Swarm Engineering

Thesis by:

S. Kazadi

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

California Institute of Technology

Pasadena, California

(Submitted and Defended 3 May 2000)

(Accepted 31 May 2000)

Acknowledgements and Dedication

This work was completed under the auspices of Dr. Rodney Goodman, to whom I am grateful for his guidance and support. The work was an attempt to begin to correct what I believe is a glaring hole in the science of autonomous robotics. Dr. Goodman provided a theoretical sounding board, many interesting problems, and a stable research environment. The experience gained outside of that illuminated in this work is every bit as valuable as that included. I am very grateful for the opportunity that I have been given.

I have also worked with or around many people from Dr. Nate Lewis's group. The perspective I found from these chemists was different from that I learned as a physics undergrad or from the engineers within Dr. Goodman's lab. The discussions I had with them were always welcome and insightful.

I owe a debt of gratitude to Dr. Maja Mataric. Her attention to my thesis went well beyond the call of duty of a thesis committee member. It was her suggestions that led to a significant improvement in the statement of my results and in the manner in which the Thesis was written. She helped me to complete this work, producing finally a document that is neither overstated nor contravertial, but rather states the work I've done, and what I think it's scope is. Her influence has affected my writing style in what I hope is a positive and permanent way.

I would like to dedicate this work to my wife and my parents. It was my parents who gave me the opportunities to learn and explore what would become my passion. Their sacrifices over the years will never be forgotten. It was my wife, my companion, who helped me struggle in these last years, helping me to find my strength when I lost it, my hope if it began to disappear, and my determination when I wavered.

Abstract

Swarm engineering is the natural evolution of the use of swarm-based techniques in the accomplishment of high level tasks using a number of simple robots. In this approach, one seeks not to generate a class of behaviors designed to accomplish a given global goal, as is the approach typically found in mainstream robotics. Once the class of behaviors has been understood and decided upon, specific behaviors designed to accomplish this goal may be generated that will complete the desired task without any concern about whether or not the final goal will actually be completed. As long as the generated behaviors satisfy a set of conditions generated in the initial investigation, the desired goal will be accomplished.

This approach is investigated in terms of three specific practical problems. First we apply swarm engineering to plume tracking, utilizing both real and simulated experiments. We initially decide whether or not this is a problem that swarm engineering should be applied to at all, A careful investigation of two plume-tracking algorithms yields the weakness of single-robot plume tracking – the tracking of low density plumes. Application of swarm engineering to this sub-problem yields performance that is capable of tracking plumes that are significantly more diffuse than those which can be tracked by single plume trackers.

The second problem is position-independent clustering, in which a cluster of objects is constructed from many initial clusters, without predetermination of the final cluster's location. Although we have completed a robotic instantiation of this type of system, the bulk of our work is theoretical and verification with simulation. A minimal condition is derived which guarantees this final global outcome, in both an unrealistic case and a case that is more realistic for real robots. Methods of generating efficiency improvements are derived. The application of this formalism to real robots is discussed. Finally, conditions are derived and demonstrated leading to stable arrangements of multiple clusters, a precursor of distributed construction.

The third and last problem is the traveling salesman problem (TSP). Marco Dorigo (see, for instance, Dorigo 1996) generated much interest in the use of "ants" on the problem. We theoretically demonstrate that Dorigo's ants satisfy a swarm engineering condition generated for this problem. However, the result of Dorigo's work is also demonstrably fragile under random fluctuations. A second more robust ant-based method is generated and tested on a small number of standard test problems taken from TSPLIB, a source of many well-known TSP instantiations. In all cases, the new algorithm is capable of reliably finding the minimum distance path.

This thesis makes a number of contributions to the literature. First, we study two systems that take advantage of the embodiment and interaction dynamics of the robot to accomplish the single robot goal. We then extend this work to answer two questions:

1. Is it possible to apply swarm engineering to plume tracking?
2. How can one apply swarm engineering to plume tracking?

We demonstrate that is indeed possible to extend one of the two previous plume tracking systems to a swarm. We demonstrate that the swarm is capable of successfully tracking plumes that are significantly more diffuse than those capable of being tracked by individual agents.

Next, we derive a formal condition which must be satisfied in order for a system made up of many clusters and robots to become a system of one cluster and many robots, if the robots do not have global information. We extend this work by deriving ways of generating more efficient paradigms, and demonstrate how to use the theory in the design of individual robots. Finally, we modify our theory to include systems generating multiple clusters, and demonstrate how to generate clusters of predefined relative sizes.

Our third contribution is a formalism for ant-based TSP algorithms. This allows us to understand why Dorigo's algorithms are relatively unstable. We create a completely new system of traveling agents that also solve TSP by having the ability to "die along the way" and to self replicate in a specific fashion.

Perhaps the most important contribution here is an alternative approach to the design of swarm systems. This allows us to explore the three problems of interest here by first determining a condition that allows the completion of the task, and then allows the generation of behaviors satisfying the minimal condition. This view to system design has the advantage of allowing the design of systems that will generate the desired behavior, concommittantly reducing the danger of generating unwanted emergent behavior.

Table of Contents

Introduction

As a child, I often found myself outside watching thousands of busy ants marching tirelessly to and fro in a seemingly endless struggle to find and exploit food sources. I would often present gifts, morsels of food that would occupy them for hours, until they disappeared completely. Other times, I would try to impede their progress by placing an object in their paths. Most times, the ants would ignore my efforts, smoothly finding their way *en masse* around the object. Individuals would occasionally get lost, but the group was quick to work out the problem, forming a stable flow as if ordered by their queen several meters away in the nest. My efforts were in vain, and the group ignored me altogether. Years later, this interest would be rekindled as I became familiar with work done in trying to understand ants and ant-based systems.

As it turns out, a number of insect species form swarms of thousands of individual animals, which form efficient societies capable of a number of interesting and startling behaviors (Camazine 1990, Dejean 1986, Deneubourg 1983, Fittkau 1973, Franks 1992, Gordon 1999, Holldobler 1990, Holldobler 1994, Karsai 1993, and Theraulaz et al. 1998). Among these are the ability to build structures with great complexity in the absence of any intelligence or planner, the ability to cooperate without any coordination, and the ability to allocate and reallocate resources automatically, without the need for an understanding of what resources are lacking in specific places. The ability of swarms to carry out tasks that would at first glance seem to require a great deal of planning and intelligence makes them interesting as possible design paradigms for swarm-based implementation of any of these types of algorithms or processes. Understanding biological swarms may provide possible implementations for behaviors that are potentially extremely simple, and remarkably robust.

Currently, the field of swarm based robotics is primarily exploratory, although a few practical applications have been built which extend or surpass the capabilities of current systems (Schoonderwoerd and Holland 1997, Dorigo et al. 1996). Very few tasks of practical interest have yet been carried out using the swarm-based methodology. However, many of the capabilities of agents in the natural world are staggering (Bonabeau 1991, Holldobler 1990, 1994) and it is interesting to understand how these occur. Once these are understood, it is possible that their extension to real engineering problems will allow the solution to problems for which the complexity of the solution is a serious detriment to its construction.

Already people are making strides in swarm-based research. The ability to build complex systems or systems with capabilities greater than those of any individual element is being explored in a number of different tasks, ranging from box pushing (Kube and Bonabeau, 2000) to distributed search (Gage, 1995) to distributed routing systems (Schoonderwoerd and Holland, 1997). Along with the raw extension of the cognitive capability of a given agent, and the development of intelligent behaviors, swarms have the possibility of extending the range of sensitivity of a given sensor modality, as is done in the first Part of this thesis. Though we are not aware of any further examples of this in the literature, the potential for future work is great. Moreover, work in this vein may help to motivate and explain the tendency for groups of animals to carry out what might seem to be tasks that can be accomplished by individual agents, but are typically done by groups, as in the long-range migration of birds during the Fall. The precision required to accomplish these tasks may not be available in individual agents, requiring the employment of groups.

Of course, engineering is goal-driven, and so techniques found in nature are not always readily applied to engineering. For instance, ant construction methods, though remarkable, cannot be applied to the construction of a building, as our goal is not an ant-style nest built into the ground. The goal must somehow be folded into the task development. The question for the engineer to answer is how one might build a simple robot or agent (for instance, for computational systems) which may be expanded into a swarm, the aggregate behavior of which will solve a given global task. One might appropriately ask questions relating to both a general condition for the accomplishment of the global task, and the tradeoff between the cost of the improvement of some measure of the task's completion quality and the level of improvement.

Much of the literature aimed at creating swarms of active agents has been aimed at understanding the global repercussions of a set of local rules. Indeed, the global behavior of systems created by specifying local rules are very difficult to predict, and the discovery of the behavior sometimes requires observation of an active system rather than the solution of a set of equations. The behaviors thus provided are more often than not *ad hoc* and their details are not well understood in the sense that one might generalize what has been learned to novel tasks. We are of the mind, however, that the goal of every engineer must be twofold: creation of a working model of a given design specification, and generalization of the techniques used in such a solution to other possibly related problems. It is in this second point that we address this current work.

We report in this Thesis work based on a general approach to developing a set of physical requirements and control algorithms for the completion of global tasks in a swarm-based architecture. This approach consists initially of the generation of a set of requirements for the behavior of the swarm which will guarantee that the task is completed correctly. Once this set of requirements is generated, by means that need not by analytic, local behaviors may be created which will globally satisfy this the requirements set down. Illustrating how this local behavior will satisfy the global task requirements is task-dependent, but typically yields insight into how one might further improve performance of the swarm on the task. We give this approach the name *swarm engineering* to explicitly communicate our intention that this technique should be applicable to a wide range of distributed engineering problems.

Note that we use the terminology "swarm engineering" rather than more common terms such as "collective robotics". The use of this term is motivated by a need to emphasize that the techniques developed here are intended to be general. The possible applicability includes robotics, which forms the bulk of our work, but may also include software swarms, as in Part 3 of this Thesis. In addition, one might apply these methods to social science, effectively engineering aspects of society, through methods similar to game theory. While the end goal is different in each case, the general techniques can be similar.

This Thesis is dedicated to the illustration of the swarm engineering approach to building swarms. We do so in several rather self-contained parts, and two chapters not contained in any part. Each part discusses a particular problem, and elucidates the design of novel systems based on this methodology. Part 1 discusses mobile plume tracking in autonomous robotic systems, both using single robots and using multiple robotic platforms. This is the most ad hoc part of the thesis, as no satisfactory theory currently addresses this problem. Part 2 discusses a distributed clustering task, in which a set of objects are clustered into a single group using a swarm of autonomous, identical, independent robots, bereft of any global knowledge. Part 3 describes the use of a computational swarm of autonomous agents in the solution of the traveling salesman problem, building theory for the work of Dorigo (Dorigo, 1996), and extending the model. Each Part consists of a self-contained review of the relevant specific topics. Chapter 1, not contained within any Part, is designed as a rather independent review of some of those topics not discussed in any of the Parts. Finally, Chapter 10, also not contained within any Part, is a concluding chapter intended to summarize the main points made in the Thesis.

CHAPTER 1

Swarms, Natural and Fabricated

In this chapter, we define swarms of agents in a way that is both general and explicit. We make every attempt to encompass all current work in swarms using these definitions. We illustrate these definitions with examples chosen from the natural world, human society, and the artificial world. Once this is completed, we define the swarm engineering approach, motivating its design, and providing an example of how this approach may be applied.

1.1. Definition of Swarms

The study of agent-based systems occasionally begins with a definition of the term *agent* (Maes 1994, Beer 1995). The concrete definition of such a basic element of what have become known as swarm-based systems would seem to be straightforward, but the precise definition has not yet been universally agreed upon.

