evolution. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 579–601. Addison-Wesley, Redwood City, CA, 1992.

[34] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant system for job-shop scheduling. *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

[35] E. Şahin, T. Labella, V. Trianni, J.-L. Deneubourg, P. Rasse, D. Floreano, L. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo. SWARM-BOTS: Pattern formation in a swarm of self-assembling mobile robots. In A. El Kamel, K. Mellouli, and P. Borne, editors, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia, Oct. 6-9, 2002. Piscataway, NJ: IEEE Press.

[36] M. Dell'Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):231–252, 1993.

[37] J.-L. Deneubourg, S. Goss, N. R. Franks, and J. M. Pasteels. The blind leading the blind: Modelling chemically mediated army ant raid patterns. *Insect Behavior*, 2:719–725, 1989.

[38] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *31st Hawaii International Conference on System Science*, Big Island of Hawaii, 1998. IEEE.

[39] M. Dorigo, G. D. Caro, and L. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[40] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[41] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[42] R. Eberhart and Y. Shi. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, page 1950, 1999.

[43] R. Eberhart and Y. Shi. Particle swarm optimization: Developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 81–86, Seoul, South Korea, 2001.

[44] G. W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation.* MIT Press, Cambridge, MA,, 1998.

[45] N. R. Franks. Teams in social insects: group retrieval of prey by army ants (eciton burchelli, hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 18(6):425–429, 1986.

[46] J. Fredslund and M. J. Matarić. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[47] M. Garey, D. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[48] E. Gat. On three-layer architectues. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 195–210. AAAI Press/The MIT Press, Menlo Park, CA, 1998.

[49] P. Gaudiano, B. Shargel, and E. Bonabeau. Control of uav swarms: What the bugs can teach us. In *2nd American Institute of Aeronautics and Astronautics "Unmanned Unlimited" Conf. and Workshop and Exhibit*, 2003.

[50] T. Gonzalez and S. Sahni. Flowshop and jobshop schedules: complexity and approximation. *Operations Research*, 26(36):52, 1978.

[51] G. Gowtham and K. Kumar. Simulation of multi UAV flight formation. In *Digital Avionics Systems Conference*, volume 2, page 6, 2005.

[52] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. Autonomous self-assembly in a swarm-bot. In K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte, editors, *Proc. of the 3rd Int. Symp. on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 314–322. Springer, Berlin, Germany, 2006.

[53] R. Groß and M. Dorigo. Cooperative transport of objects of different shapes and sizes. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 107–118. Springer Verlag, Berlin, Germany, 2004.

[54] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067–1141, Dec 2001.

[55] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000.

[56] J. Hodgins and D. Brogan. Robot herds: Group behaviors for systems with significant dynamics. In *Proceedings of Artificial Life IV*, pages 319–324, 1994.

[57] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo. Self-organizing collective robots with morphogenesis in a verticalplane. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 2858–2863, Leuven, Belgium, 1998.

[58] S. Kazadi, A. Abdul-Khaliq, and R. Goodman. On the convergence of puck clustering systems. *Robotics and Autonomous Systems*, 38(2):93–117, 2002.

[59] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE Int'l. Conf. on Neural Networks*, volume 4, pages 1942–1948, Piscataway, NJ, 1995.

[60] J. Kennedy and R. Eberhart. *Swarm Intelligence.* Morgan Kaufmann, 2001.

[61] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[62] R. C. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, Volume 30(1,2):85–101, 2000. ISSN: 0921-8890.

[63] G. Lapizco-Encinas and J. Reggia. Diagnostic problem solving using swarm intelligence. In *IEEE Swarm Intelligence Symposium*, pages 365–372, 2005.

[64] E. Lawler, J. Lenstra, A. R. Nan, and D. Shmoys. *Logistics of Production and Inventory, Handbooks in OR & MS 4*, chapter Sequencing and scheduling: algorithms and complexity, pages 445–522. Elsevier Science, Amsterdam, The Netherlands, 1993.

[65] M. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, December 1995.

[66] M. Matarić, M. Nilsson, and K. Simsarian. Cooperative multirobot box pushing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 556–561, 1995.

[67] M. J. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, 1995.

[68] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics.* Springer, 2004.

[69] N. Miyata, J. Ota, T. Arai, and H. Asama. Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. *IEEE Transactions on Robotics and Automation*, 18(5):769–780, 2002.

[70] L. Parker. Designing control laws for cooperative agent teams. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 582–587, 1993.

[71] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[72] H. Parunak and S. Brueckner. Swarming coordination of multiple UAV's for collaborative sensing. In *Proceedings Second American Institute of Aeronautics and Astronautics "Unmanned Unlimited" Systems*, 2003.

[73] H. Parunak, S. Brueckner, and J. Odell. Swarming pattern detection in sensor and robot networks. In *American Nuclear Society (ANS) Topical Meeting on Robotics and Remote Systems*, 2004.