Maes (Maes, 1994) describes an *agent* as an autonomous, situated entity carrying out a set of goals in the environment in which it is situated. In this way, Maes characterizes the agent as first having a goal or set of goals to be accomplished. The use of the term "goal" indicates an understanding of the desired end result of the behavior of the agent in its environment. Quite generally, this level of understanding of the behavior of an agent cannot be thought of as part of the design, but rather imposed by the observer of the system, which may not be the agent in the first place. Yet, some biological systems which might seem to exhibit goal-driven behavior are simply reactive systems driven by nothing more than physics and chemistry. For instance, a virus can have goals attributed to it, but it does not *pursue* them. Rather, it behaves according to a well-defined set of rules, yet still manages to survive. One might argue that the 'goal' of the virus is its replication, but its dynamics are simple reactions to local conditions, and are completely dictated by the chemistry. Except in abstraction, it is unclear how one might attribute a goal to this behavior.

Beer (Beer, 1995) proposes a formalism which seeks to model agents as parametric *non-autonomous* systems, whose dynamics are parametrized by the interaction with another system. The other system is the environment in which the agent finds itself. In this view, only closed systems whose evolution is determined entirely by consistent equations of motion are truly autonomous, and so no agent can ever really be autonomous, as it is never independent of its environment. This view requires the agent to be inextricably tied to its environment, unable to be viewed in any other context. Many aspects of agents and their characteristics are well treated by this way of thinking. For instance, the characteristics of ants and the substances they secrete are carefully built around the physics of the ants' environments. Were it not for the detailed properties of the ants' environments, the ants would not be able to depend on these characteristics, and the species would be forced to have developed differently. Thus, the ant is a reflection of the system in which it is found, and it may not be viewed in totality without considering the larger system.

We take a somewhat complementary tack to both of these approaches. We relax the requirement that the agent must be following some explicit goals, and rather defer to a more general formulation. In our formulation, those agents not capable of independently sustaining themselves may be viewed as agents. This includes agents in a symbiotic (and so codependant) relationship.

We define an *agent* to be any discrete component of a system which satisfies the following properties:

1. An agent must be situated within the system. That is, the agent must be affected by the physical properties of the system it is in, restricted by its design, and may not be capable of modifying the fundamental structure of the system.
2. An agent must have the ability to act within the system. This means that it must have the ability to change at least one external degree of freedom based on some locally decided upon actions.

The first condition serves to limit the agent to an actor in the play unfolding in the system. This requirement neither restricts the agent to a specific locality nor assumes that it cannot accumulate information about the system through interactions with other agents, and control other agents in some way.

The second condition would seem to be an extension of the first condition. Agents cannot behave in a system in which they do not find themselves. That is, the system in which they are situated must be *closed* under the actions of the agent. In this way, an agent is different from an observer who may be limited by a given system, but may not change it. This does not presuppose that an agent works in a location or by changing a degree of freedom local to itself – it may communicate intentions to other agents.

Note that a virus is not identified as an agent when it is outside of a very specific environment, as it has no control over any degrees of freedom. In environments where it is incapable of independent animation, a virus would be viewed as a part of the system, but not an agent. When it transitions to a system in which it can exert independent control, it becomes an entity that can be viewed as an agent. This may or may not include the interior of a cell, depending on whether the virus actually acts on the cell, or instead is acted upon by parts of the cell.

We define communication to be *any action of one agent which influences the subsequent behavior of one or more other agents in the system, directly or indirectly.* In an information-theoretic way, it may be defined as the reduction in entropy of a given agent's behavior as a result of the receipt of some communication signal (Shannon, 1953). This includes two types of communication. In *direct* communication, the action of the first agent is capable of being sensed directly. This is as in a system in which agents create sounds and respond to sounds produced by other agents. *Stigmergic* communication (Beckers 1994) is communication through the environment, in which a change in the environment which is not made explicitly for communication is made, sensed by another robot, and in turn produces a different behavior in the sensing robot. Note that the way in which the stigmergic communication occurs may also be the desired goal of the agent. Moreover, an agent can communicate with itself, stigmergically.

A *swarm* is a set of two or more independent agents all acting in a common environment in a coherent fashion, generating an emergent behavior. An *emergent behavior* is one in which the behavior of the swarm is a result of the interactions of the swarm with its environment or members, but not a result of direct design. Emergent behavior requires inter-agent communication, which need not be direct.

Note that a swarm is *not* a group of identical robots which simply behave in the same environment by a similar or related set of rules of behavior. Such groups of agents may appear to create swarm-like behavior, but lack any deliberate communication that is the hallmark of a swarm. This distinction in behavioral traits also has a related distinction in capabilities, as we shall see in the first Part of this Thesis. Such groups of agents are hereafter referred to as *pseudoswarms*, in the spirit of Owen Holland and Chris Melhuish (Holland and Melhuish, 1999) to denote explicitly that these systems are different from swarms.

We pause briefly to address the question of the minimal size of a swarm. When does one move from a small collection of robots to a swarm? This would seem to require that a lower bound on the number of robots be created which leads to the conclusion that a swarm has been made. The most important criterion for a swarm is that it consists of agents whose actions affect those of other agents in the same system. The use of stigmergic communication allows this to occur even with single agents. This would seem to imply that single agents are capable of acting with the dynamics one might find in a swarm. However, in this work, we explicitly define swarms as requiring multiple members. The intriguing question which seems to come from this is whether these should be viewed simply as swarms. It may be that there is a larger class of dynamic agent-based system, to which swarms belong, which behave in the same general way as swarms do. Though we do not consider this question in more detail in this work, we consider it a fascinating possible future research direction.

Of course, not every interacting pair of agents constitutes a swarm. A pair of nesting birds exhibit communication to be sure, but they do not exhibit by themselves emergent behavior (or at least the author is not aware of any examples of such behavior). On the other hand, groups of migrating birds might exhibit the capability of navigating over large distances much more accurately than any one bird would be capable of doing on its own. This is an emergent property of the swarm. It is not currently known what the minimum number of birds required to create this increased capability is, and the author is not aware of any studies of this effect.

1.2. Examples of Swarms

Many familiar things found in the natural world are swarms. The swarms are, of course, made up of many individual animals, but in many cases may be thought of as life forms in and of themselves. Groups of ants, particularly those inhabiting the same colonies are swarms. Similar organization may be observed with groups of bees living in hives, and termites living in nests. Each of the agents in these examples are independent enough to act alone, but ineffective without the combined power of the swarm. The dynamics of the swarms found in the natural world are both interesting and awe-inspiring. Indeed, they have been characterized as *superorganisms*, the dynamics of which are the interesting measurables (Holldobler 1990, 1994).

Indeed, the success of swarms in the natural world is nothing less than staggering. Since their rather late introduction to the world, somewhere in the Cretaceous period (Holldobler, 1994), swarms of insects have come to dominate every habitat where comparative studies have been made (Fittkau 1973, Dejean1986), making up more than 75% of the biomass of the land-based insects in the Amazon rain forest and in the forests and savannas of the Democratic Republic of the Congo (formerly Zaire). It is not clear from studies done to date whether this success is the result of advantages of swarms over their individual single-living brethren. More extensive study is required to obtain an understanding of both the biology and possible implementation of similar methodology in artificial systems.

The natural world is not the only source of swarms, however. In human culture, swarms abound, varying in size, complexity, and levels of communication. Perhaps the two most visible swarms are those made up of deliberately information sharing agents, such as investors in the stock market, or members of the various scientific communities. In much the same way as the dynamics of the superorganism of an ant hill or termite nest has interesting dynamics and complex actions contributed to by the individual agents, so too do the larger superorganisms made up of groups of interacting humans.

The stock market is an example of such a dynamic system which evolves and grows, albeit in a constrained manner, subject to the contributions of thousands of individual agents whose behaviors collectively determine the course of the market, but who individually are rather less than the whole, both in dynamics and capability.

Traffic swarms represent a second class of common human swarms in which agents interact using a small amount of information, the nature of which is imprecise and approximate, and whose actions are quite constrained (Helbing and Schreckenberg 1999, Treiber and Helbing 1999, 1999a, Resnick 1994). These systems have a rich behavior including traveling traffic jams and stationary traffic jams, despite the fact that the agents in these swarms enter and exit the system continually.

Among the most well-developed swarms in human society are groups of technology producers and users. In much the same way that ants will become locked into a particular path after many minutes of exploration of different paths, so too does technology become locked into a specific standard as a result of the interactions of the technology producers and consumers. One example is the right-handed clock, which could as easily be implemented as a left-handed clock (and was at one time), but which came to dominate as a result of a feedback effect. This feedback effect is not explicitly designed in any of human behaviors, yet is an emergent result of these behaviors, the hallmark of a swarm. Subsets of this global swarm exist throughout society, and it is the belief of the author that this is indeed what defines society in the first place.

However, it is not to imply that every human group interaction has swarm-like tendencies. The author is not aware of emergent properties of travelers going through airports, although there is indeed communication and interference. In such a setting, the travelers are constrained strongly, and have only a few behavioral options available. These options do not allow for the development of emergence, as would be required to be considered a swarm. Such travelers are an example of a pseudoswarm.

1.3. Creation of Robotic Swarms

The creation of artificial swarms has become an interesting topic of research in the last fifteen years. This research finds its roots in the seminal work of Rodney Brooks (Brooks 1986) which advocated the creation of simple robots with simple behaviors in the place of robots of complex abstract models and interpreters for intelligent behavior. The method is called *behavior-based control* and has become widely accepted, due to the simplicity of the approach, the flexibility and potential for generalization to learning and and the capability of researchers to create swarms of simple behavior based robots capable of carrying out interesting behaviors. Several researchers have proposed interesting methods of creating control algorithms for such simple robots, including learning and evolutionary algorithms (Harvey 1994, Mataric and Cliff 1996), and work continues on both of these general methods.

Artificial swarm design has thus far been largely along two different lines of design. The first of these is the practical approach (Kelly 1996, 1998, 1998a, Mataric 1995, 1995a, 1997a), in which a particular task is identified and a group of robots is built with the task in mind. This work typically contains multiple iterations of design and redesign of the physical robot and/or its controller to complete the task (Goldberg and Mataric, 1997). The second of these is the theoretical approach in which a task is examined theoretically, and a condition is derived which will yield the global goal. The design of the robots is then an exercise in obtaining robots whose real dynamics matches the desired dynamics, which can take one iteration. The main difference between these is that in the first approach, an understanding of the underlying system dynamics is generated through the building and improvement of various versions of the system, while the second attempts to generate an understanding of these dynamics, and subsequently build robotic systems based on this understanding. In this way, the second approach is more similar to traditional methods in robotics, in which the system dynamics are worked out before the generation of a robot. The use of this approach in swarm robotics would seem to provide validation of the swarm-based approach to system design, indicating that these methods are not unrelated to work done on individual robots carrying out sophisticated tasks with high precision. Rather, they may be seen as an extension of those methods to tasks requiring a smaller degree of individual precision.

Many studies have been undertaken using the practical approach to swarm construction. Among these are studies investigating navigation and exploration tasks, rudimentary construction tasks, task allocation studies, and communication studies.

Cohen (Cohen 1996) reports on a mapping and navigation task in which a swarm of robots spreads itself in a maze, and collectively maps the terrain. The map is detailed enough to allow an agent to travel from a point in the map to any other point in the map in a minimum amount of time, while using minimal individual processing, and taking advantage of an elegant communication scheme.

Mataric (Mataric 1995a, 1992a) describes a method for designing simple orthogonal local behaviors which may be used to create diverse global group-level behaviors. The design is a process of simultaneously satisfying top-down (task-driven) and bottom-up (robot and environment-driven) constraints of the problem. This work may be seen as complementary to ours. Whereas our goal is the generation of conditions which when satisfied will generate the desired behavior, this work provides a set of techniques which may be used to satisfy the given condition.

Gage (Gage 1995) details the design of a multiple robot system for mine detection. This work focuses on illustrating the benefits in time and correctness of coordinated and communicating multiple robot detection schemes. The use of communication and coordination is found to be a serious advantage over the lack of it, in terms both of detection speed, correctness, and cost/success ratio.

Dorigo and Gamardella (Dorigo 1997) describe the creation of a swarm of artificial virtual agents traveling on a graph made up of nodes and links. This swarm is designed in such a way as to solve the traveling salesman problem (TSP), which we discuss later in this Thesis. The design is based on the behavior of ants, and is systematically improved to produce a rather computationally quick method of solving the problem.

These four examples of swarm behavior illustrate the power of the method in a concrete way. We discuss further examples of this later, in the separate Parts of this Thesis. The generation of real working models represents an important and necessary step in the design and verification of a new technology. However, the main weakness in these studies is their ability to be generalized beyond closely related tasks, such as those investigated by Dorigo and colleagues as extensions of the ant-based TSP work (Dorigo 1999). It is not clear how these studies might be used to design new, very different swarms without significant additional effort.

In contrast, a method based on artificial physics and derived from artificial life is gaining increasing support. This method purports to create a dynamic set of equations capable of being followed by given agents in a system which produce a desired behavior of the swarm. The advantage of such an approach is that these equations may be completely solved, yielding the stationary states of the system, and the modes of evolution of the system. As long as the robots are capable of following the dynamic equations of motion, the method will provide the desired behavior (Spears 1999).

Spears and Gordon (Spears 1999) report work on an artificial physics-based control algorithm intended to be employed on MAVs [1]. The central design condition is the creation of a set of dynamic equations based on an attractive/repulsive force. This force is of order r^{-2} and reverses sign at a given distance between two MAVs, switching from an attractive force to a repulsive force. A modification of the system in which a sorting behavior is employed allows the system to reliably create perfect hexagonal and square lattices. This is the desired global outcome, and is accomplished using only local actuators and sensors, making the possibility of porting the algorithm to a MAV realistic. This work has only been carried out in simulation thus far; future work may include the use of these methods.

1.4. Swarm Engineering

This Thesis is complementary to both of the previous approaches. It is important to be able to both create working robotic platforms and simulations. However, it would seem to be desirable to be able to express the work in a format that allows generalization and ease of implementation. Thus, we propose the formal field of *swarm engineering* and use the remainder of this Thesis to illustrate it. We define swarm engineering as a formal two-step process by that one creates a swarm of agents which complete a predefined task.

Swarm engineering has two formal steps. The first step involves the generation of a *swarm condition*. This is a condition which, when satisfied, will guide the generation of a swarm of agents which is capable of carrying out the desired action or actions. No specific method has yet been formulated for this step, and it is largely problem-dependent. Many methods may be employed to carry out this step including formal mathematical analysis, a generation of understanding of the problem domain through simulation or fabrication of working models, or any other general method. What must be understood in this step is the behavior of the swarm in the problem domain. For instance, as we shall see in Part 1, transfer of information about a plume should occur only in the upstream direction, rather than the downstream direction, as the latter may confuse the entire swarm.

This first step is extremely demanding in its requirements, however. It requires that the problem be stated in such a way as to permit the creation of swarm-based actions. This means that the statement of the problem must be amenable to being satisfied by distributed agents, rather than a single coordinated statement.

[1] Micro-air vehicles

EXAMPLE 1.4.1. Position-independent puck clustering is the process whereby an initial set of pucks are clustered into a single cluster, whose position is not specified ahead of time. This problem is simply stated, yet as we shall now illustrate, the statement will often affect the proposed solutions, making this expression of the problem insufficient.

Viewing the puck clustering problem as the generation of behaviors that bring the pucks from smaller clusters to larger clusters implies that the robot should in some way have some global knowledge. The robot should understand where the large clusters are, and where the small clusters are, and behave in some way as to bring the pucks from the small clusters to the large cluster. However, this articulation of the problem does nothing to generate a swarm behavior, deferring to the generation of a much more high level behavior.

We choose to view puck clustering as a behavior that causes more pucks to enter than to leave large clusters, and more pucks to leave than to enter smaller clusters. This expression leads to a general microscopic condition of each of the swarm elements, which leads to the global behavior.

In general, care must be taken to avoid generating expressions of the problem which focus on the macroscopic goal. Rather, the expression should be centered around the microscopic goal, which may be satisfied by the agents of possibly differing designs.

The second step in swarm engineering involves the fabrication of a behavior or set of behaviors that satisfy the given swarm condition. If the system is understood well enough to generate a veritable swarm condition, the generation of a swarm behavior is often times straightforward. For instance, if the swarm condition is that robots must stay in close proximity, it is easy to generate sensors and actuators to satisfy this.

The goal is to create one of any of a set of behaviors which provably fulfills the swarm condition. Once this has been accomplished, questions about the efficiency of such approaches may be asked. However, at this point, the goal has been achieved and the new goal is to improve the performance of the algorithm, not to generate the initial behaviors.

Swarm engineering is similar in its goals to behavior-based group robotics (Mataric 1993). In both approaches, the goal is to generate a predefined group behavior. However, in behavior based group robotics, the approach is largely intuitive, and the process is complete when a single working prototype system has been generated (Goldberg and Mataric, 1997). The goal of swarm engineering is to produce a general condition or set of conditions which may be used to generate many different swarm designs any of which can complete the global goal. Once this condition or set of conditions is generated, specific behaviors which satisfy this (these) condition(s) may be generated, with absolute certainty that these will yield the desired global behavior.

Thus, swarm engineering consists of two related steps. The first step is to propose an expression of the problem which leads to a set of conditions on the individual agents which, when satisfied, will complete the task. The second step is to produce a behavior or set of behaviors for one or many robots which accomplishes these requirements. We shall investigate, in the next three Parts, three independent problems which may be tackled using swarm-based techniques. In the first Part, no general theory has been formulated, but a sufficient understanding of the domains in which a swarm will be advantageous as well as the condition for such a swarm to be effective is created. The second and third Parts generate a succinctly stated theoretical swarm condition. However, the swarm conditon of the third Part suffers from the effects of random changes. We illustrate a method of circumventing this difficulty.

Part 1

Plume Tracking

CHAPTER 2

Plume Tracking

2.1. General Considerations

A *plume* may be defined as a coherent structure made up of odorant particles carried by a fluid flow from a single source, in which the rate of speed of the fluid is much greater than that of the diffusion rate of the odorant. Plumes occur when an odor-producing source is placed within some fluid flow. Odorant particles are produced from the source and evolved into the fluid. Once the odor particles are in the fluid, they are moved by two forces: diffusion and fluid flow. The effect of the two varies with variations in Reynolds number. In fluids of relatively low Reynolds number, the flow tends to be very laminar, and the spread of odorant is typically governed by diffusion. As a result, the odorant tends to form a long cone in space, spreading out as it continues downwind. However, in fluids of relatively high Reynolds number, the fluid tends to be quite turbulent. In this case, the plume will be shredded by the turbulence, and will consist only of relatively small packets of odorant at various concentrations.

In natural systems, plume tracking is often critical to the survival of the individual, providing information about prey and about potential mates. Depending on the nature of the target, a number of different strategies have been developed over evolutionary time to find the source of the plume. As one might imagine, the differing evolutionary pressures and the requirements of differing environments and purposes have created differing strategies for plume tracking (Zanen and Carde 1996, Belanger and Willis 1996, Murlis, Elkinton, and Carde 1992). Several flies, for instance, are capable of tracking plumes by making use of an iterated scheme in which they stop moving, integrate the approximate average wind direction over many seconds, and then move off again in that direction. This seems to be a response to the lack of a need to quickly determine the target position, and a commensurate requirement to save energy.

When mating or searching for food, an insect typically has a large pressure to be the first to the source. This pressure leads to behaviors that consume a great deal of energy. These behaviors are often modulated by the instantaneous structure of the plume, including the frequency of delivery of packets of odorant, and perhaps their detailed structure. Changes in behavior include increased or decreased frequency of turns in the track and increased or decreased speed.

Much work has centered around the study of moth and fly plume tracking *[zanen1996]*. As opposed to the work on artificial systems, this work has been centered around the specific neural functions required to get the animal to behave as it does. In the remainder of this section, we will briefly review some of this work.

2.2. Autonomous Robotic Plume Trackers

A growing body of work is available which describes partially or fully autonomous robotic systems carrying out olfactory tracking. In general the state of the art seems to be partially autonomous robotic systems, and emphasis is usually placed on the search itself, not its termination. Indeed, there is no specific consensus on which details of the plume structure and dynamics are useful; this is a topic of current research. Thus, it is not surprising that there are a large number of strategies for plume tracking. One might argue that the number of strategies one might find or try is as varied as those found in nature, a very large number of strategies indeed!

2.2.1. Enabling Technologies. A number of olfactory technologies are available for use in plume tracking system (Russell 1994). In systems where speed of search is not an issue, one may use very slow olfactory technologies. In systems in which the speed of search is an issue, one finds that sensitivity is often traded off for speed, something that is not generally a problem in biological systems. A number of different technologies have been developed, ranging from purely artificial systems to remarkable fusions of real biological technology and artificial technology. We do not attempt to be complete, but rather to give a brief flavor of what has been built.

An interesting odor recognition system has been constructed by Russell et al. (Russell et al. 1994). In this system, a quartz crystal is used as a primary odor recognition device. This has the ability to change its intrinsic properties due to external conditions. Thus, when treated externally with a chemical which preferentially holds onto other volatile compounds, and then exposed to some or all of these compounds, the resonance frequency of the crystal will change, indicating the presence of at least one of the compounds of interest.

This system has a simple implementation and it is capable of being very accurately tuned due to the resonance properties of the crystal. The transduction of the signal obtained by the crystal is nearly instantaneous. Finally, depending on how the crystal is coated, the sensor may be either broadly or narrowly tuned.

Despite these advantages, however, the system has a serious disadvantage. In order to function properly, the sensor must be coated with a substance that attracts the odorant of interest. This coating, in order to be sensitive, needs to be able to hold enough of the odorant to change the resonance properties of the crystal. However, this can make the reaction of the sensor slow, and so unsuitable for use in an autonomous robotic system.

A second type of technology makes use of the properties of polymers (Lewis et al. 1996, White and Kauer 1997). Many polymers react to different odorants in a variety of interesting ways. These effects may manifest themselves as changes in the conductance, the density, the tensile strength, etc., of the polymer. These may be used in a number of interesting ways, from changing the index of refraction of the polymer to changing its size or conductive properties. Doping allows the last two to be coupled in non-conducting polymers.

Polymer-based sensors have a number of desirable properties. These are broadly tuned, and many have reproducible responses to low odorant concentrations. In many cases, these sensors are very quick to respond fully to an odorant, and can reach equilibrium in tenths of seconds. Finally, polymers may be applied to a number of surfaces, making the development of specialized sensors built into uncommon parts of the robot simple. Thus, polymers may be applied to parts of the robot commonly used to probe difficult-to-reach areas of the robot's world (Lewis et al. 1996).

The main limitations of polymer sensors also derive from their physical structure. Polymers are long chains of repeated units. When these chains break, the properties of the polymer change, including their sensitivity to some substances. This change is irreversible, giving the sensors a finite lifetime that may depend on the chemicals they encounter and the odorant concentrations they are exposed to. Polymer sensors seem to have a limited responsiveness. This responsiveness changes as the polymers encounter different environments. Indeed, the optimal environment for a polymer sensor may be one which is climate controlled, and isolated from the rest of the world. Some labs are attempting to develop such systems on mobile robots, but the work is in its infancy[1]. Some sensors require doping in order to produce conductive pathways. This adds complexity to the sensor, as the dopants themselves may have nontrivial properties (Lewis et al., 1996).