[74] H. Parunak, M. Purcell, and R. O'Connell. Digital pheromones for autonomous coordination of swarming UAV's. In *Proceedings 1st UAV Conference*, 2002.

[75] D. Payton, R. Estkowski, and M. Howard. Compound behaviors in pheromone robotics. *Robotics and Autonomous Systems*, 44(3-4):229–240, September 2003.

[76] B. S. Pimentel, G. A. S. Pereira, and M. F. M. Campos. On the development of cooperative behavior-based mobile manipulators. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 234–239, Bologna, Italy, July 2002.

[77] M. Quinn, L. Smith, G. Mayley, and P. Husban. Evolving teamwork and role allocation with real robots. In *Proceedings of the 8th International Conference on Artificial Life*, pages 302–311, 2002.

[78] V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith. *Modern heuristic search methods.* John Wiley & Sons, 1996.

[79] W. T. Reeves. Particle systems: A technique for modeling a class of fuzzy objects. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 359–375, New York, NY, USA, 1983. ACM Press.

[80] M. Resnick. *Turtles, Termites and Traffic Jams.* MIT Press, 1994.

[81] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[82] C. Reynolds. Steering behaviors for autonomous characters. In *Proc. Game Developers Conference*, pages 763–782. CMP Game Media Group, 1999.

[83] C. Reynolds. Interaction with groups of autonomous characters. In *Proc. Game Developers Conference*, pages 449–460. CMP Game Media Group, 2000.

[84] C. Reynolds. Big fast crowds on PS3. In *Proceedings of Sandbox (an ACM Video Games Symposium)*, Boston, Massachusetts, July 2006.

[85] G. E. Robinson. Regulation of division of labor in insect societies. *Annual Review of Entomology*, 37(1):637–665, 1992.

[86] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. A. Brueckner. Performance of digital pheromones for swarming vehicle control. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 903–910, New York, NY, USA, 2005. ACM Press.

[87] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-like agents for load balancing in telecommunications networks. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 209–216, New York, NY, USA, 1997. ACM Press.

[88] R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.

[89] T. D. Seeley. The tremble dance of the honey bee: message and meanings. *Behavioral Ecology and Sociobiology*, 31(6):375–383, 1992.

[90] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, 1948.

[91] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong. Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, 17(1):93–105, November 2004.

[92] K. Sigurd and J. How. UAV trajectory design using total field collision avoidance. In *American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[93] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence*, pages 832–838. Morgan Kaufmann, 1997.

[94] J. H. Sudd. The transport of prey by an ant pheidole crassinoda. *Behavior*, 15:295–308, 1960.

[95] J. H. Sudd. The transport of prey by ant. *Behavior*, 25:234–271, 1965.

[96] T. Sugar and V. Kumar. Control and coordination of multiple mobile robots in manipulation and material handling tasks. In *The Sixth International Symposium on Experimental Robotics VI*, pages 15–24. Springer-Verlag, 2000.

[97] D. Thalmann, S. R. Musse, and F. Garat. Guiding and interacting with virtual crowds in real-time. In *Proceedings of Eurographics Workshop on Animation and Simulation*, pages 23–34, Milan, Italy, 1999.

[98] V. Trianni. Evolution of coordinated motion behaviors in a group of self-assembled robots. Technical Report TR/IRIDIA/2003-25, IRIDIA - Université Libre de Bruxelles, Belgium, May 2003. DEA Thesis.

[99] V. Trianni, S. Nolfi, and M. Dorigo. Cooperative hole avoidance in a *swarm-bot*. *Robotics and Autonomous Systems*, 54(2):97–103, 2006.

[100] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Computer Graphics 28 Annual Conference Series*, pages 43–50, 1994.

[101] R. Vaessens. *Generalized Job Shop Scheduling: Complexity and Local Search*. PhD thesis, Eindhoven University of Technology, 1995.

[102] D. Vail and M. M. Veloso. Multi-robot dynamic role assignment and coordination through shared potential fields. In A. Schultz, L. Parkera, , and F. Schneider, editors, *Multi-Robot Systems*, pages 87–98. Kluwer, 2003.

[103] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987.

[104] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992.

[105] V. Verth, V. Brueggemann, J. Owen, and P. McMurry. Formation-based pathfinding with real-world vehicles. In *Proceedings Game Developers Conference*, 2000.

[106] K. von Frisch. Decoding the language of the bee. *Science*, 185:663–668, 1974.

[107] M. M. Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos*. Penguin Books Ltd., 1994.

[108] B. B. Werger and M. J. Mataric. From insect to Internet: Situated control for networked robot teams. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):173–197, 2001.

[109] A. Yamashita, T. Arai, J. Ota, and H. Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, 19(2):223–237, 2003.