Some sensors derive from biological systems themselves. In some ways, they are extremely well suited sensors for olfactory search for a source of specific moth pheromones. These sensors are extremely sensitive, but are typically very specific to particular substances, such as pheromones.

[1]An example of two places in which this type of control goes on are the Lewis group in the Chemistry Department at the California Institute of Technology, and Cyrano Sciences, a company recently formed to develop polymer olfactory sensors.

In an elegant study, Kuwana and colleagues (Kuwana et al. 1995) removed the antennae from a male silk moth (*Bombyx mori*) and installed them on an autonomous mobile robot. These sensors were then used to carry out a variety of different algorithms for plume tracking. As the sensors were shown to have activity proportional to the concentration of the odorant (female pheromones), both edge-tracing algorithms, and gradient-based algorithms could be used. The beauty of these sensors is that they are already engineered for a variety of environments commonly encountered by the moth, and need no further engineering. One drawback is that they cannot be further engineered, and so their properties are unalterable, barring any genetic engineering or related work.

2.2.2. Tracking Strategies. Along with the different types of sensors come different types of search strategies. These strategies are widely varied, and each strategy is dependent on the particular olfactory technology available. As the dependence is quite strong, the group defies classification, except in two areas. The first is the set of strategies that use wind direction, and second is the set that doesn't. A possible subgroup is that which uses dynamic properties of the odorant packets sensed by the robot. However, only one group (Grasso et al. 1996) has reported the possible use of this. Airborne plumes typically move too quickly to make use of this data.

2.2.2.1. *Strategies Utilizing Wind Direction.* The most successful strategies in plume tracking are those that utilize wind direction information. This piece of information is so important, that the lack of it can be the main reason for failure. Despite the great need for this piece of information, the way in which it is obtained can vary widely. Moreover, the accuracy of this information is less of a critical issue than one might expect, and four directions can normally suffice for good performance, though only two need be used.

Russell et al. (Russell et al. 1994) report on the design of a robot which is capable of tracking a camphor plume upwind to its source. In these experiments, the wind direction is used in conjunction with gradient information. Assuming that the robot can reliably determine the center of the plume by following a gradient to the center of the plume, the robot's two behaviors include acquisition of an approximate center of the plume by determining the differential concentration at two crystal-based sensors spaced apart, and the following of the plume upwind, as determined by an on-board wind vane. An obstacle avoidance behavior was included by adding a sensitive bumper to the system and forcing the robot to make on-the-spot turns when it came into contact with an obstacle, effectively avoiding it. The robot was capable of reliably tracking the plume and avoiding obstacles from a meter away.

Ishida et al. (Ishida et al. 1996) describe a robotic system in which an 'active probe' is constructed. This 'active probe' uses a fan to draw air to the sensor, effectively handling the boundary layer problem[2], and the difficulty associated with a limited amount of available odorant. Wind direction information is obtained by rotating this sensor by 360° and recording the direction that has the largest reading. Later versions included stationary directional sensors fabricated from four sensors arranged in a cross, providing quicker but less accurate direction information. The new design could also be used to find three-dimensional plume information, though this information was unused.

2.2.2.2. *Strategies Without Wind Direction.* The simplest algorithm for determining the position of the plume source is one employed by Kuwana et al. (Kuwana et al. 1996) in which the robot remains completely within the plume as it is carrying out its tracking behavior. The robot is inactive when outside of the plume, and active when inside the plume. Its behavior is simple, forcing the robot to execute turns when it approaches the edge of a plume, and to walk straight when completely within the plume. This strategy has two main weaknesses. Firstly, the robot cannot find the plume, but rather must be initially placed within the plume in order for it to function at all. Secondly, half the time the robot will tend to go downstream, and half upstream. A bad turn produced by noisy or unexpected sensor data might cause the robot to move in the wrong direction down the plume.

A similar algorithm may be used to track saline plumes in water. Grasso et al. (Grasso et al. 1998) report the construction of a robotic 'lobster' designed to track saline plumes in water. The robot is endowed with sensors which can determine the local concentration of saline. Using this information, the robot climbs a concentration gradient from the plume source. The speed of each motor is controlled by the sensed concentration of the salt in the water. Turns can then be affected by changes in the concentration. The robot, which initially moves from within the plume, will also employ a backing behavior if the concentration becomes too low, indicating that the plume has been exited.

Kazadi (Kazadi 1998) produced a robot (Figure 2.1) capable of tracking a plume to its source by traversing the plume and executing turns upon loss of the plume. The robot utilized resistive polymer sensors (Lewis et al. 1996) and simple analog circuitry. Although these results were not published, a similar robot, which is thoroughly investigated in Chapter 3, was built with a very similar design.

[2]The *boundary layer problem* refers to the tendency of an odorant to pass over a sensor without directly interacting with it. This is a serious problem when dealing with odorants whose presence is intermittent or sporadic.

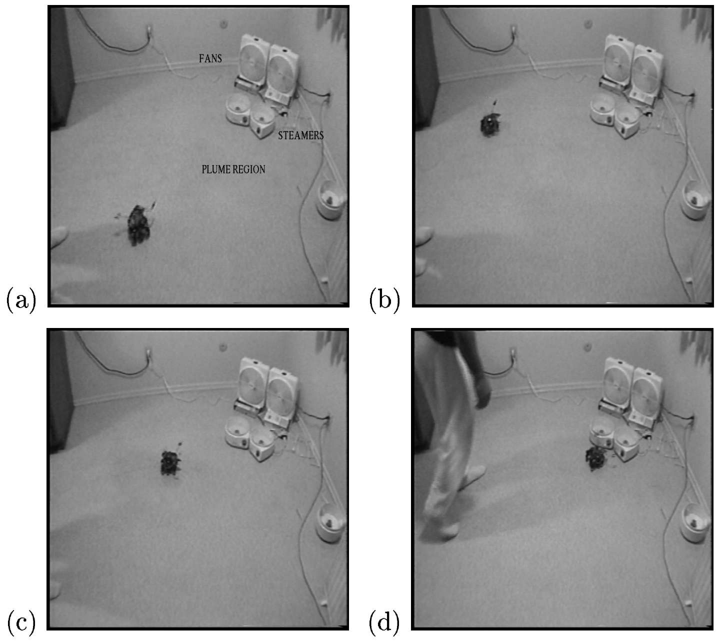


Figure 2.1: Our plume-tracking robot tracking an odor plume to its source (steamers).

In Chapter 4, we will examine a plume-tracking algorithm based on a spiral which does not use wind direction, and is robust to loss of source.

CHAPTER 3

A Case Study in Plume Tracking

One of the most important and ubiquitous behaviors in the animal world is plume tracking. This skill is arrived at in a variety of ways, and is required for a variety of reasons. Some animals require effective tracking capabilities in order to locate prey. Other animals find potential mates by taking advantage of pheromone plumes emitted by members of the opposite gender. Plume-tracking strategies are varied in the natural world. No single algorithm is universal. Many animals have 'hardcoded' tracking algorithms, while others have the capability to learn tracking and take advantage of new and more efficient methods of tracking. Many animals will change their tracking behavior in response to differing wind conditions. Though there is a great variation in plume-tracking strategies, constants across all algorithms are likely to be those characteristics of plume-tracking which will be required even in a minimal algorithm. Understanding these minimal requirements will allow for the generation of many varied plume-tracking algorithms, as long as these conditions are met.

In this work, our goal is to elucidate the potential use of swarm-based methods in plume-tracking. Our first step to this end is the generation of a set of minimal conditions required to accomplish plume tracking. Once this is accomplished, we will be in a better position to determine whether or not it is advantageous to use a swarm at all. Generation of a swarm, once the possibility of an advantage is discovered, will need to obey these minimal conditions, among possible other conditions.

In this Chapter, we examine a legged autonomous plume tracking robot. The robot, originally developed as a precursor to a true plume tracking robot, is capable of detecting when it is in a water vapor plume and walking upstream to the source of the water vapor. The capability of the robot is quite unexpected, as its development was not intended to allow the robot to accomplish this task. We describe our current understanding of why this occurred. We also briefly generalize the lessons obtained in this investigation to the swarm regime, giving two minimal conditions a swarm will need to obey in order to successfully track the plume upstream.

3.1. Basic Plume Tracker Design

Tracking a plume to its source is a difficult task, as it is highly affected by the turbulence of the media and by the sensitivity of the sensors to both the media and other odorants in the media. In air, a sensor that is quite sensitive is often quite slow, and may exhibit hysteresis or a changing baseline in the presence of the odorant in question. An autonomous agent tracking a plume to its source is required to use one or both of the time-averaged behavior of the plume or its instantaneous concentration to locate the source. When the agent moves quite slowly, its sensors may be very slow without affecting the algorithm, and so sensitive sensors may be preferred, despite their speed. However, when an autonomous robot moving through air is used to track plumes in real time, the signal must be available and stable in a matter of tens to hundreds of milliseconds, depending largely on the scale of the plume and the speed of the robot.

Once a sensor has been built that is sufficiently fast and reliable to be able to detect plumes in real time, the algorithm that tracks the plume must be built. This algorithm must be able to handle the effects of turbulence, which include large regions of space with no odorants, intermittent signals, and packets with different properties. As in all robotic instantiations, it is important to learn what information is necessary for the search, and to make use of this information efficiently, adding extra information only when necessary. This research is still ongoing, particularly in the biological regime. For instance, the properties of odorant "packets" are being considered as possible sources of information in the biological regime (Grasso et al. 1996). What information is encoded in those odorant packets, and how it is used has not yet been generally agreed upon.

In general, most attempts at plume tracking have used the “PC on board” philosophy. The assumption is that a great deal of processing is required to extract enough data to track a plume, as the data used by biological systems (Grasso et al. 1996) may be quite detailed and subtle. Data ranging from edge detection to gradient calculations might be used to track plumes. The “PC on board” approach has the advantage of making it easy to prototype different schemes quickly, as the strategy may be programmed conveniently and tested. However, this requires the robot to carry a load of computational equipment that may not be more effective than simple analog circuitry with minimal controllers; it is not clear that extensive computation is required to lead a robot to an odor source. Rather, it is postulated that a simple reactive robot might be capable of tracking an odor to its source, requiring neither explicit memory nor formal computation.

The robot we use is a reactive walking robot with two different olfactory sensors located at the end which faces front when the robot walks (Figure 3.5). As the fabrication process for these sensors is in its infancy, no two sensors are identical. These sensors, are the resistive polymer sensors described in Chapter 2. The robot uses a reactive strategy which is identical in design to that developed by Kuwana et al. (Kuwana et al. 1996) which was described in Chapter 2. As with their model, the expected behavior is to remain within a plume and walk upstream or downstream, bouncing back and forth from edges of the plume, with equal probability of walking in either direction. The observed behavior, however, is quite different from that expected. In actuality, the robot executes turns often, preferentially moving upstream, and finding its way to the odor source 70% of the time, regardless of initial direction. Herein lies the mystery which inspired this study. Clearly some property of the sensors and/or circuitry biases the robot sufficiently so as to make it likely that in each instance, the robot finds its way to the plume source. The challenge is to understand the mechanism. Once this is understood, the question is whether or not this property may be exploited in the design of robot plume tracking swarms.

3.2. Description of the Sensors

An odor sensor must fulfill a number of requirements in order to be useful for robotic olfaction. It must respond differently to a wide variety of odorants. It must be relatively fast on the behavioral time scale of the agent employing it. It must also be lightweight, be robust, and require little power.

In this section, we briefly describe a set of sensors which seem to fulfill these requirements. These sensors have another desirable property derived from their physical deposition method – they are sensitive to the direction of wind. In Section 4, we will see that it is this property which leads to the ability of our robot to find an odor plume's source.

3.2.1. Resistive Sponges.

Sensors used to track odor plumes in air must have the ability to react reliably to airborne odorants. Moreover, they must react to differing intensities of odorants fast enough to allow the robot to react in real time to changing local concentrations of odorant. In addition, the sensors' range must be sufficiently wide for all regimes of use. This last condition is the most difficult to fulfill in general, as many conventional sensors with acceptable sensitivities are slow to react and have sensitivities which are dependent on environmental conditions.

A new class of polymer-based sensors has recently been made available (Lewis et al. 1996, White et al. 1997), and is under development in several laboratories. These sensors react to a variety of different odorants by altering their physical attributes including audio resonances, fluorescence, and conductivity. The degree to which their attributes change depends largely on the identity of the analyte, but is nonzero for large classes of analytes. These sensors are surprisingly robust, generally reactive (or *broadly tuned*), easily fabricated, and may be cobbled into arrays of different sensors allowing individual identification of the odorant in question by making use of the responses (both temporal and absolute) of the different sensors. A variant of this sensor, developed by the Electronic Nose project at Caltech, is a passive reactive resistor whose properties change when interacting with specific classes of analytes. This resistor fulfills the requirements, as it is sensitive when properly fabricated, fast in its response, broadly tuned, and passive.

The Caltech resistors are polymer-based resistors which act as sponges under the influence of specific analytes. Although these polymers react to a wide range of analytes, their responses to each one is variable, providing combinatorial signatures across many polymers. These resistors are doped with carbon black [1]. This doping produces a conductivity much greater than the natural conductivity of the polymer. The conductivity is dependent on the absolute density of carbon black in the polymer. When the polymer expands under the influence of an analyte, the carbon black granules separate, and the conductivity decreases. On the other hand, in response to some analytes, some polymers will contract, and this will tend to increase the conductivity of the polymer. Thus, an analyte may be detected by an increase (or decrease in some cases) in the resistance of the sensor.

In the laboratory, when generating control strategies for robots, one wishes to avoid the complexity of odor identification. One would rather use a simple sensor that reacts reliably to a given odor not normally found in abundance in a laboratory environment. As these experiments are not carried out in a wind tunnel or a lab with a separate air supply for test objects, odorants employed must be easy-to-make and non-toxic. Moreover, one would prefer a sensor whose electronic interface is relatively simple. Water-vapor plumes generated with vegetable steamers and fans were used for this study. The sensors used were poly N-vinyl pyrrolidone based resistive sensors. The design and cost overhead for the plume is minimal, and mimics wind-blown odor plumes in natural systems.

In resistive sensors, the dynamic variable is the resistance. This will change in the presence of the object or phenomenon being detected. One attractive element in a

[1] A form of carbon in which the carbon is a black powder, a conductive substance which may be mixed into gels and polymers.

polymer-based sensor is its ability to give reliable $\frac{dR}{R}$[2] readings for identical concentrations of analyte. This allows one to measure the concentration of an analyte no matter the resistance of the sensor. Since fabrication of the sensors is in its infancy, and no two fabricated sensors are identical (within any reasonable tolerances), this is a nice property to base sensor circuitry on. In these experiments, we employ a set of simple four-op-amp-circuits[3] that automatically match the baseline resistance of a given sensor. This baseline can then be compared to the instantaneous resistance, providing a measurement of the analyte concentration change. In our trials, we restrict this to one well-known analyte, assuming that all others have a fairly constant concentration which may be ignored.

3.2.2. Switching Properties of Sponge Resistors. The behavioral properties of a robot are a direct reflection of the properties of its sensors and their interaction with the environment. The sensors exhibit behavior that depends on the direction from which the wind is blowing. In order to understand the behavior of the robot, we must first understand the sensors' behavior thoroughly.

The relevant properties of the sensor to our plume tracking robot are the ability of the sensor to detect airflow, and its dynamic properties in the presence or absence of wind. Most important is the speed with which the signal changes state (i.e., from on to off or off to on).

The speed of wind near and around this sensor is a strong effector of the sensor's adaptation characteristics. In the absence of wind

[2]Sensors are characterized by their change in resistance, as the ratio of the change in resistance to the whole resistance is relatively constant when the concentration is low. The aging of the resistors makes measurement of the exact resistance an unreliable reflection of the local odorant concentration.
[3]A set of circuits containing only four operational amplifiers.

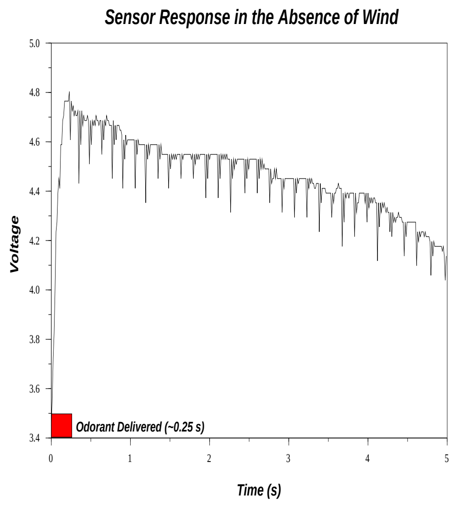


Figure 3.1: This graph gives the response of a typical polymer-based sensor after a brief exposure to water vapor. Despite a fast increase in sensor resistance, the sensor does not rebound to its previous baseline resistance for tens of seconds. This response is unacceptable for behaving robotic systems.

the sensor quickly (<200 ms) reaches its peak resistance. However, the resistor takes more than ten seconds to return to the baseline once the peak has been reached, and the odor source has been removed (Figure 3.1). In the presence of wind the situation changes, and the baseline resistance is recovered more quickly. The same resistor placed just outside of a wind stream produced by a fan with an airspeed of ˜0.5 m/s, displays a behavior

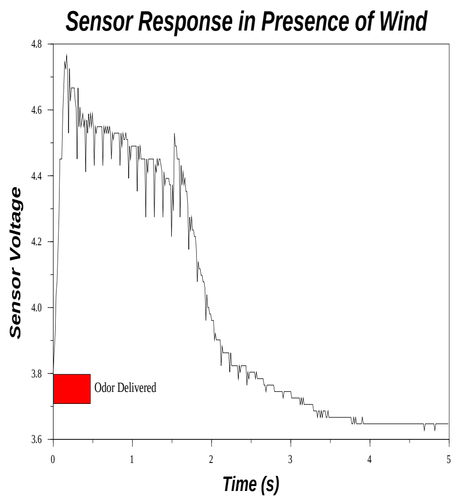


Figure 3.2: This graph gives the response of a typical polymer-based sensor after a brief exposure to water vapor in a system with sustained airflow. The sensor's resistance much more quickly rebounds to its baseline value. This response is fast enough to be useful for behaving robotic systems.

in which the baseline resistance is quickly recovered (Figure 3.2).

This dependence on the presence of wind is a key factor. The fast response allows one to use simple thresholding circuitry to determine whether an analyte is present. In its absence, we would need to determine the time-dependent derivative of the sensor response in order to determine whether or not we are currently sensing the odor; we would also require one bit of memory indicating the state of the boundary we just crossed.

The sensor's properties are not only dependent on the presence of wind, but also on the direction of wind with respect to the face of the sensor. As sensors are coated with the polymer/carbon black solution only on one side, this is expected[4]. This effects primarily the decay rate after presentation of an odorant. In two typical responses, a sensor facing upwind (Figure 3.3) will have a decay rate which is less than half that of the same sensor facing downwind (Figure 3.4).

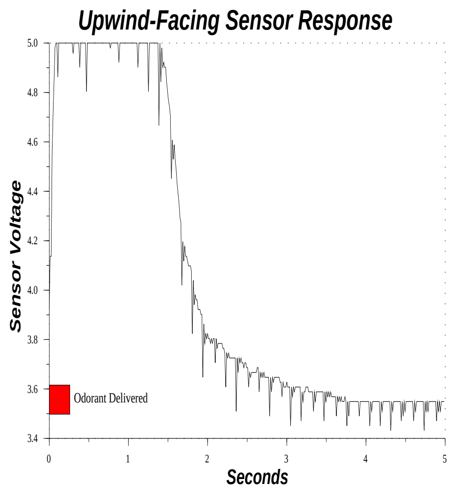


Figure 3.3: This graph gives the response of a typical polymer-based sensor facing upwind, and located in the wind stream. The resistance increases rapidly with the onset of the odor signal, and decrease rapidly with its removal.

[4]Wind directly impinging on the sensor face will tend to increase the rate at which odorant is removed, while that impinging on the other side will also increase the rate, but to a lesser degree.

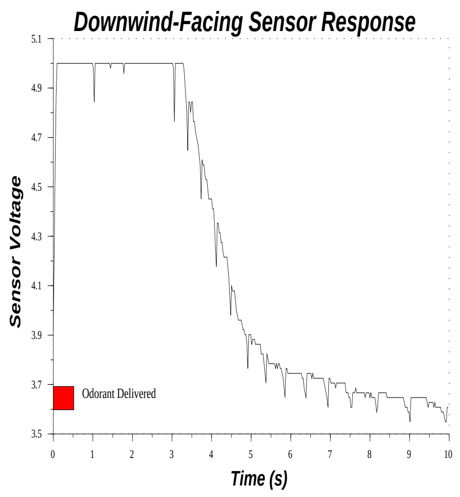


Figure 3.4: This graph gives the response of a typical polymer-based sensor facing downwind. This particular measurement is from the same sensor that produced the upwind response shown in Figure 3. Note that the time required to fall back to baseline is more than twice that of the previous sensor. This indicates that the sensor's response is dependent on its orientation in the stream.

Typically, the time required for sufficient odorant to "off-gas", returning the sensor's resistance to its baseline value, is longer when the sensor is facing downwind than when it's facing upwind. This property arises because the polymer is deposited on only one side of the printed circuit board. The polymer absorbs the analyte quickly, with a correspondingly fast rise time, but does not release it as quickly. The fall time is more than twice that observed with the sensor facing upwind. In this case, the sensor has a two second fall time rather than a one second fall time when the sensor is facing upstream. Concomitantly, as one might expect, the rise time, while being fast, is still not equal to that observed with the sensor facing upwind.

3.3. Description of the Robot

In this section, we describe the robot used in the development of our plume tracking algorithm. We built the robot using a toy chassis developed and marketed by the Elekit company as part of a kit designed to provide a hands-on introduction to robotics (Figure 3.5). This particular chassis is ideal as it is inexpensive, simple to use, and capable of walking. The robot is powered by two nine volt batteries, which provide reliable power for a half an hour behavior.

This system has two severe drawbacks. First, the motors are simple DC motors with unreliable control characteristics. That is, no two motors seem to rotate at identical or comparable speeds in any kit. This produces a ‘limp’, making the robot trace out a large circle, rather than follow a linear track, as designed. The second drawback is the very limited space available in its cargo bulb, making the circuits by necessity, small and simple. This limitation drives the design of simple analog circuits with limited processing, and led to the development of the simple strategy to be discussed.

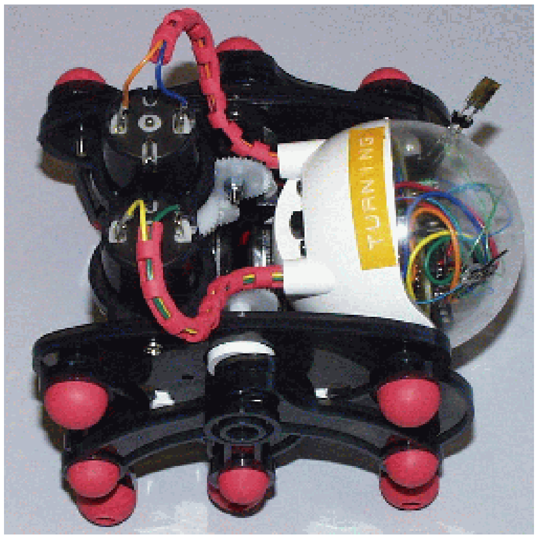


Figure 3.5: This robot, built from a toy kit produced by the Elekit company, contains only two sensors oriented at a 90° from each other, with their mid-angle located along the robot’s apparent line of motion. The robot is powered by two DC motors, each controlling one set of ‘legs’, and has a simple combinatorial logic controlling these motors.

Each sensor is fabricated as described in Appendix A. It consists of a printed circuit board with two conductive leads between which the polymer/carbon black solution has been deposited. Each sensor is oriented with its coated side facing 45° from the direction of forward motion of the robot, and oriented 90° from the other sensor. It is partially occluded from behind or the side by the robot’s cargo bulb.

The resistance is read via a voltage divider, and compared to an adaptive voltage which represents the time-averaged sensor reading. The circuit is given in Figure 3.6.

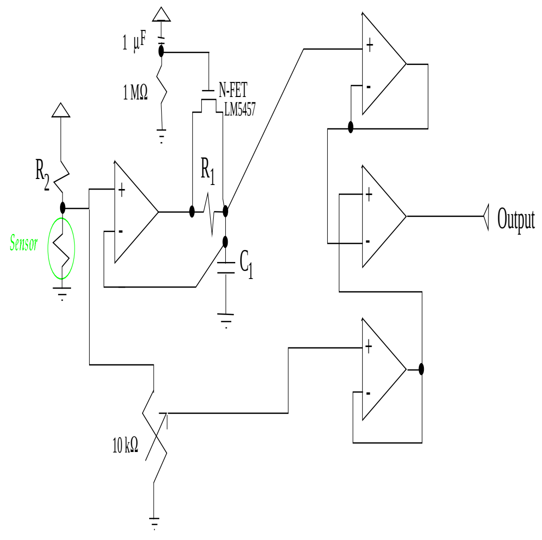


Figure 3.6. This is the sensor circuitry of the robot. The robot consists of two sensors, each with this circuitry. The output of the two circuits is fed into combinatorial logic, which determines the control signals for the motor driver. The circuitry consists of three parts: a voltage divider, a slow voltage follower, and a comparator. This provides binary sensor information.

This sensor circuitry consists of three major parts. The first part is a voltage divider made up of the sensor, which is essentially a variable resistor. The sensor resistance varies with the concentration of odorant – linearly at low concentrations – and with substance being detected.

The second part of the sensor circuitry is a slow voltage follower. The capacitor charges through a 10 $M\Omega$ resistor, making the time for charging (discharging) extremely long, depending on the value of the capacitor. In our experiments, we used a 47 μF capacitor, making the time constant 470 sec, a long time on the time scale of the robot's actions. By contrast, typical resistance changes of the resistor were greater than a 100% change, and could occur completely in $0.1-0.2$ sec. This means that the voltage measured at the capacitor is virtually unchanged for complete resistance cycles which take a matter of seconds. On the other hand, lasting changes in the baseline resistance of the sensor, which would persist indefinitely[5], would be 'followed' by the slow voltage follower, providing an accurate measurement of the time-averaged behavior of the sensor.

The third part of the sensor circuitry is a comparator, which compares a ratio of the voltage across the sensor with that at the capacitor. The ratio is chosen to

[5]These sensors tend to have a monotonically increasing baseline resistance. The cause of this resistance increase is not known.

be small enough that the sensor noise does not sporadically trigger a false sensor signal. A signal, which is reflected by a rise in the voltage across the sensor, will cause the comparator to register a higher voltage across the sensor, and so the arrival of an odorant is registered. The comparator output is then fed into a PAL[6], which decodes the four possible states, and generates motor control signals.

The robot's behavior is based on four possible sensor states, and is determined by the current state. It does not employ any explicit memory in the generation of an action. The four possible sensor states and corresponding behaviors are given in Table 3.1.

Sensor Signals	Robot Behavior
No signals.	Do nothing.
Left sensor signal.	Turn right.
Right sensor signal.	Turn left.
Both sensors signal.	Move forward.

Table 3.1: The four sensor states (left), each correspond to a unique behavior. These actions are determined entirely by the current sensor state of the robot, and is not determined in any way by the previous state of the robot.[7]

Although this strategy does not process any information about the robot's current state or history, it is capable of keeping the robot within the plume. The robot is at an edge when one sensor is on and the other off. The robot then executes a turn, walking back into the plume. There is no explicit directionality built into this design, however.

3.4. Trials

The robot was run through several trials in a simple plume generated in the laboratory. We now describe the trials and performance of the robot in our system.

3.4.1. Plume Generation.

The robot described above was tested in a plume made from off the shelf parts, in an attempt to include as much turbulence and noise in the plume itself as in a realistic case. In order for a robot to work essentially off-the-shelf and in a wide variety of climates and situations, it must have the ability to work in a less controlled environment than that available in a wind tunnel. Thus, our plume was generated in open air from a steam source.

[6]PAL is an acronym for Programmable array logic.
[7]The robot is a Markovian embodied reactive system.

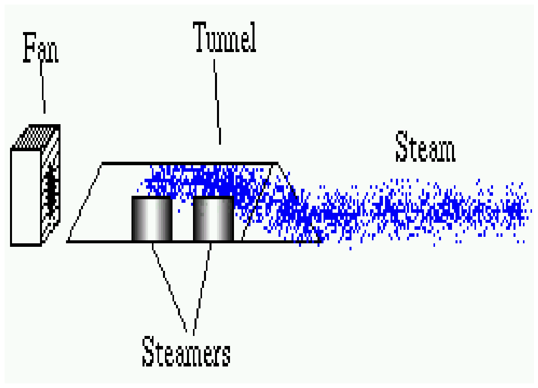


Figure 3.7: This figure gives a schematic view of our plume generator. The generator consists of two steamers within a columnating "tent" and a fan which blows the air both through the "tent" and around the "tent".

Figure 3.7 gives the design of our plume generating system. It comprises of two vegetable steamers inside a small columnating "tent". In our trials, the "tent" is made of Plexiglas, although we have had success using paper, cardboard, and wood, in the same configuration. A single fan generates an ambient wind speed of ˜0.5 m/s, and is at one end of the "tent", pushing the steam out the other side. Figure 3.8 shows a plume being generated using this apparatus.

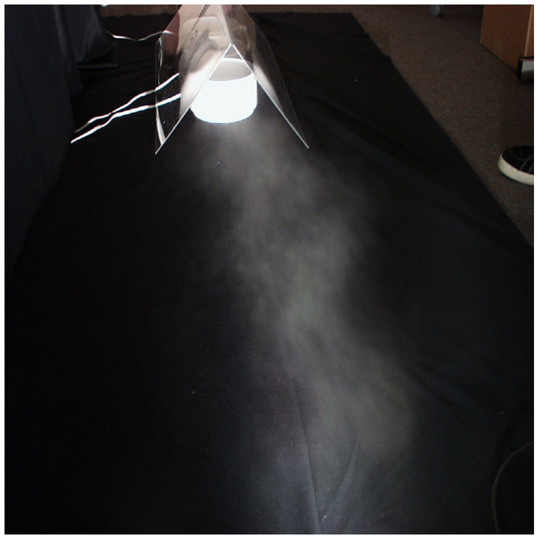


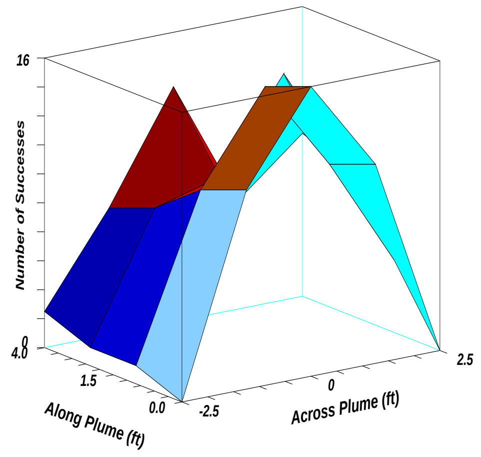
Figure 3.8: This is a picture of our apparatus and a steam plume it generated.

The steam plume has nontrivial structure, as it is generated in a medium of differing temperature. Moreover, the quantity of steam produced by the steamers varies widely, despite continuous operation of the steamers. While the heat has artificially increased the amount of water vapor in the air, this plume is qualitatively similar to other "leaky" plumes.

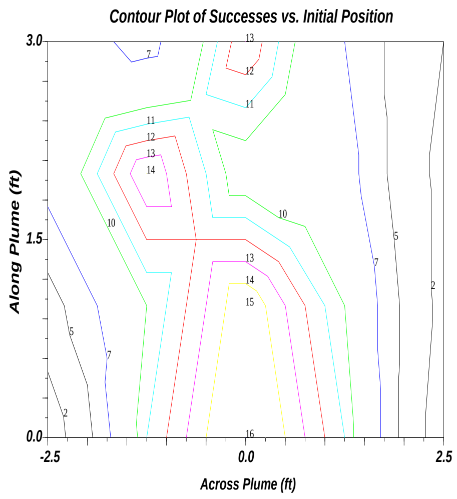
The plume has a well-defined concentration gradient which extends both across the plume axis and along it. Using the sensors described in Section 3.2, we are able to measure contours of average plume concentration. Commensurate with Grasso et al. (Grasso et al. 1997), we find that the plume consists of two regions. The near region consists of high concentration odorant, whose concentration falls off sharply at the edges of the plume, and whose average concentration decreases as one proceeds downstream. As one proceeds downstream, the plume expands, and the edges become less sharply defined. The time-averaged concentration decreases smoothly while the instantaneous concentration becomes highly variable. Despite this, the time-averaged plume remains well defined over several meters.

3.4.2. Robot Trials. Five hundred trials were run during which the robot was placed in the plume and allowed to behave. These trials were begun at twenty different positions located within the plume. These positions were located at intervals of one foot both across the plume and downstream along the plume axis. Each row consisted of five positions. Thus, the robot's initial positions covered two feet across the plume, and four feet downstream. At each position, twenty trials were run. Four initial orientations of the robot were employed: upstream, downstream, and each direction transverse the flow direction. The robot was allowed to behave for ninety seconds, during which the aperture of the "tent" would either be located (the robot entered the aperture), or the run would be declared a failure.

Our data is given in Figure 9. Along the plume center axis, as many as 80% of the runs were successful.



(a)



(b)

Figure 3.9: This data give the number of successes (of all initial orientations) of the robot from differing initial positions. Interestingly, within the plume, the number of successes is quite high, and is surprisingly consistent both upstream and downstream. The general shape of the peaks follows the plume structure, as determined independently using the same sensors and visual inspection. The data indicates that the plume is not straight, but rather curves downstream, a reality that was verified after inspection of these data. Each of the contour levels represents the physical boundaries within which we expect to have that number of successful trials, given the twenty total trials.

The success of the individual run is weakly dependent on the initial orientation of the robot[8]. This indicates that the ability of the robot to locate the source was not heavily affected by its initial orientation, but is rather a result of the long-term behavior of the robot. Notably, the behaviors that led the robot to the source occurred only at the edges, as no turns other than the built in "limp" of the robot occurred while the robot traversed the plume.

Interestingly, as the robot's initial position is moved further downstream, the robot seems to have an increasing tendency to be successful in finding the source when its perpendicular distance from the plume axis is greater. This is commensurate with the expectation that further downstream, the plume will spread out widthwise.

During our trials, two general behaviors were observed which tended to bring the robot to the source when it was headed downstream.

1. The robot would approach a boundary and turn upstream, despite facing downstream at the time.
2. The robot would turn directly upstream from its given spot, and walk directly toward the plume.

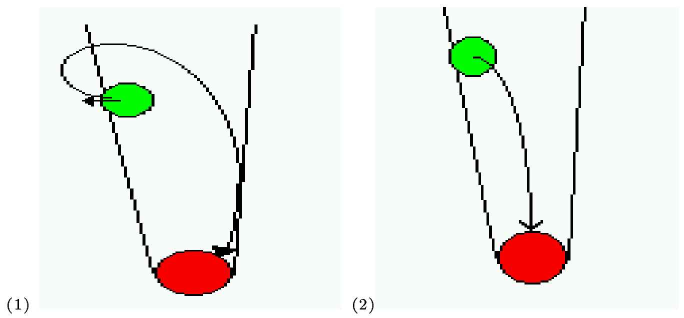


Figure 3.10: These are two general behaviors that brought the robot to the source. The first behavior (1) would be a wide turn upstream, where a downstream turn is expected, and the second (2) is an upstream turn near or at a boundary.

Both unexpected behaviors were observed commonly during robot runs. As these behaviors are not part of the robot's design, their surprising emergence requires explanation. As our electronics are quite simple and understandable, this behavior must have its root in the sensors' intrinsic properties and the dynamics of the interaction of the robot with the environment.

[8]The number of successes for each initial orientation also exhibited this gross profile, but has not been included due to the small number of trials with any given initial orientation. No significant statistical conclusion can be drawn from such a small sample.

Recall that the polymer sensors may be conceptually understood as sponges capable of absorbing specific analytes from the fluid in which they are placed. While these sponges are capable of quickly adsorbing analytes, they have a much lower tendency to release analytes once they are adsorbed. The probability is dependent on the concentration of analyte in the surrounding medium. Thus, as the air speed is increased, hence increasing the ability of the air to remove the analytes surrounding the polymer, the ability of the polymer of off-gas analytes increases.

As the polymer's off-gassing behavior is heavily influenced by wind traveling over it, the increased resistance may be retained over a long period of time when ambient wind is blocked. Thus, a resistor leaving the plume, which should normally return to a low resistance quickly, may retain its high resistance when the robot blocks the wind that otherwise would remove odorant from around the sensor. It may also retain its high resistance if the sensor itself is facing downstream. Moreover, when a sensor enters the plume and is again facing downstream, or is in the "shadow" of the robot, it may not turn on until it is out of the robot's "shadow" and/or facing upstream again, depending on the local concentration of odorant.

We can now understand why the robot turns upstream so readily when placed in the plume. Note again that both sensors are placed at 45° from the direction of travel. This means that when the robot is facing downstream by less than 45°, the upstream sensor is facing upstream. Moreover, the downstream sensor is facing directly downstream, making its dynamics considerably slower than the other sensor. In this scenario, the downstream sensor's resistance will return to baseline more slowly than the upstream sensor's *even when the downstream sensor leaves the plume first.* This will, in turn, cause a turn of type one, leaving the robot facing upstream.

The second type of unexpected turn is of type two. This turn is caused by the upstream sensor turning on before the downstream sensor turns on. Note that while the robot is not facing directly upstream, the downstream sensor is partially or completely occluded by the robot in addition to simply facing downstream. This makes the upstream sensor turn on exclusively, and the robot will turn. Note that this does not happen in all trials. It would seem that the occlusion is only partly successful, and in 70% of trials, both sensors will turn on, making the robot move directly forward.

Failures of the robot are generally of two classes.

1. The robot could not detect the plume and stopped downstream. The robot would occasionally sporadically spin in both directions as a single sensor was triggered.
2. The robot walked completely out of the plume.

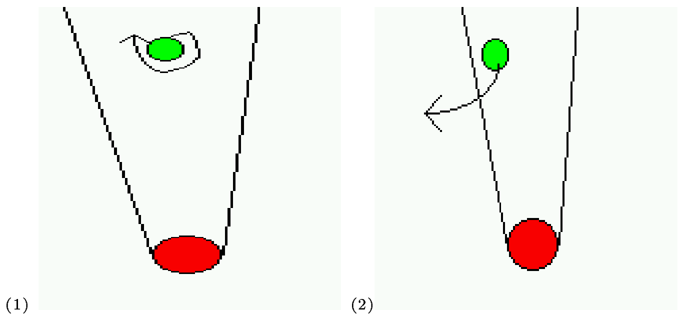


Figure 3.11: These are two behaviors that signalled the failure of a robot during any given trial. The first (1) behavior is an intermittent rotation of the robot, caused by intermittent sensation of the plume. This could be caused by the auto baseline of the sensor, or by the plume dissipation far downstream. The second behavior (2) is the loss of the plume due to the robot walking out of the plume. This often happened when the robot's batteries were fresh, making the robot move faster than its sensors could react.

The first failure is caused by the robot losing the plume as it walks too far downstream to reliably detect the plume. The second failure is caused by the robot walking out of the plume, and out of the airflow. As the robot leaves the airflow, the slow reaction of the sensors is recovered, allowing the robot to walk too far out of the plume to reacquire it. Of the two types of failures, the second one dominates with 70% of failures being of this type.

3.5. Lessons for Plume Tracking Swarms?

This chapter has illustrated plume tracking behavior using an extremely simple strategy coupled with a flawed robotic chassis and complicated odor detecting sensors. The simplicity of the algorithm illustrates that it is possible to solve the problem without powerful processing or careful concentration measurements. Rather, it is possible to use only boundary sensing and wind direction information to find the source.

This raises some interesting questions about biological plume tracking. We have an algorithm that is surprisingly robust, and uses very little processing power. It is a simple consequence of the physical properties of the sensors being used. In biological systems, similar signalling mechanisms may be involved. An adaptive baseline is employed to make gross measurements of odor edges finer, while nulling out sensation after a well-defined sensitive period. It does not seem surprising that olfactory sensors are active at the boundaries, both temporal and physical, of odors, and less active after a somewhat short exposure to the odor.

This study illustrates how two important pieces of information, the local wind direction and the plume edge, may be used when tracking a plume to its source. More notably, we have fashioned a robust algorithm out of only these two pieces of information, and have obtained the relevant information from a single sensor. The way that this information is acquired, by physical properties of odor sensors or by mechano-receptors located on hairs on the agent's body does not seem important. A stochastic walk with a higher probability of moving in the right direction will eventually end up at the target. Hence, simple strategies that keep an agent from losing the odorant and at the same time move the agent upstream would seem to be all that are required. The diversity of natural plume tracking systems serves to illustrate this point. These strategies are largely different, but all seem to include some knowledge of wind direction in quickly moving systems in air. On the other hand, the exact strategy for finding the source varies from species to species. We now believe we understand why information about wind direction is so ubiquitous in search strategies, but any other specific strategy element is not. We would be interested to find a biological agent that uses a similar strategy to that we've illustrated.

In generalizing this result to a swarm, one design parameter must be that the swarm is reactive to wind direction. This means that, at least at the lowest level, the robots making up the swarm must be reactive to wind direction, or employ some method of search that uses this piece of information in some way. As we have noted, the plume structure itself has embedded in it information about wind direction, which may be exploited without the use of explicit wind direction sensors, or sensors which are preferentially responsive, as in this study, to odors arriving while the sensor is facing upwind.

In the next Chapter, we turn to a general class of algorithms based on exploiting the geometric properties of spirals in finding plume sources. We find that the algorithm is capable of tracking the plume to its source by executing spirals, both explicitly and implicitly using wind direction.

CHAPTER 4

Spiral-Based Plume Trackers

In the last Chapter, we presented a simple plume tracking algorithm which was surprisingly successful. It was capable of tracking a plume upstream, an extremely surprising experimental fact. The robot was not originally designed to be capable of tracking the plume, but it was not only capable of exactly that, it also was capable of recovering from an initial downstream-pointing initial state, which was even more surprising. The interesting part of the study was the explanation of this behavior, which found its root in the physical properties of the sensors employed.

In this Chapter, we present a new plume tracking algorithm based on a spiraling behavior. This behavior is based, initially, on the presumption of a single source. The algorithm demonstrates good line-following behavior, which can be generalized to plume tracking, inasmuch as a plume may be considered to have the general shape of a line, which need not be straight. Various parameter variations are explored, generating a good understanding of the algorithm. Of particular importance is the understanding of the realms of applicability of this algorithm. Real robot trials are presented, although the presentation is anecdotal in nature; further wind tunnel-based trials are planned for a later study. Lessons for our eventual leap to a swarm implementation are discussed at the end of the Chapter, and lead into Chapter 5.

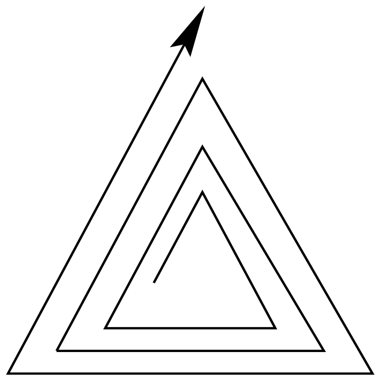
In this Chapter, we utilize two sources of data. One is derived from real digitized plume data obtained in a water plume using flourescing dye (Cowen and Chang 2000). The second is obtained from a simulated plume in which odorant is generated in the form of discrete "puffs" which propagate under the influence of a simulated wind. The use of both models is an important step toward the verification that the simulated data sets perform well enough to represent viable data for extension of plume tracking simulations to behavioral regions that are inaccessible to simulators due to a lack of experimental data. We compare the effects of varying parameters in both models. We view the generation of similar behaviors under such variances to be justification that the important details of the plume dynamics are captured with enough accuracy to validate the use of the simulation in similar regimes.

4.1. Basic Spiral Algorithm

The basic spiral algorithm is designed according to the desire to develop a minimal algorithm to bring an agent from its current position to the source of a plume, without requiring the agent to be initially near the plume. This algorithm, in its basic form, does not require any knowledge other than local odorant intensity, but may be easily altered to include wind direction of various levels of resolution in order to improve efficiency. It is also designed according to an absolute lack of tolerance to failure, a condition we can later relax.

Some applications of this algorithm require that a total lack of tolerance for failure be adopted. That is, these applications, such as the local search of a desert ant for its nest, absolutely require that, in a finite search time (which need not be short), the agent in question will obtain the location of the source of the plume. Therefore, we need to equip the agent with a search algorithm that will never miss the plume from any starting position in an unbounded search space.

We may use a minimal efficiency condition in our design. This is that the algorithm should produce a track which does not cross itself. This requires us to have a *generalized spiral* algorithm. A generalized spiral algorithm is any algorithm which produces a basic movement at the center of an outwardly growing path.



For instance, one might create a triangular motion in the center of the track, and create increasing triangles around that original triangle. Inefficiencies in agent locomotion occur if we use patterns with corners or sharp turns. Thus, we settle on a simple circular spiral algorithm.

The motivation for use of the spiral algorithm is twofold. First, the algorithm is guaranteed to find the source of interest, provided the size of the spaces between loops does not exceed the size of the object in question. Second, rather small modifications of the algorithm can create plume tracking behaviors that are highly robust and quite efficient in the right circumstances, as we shall see.

We begin with a simple example. In the most benign form, a plume may be thought of as a line. In this case, the simplest tracking behavior is one in which the agent moves forward when in contact with the line, and executes a turn when contact is lost, reversing the direction of the turn at each successive loss. This will provide the line following behavior given in Figure 4.1.

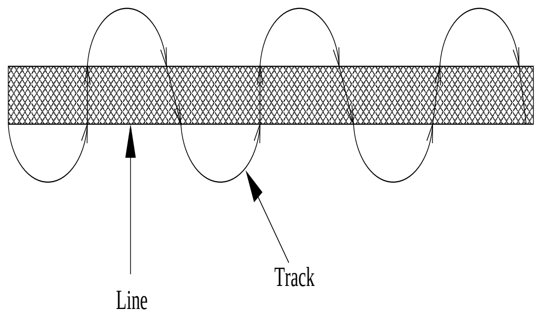


Figure 4.1: This is an example of the use of a line following algorithm in which the agent executes successive turns when it loses contact with the line.

The elegant feature of this method is that it has some measure of error tolerance. That is, if it mistakenly takes the wrong direction along a line the head of which it is meant to find, it will automatically reverse itself and continue in the other direction, as in Figure 4.2.

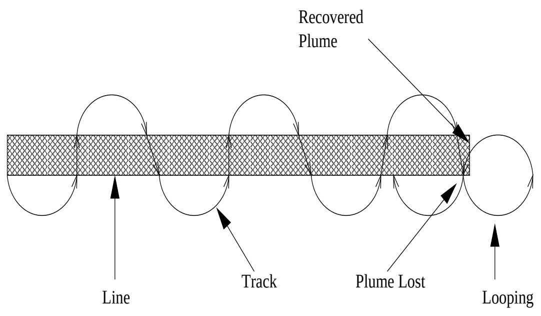


Figure 4.2: The error correction capability of the line following. Reaching the wrong end of a line is easily corrected.

What makes this somewhat unattractive for plume tracking is that plumes are stochastic. The exact placement of a particular odorant packet does not follow any systematic design, and a agent carrying out this simplistic line following behavior may find itself far from the plume due to saturation of sensors or a random variation in the plume's trajectory. If this occurs, the agent will be unable to recover.

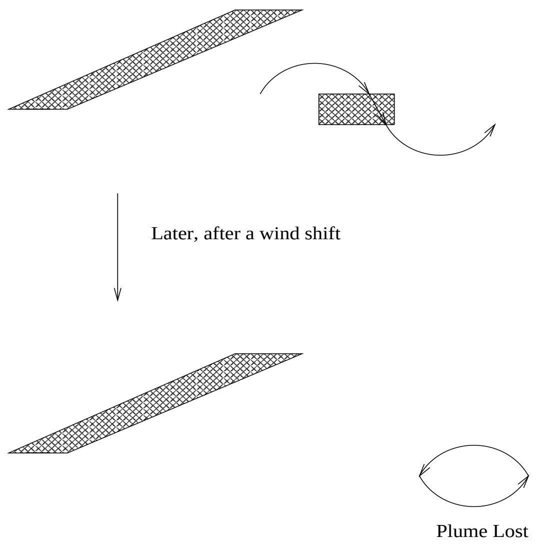
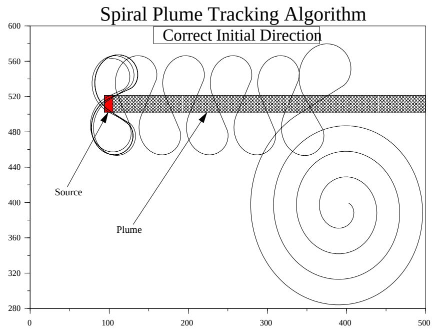


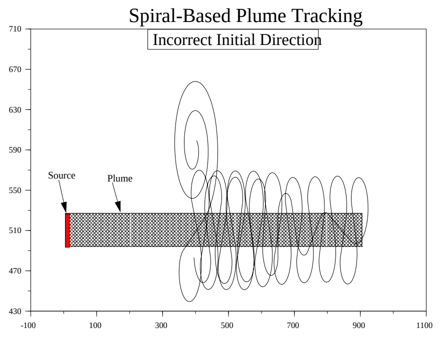
Figure 4.3: The failure of a agent that loses the trail.

There is a movement which gained significant momentum the 1980s after a seminal paper by Rodney Brooks (Brooks 1986) and is still ongoing to make robots simple, and to somehow learn how to do complex things with these simple robots. The goal of the present work, along a similar philosophy, is to report on the development of a particularly simple but suprisingly robust method of plume tracking which is similar in spirit to the algorithm just considered, but also quite robust. We investigate three different algorithms, all of which correct the fault of the line-following algorithm. These are the *basic spiralling algorithm (BSA)*, the *maximum-only switching spiralling algorithm (MOSSA)*, and the *wind-based spiralling algorithm (WBSA)*. We explain each in turn.

4.1.1. The Basic Spiralling Algorithm. The BSA utilizes two behaviors. The first is the basic spiral, while the other is a straight-line tracking. In a basic spiral, the agent initially begins a clockwise or counterclockwise spiral. Upon meeting a trail of interest, the agent travels in a straight line through the trail. When the trail has been traversed, the agent reverses its direction of spiral and reinitializes the spiral. The result is a set of segments of length comparable to the trail length, assuming that the agent executes tight turns.



(a)



(b)

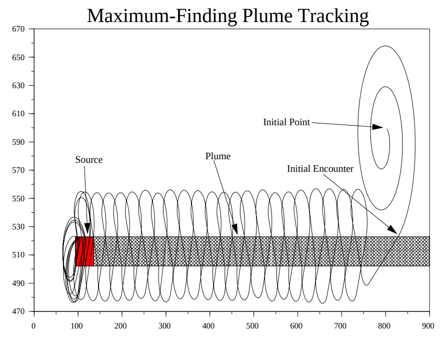
Figure 4.4: These are two typical line following behaviors using a spiral algorithm. Note that it is capable of recovering when it initially chooses the wrong direction. The behavior is very reminiscent of casting[1].

The characteristics of the BSA are quite interesting. As illustrated in Figure 4.4, initially, the agent executes a spiral that is guaranteed to bring it within sensor range of the target trail. Once it reaches the trail, it takes a straight line path through it, and changes direction, beginning its spiral through the trail. The algorithm recovers after reaching the wrong end, and tracks the plume in the opposite direction.

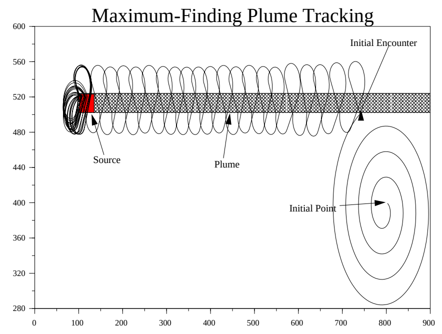
4.1.2. The Maximum-Only Switching Spiralling Algorithm.

The previous algorithm is sufficient to follow lines, but unfortunately, as is evident in Figure 4.4, it does not know which way to track the lines. If a plume is sufficiently long and well defined, a agent employing this strategy might waste a great deal of time tracking the plume in the wrong direction.

[1]A process by which an agent repeatedly traverses a plume as it makes its way upstream.



(a)



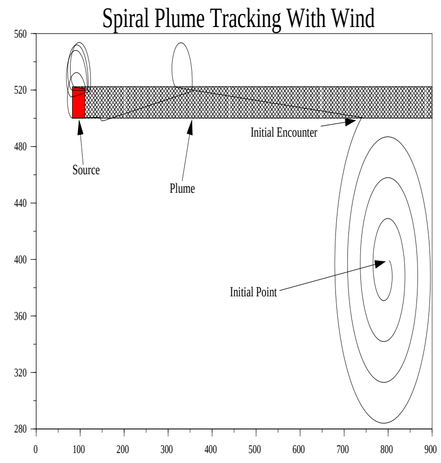
(b)

Figure 4.5: A typical MOSSA running on a gradient plume. The plume's intensity decreases along the x-axis. Notably, from either starting configuration and initial interaction point and orientation, the algorithm correctly chooses the upstream direction to travel.

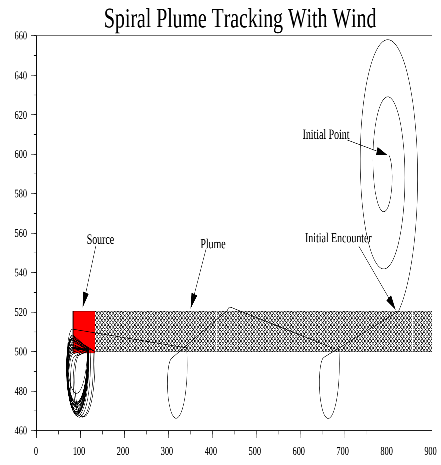
Plumes are stochastic structures which have a higher probability of having a high concentration packet closer to the source. Thus, we are more likely of interacting with a high concentration packet near the source, as compared to further from the source from any given point. By requiring that the new spiral seed point exceed that of the previous seed point, we may restrict the agent from moving downstream, and rather keep it moving upstream. This is, of course, not entirely effective, as it is possible to obtain a high concentration point far downstream of the last sampling point. However, simple modifications of this basic paradigm such as a limited memory of the last point can serve to alleviate this problem.

4.1.3. Wind-Based Spiralling Algorithm. The most commonly used element in a plume traching paradigm is the use of wind direction information. The use of wind direction is nearly ubiquetous in nature, and for good reason. The ability to obtain such information is computationally cheap, and can be done using a passive sensor attached to one of many different types of structures that would otherwise still exist on the animal. The many ways in which wind information may be obtained are as varied as life itself, though they seem to cluster around three main categories: hair based, olfactory based, and temperature based. It is beyond the scope of this paper to investgate these, so we leave off further discussion.

We may use wind information in our spiralling algorithm in order to improve the performance of the algorithm. This information may be used while in the plume, giving us a way of quickly tracking directly toward the plume source. The WB-SA is then identical to the MOSSA with one modification: the agent will turn directly upwind when it has obtained a whiff of odorant above its current baseline measurement.



(a)



(b)

Figure 4.6: The use of wind information in the WBSA is extremely helpful, changing the effectiveness of the algorithm a great deal. In this example, the agent first encounters the plume, turns upwind, and finds the source.

The use of this information is quite effective, as shown in Figure 4.6. Notably, the trajectory the agent now uses is considerably more direct.

4.1.4. Simulations. In the following subsections, we explore several interesting aspects of these algorithms. Several parameters are relevant. We now discuss these briefly.

4.1.4.1. *Sensor Properties.* The sensors utilized in a given robot can be widely varied, and their properties are typically very different. However, each sensor generally has one or two fundamental time constants. These constants correspond to the amount of time it requires to register an increase or decrease in odorant concentration.

For a mobile robot, both of these have important consequences. If a robot has a very slow sensor, it will not be able to react when it enters the plume. This means that it will generally be unable to respond to diffuse or small plumes. On the other hand, if it has a very quick sensor, then it may react immediately to very small changes in the concentration, reacting to a single packet far from the plume, on occasion, or to local variations in a diffuse plume. These parameters determine how a robot will interact with the plume in question, and their choice must be carefully done.

In practice, of course, it would seem to always be more advantageous to use very fast sensors, if available. An increase of the response time of the sensors may be done in software or hardware. However, this does not alleviate the requirement that *behaviorally* the time constant must be well chosen.

4.1.4.2. *Memory Length.* In the MOSSA algorithm, the robot or agent actively carries a memory of the last largest concentration hit it has encountered. However, in real plumes, high concentration packets may move freely downstream, without greatly altering their concentration. This means that the agent may encounter a high density packet while travelling downstream from the last spiral initialization point. In this case, the agent will start a new spiral further downstream. More damaging is when the packet is very near the highest concentration to be encountered. This will cause the agent to spiral indefinitely until some stopping condition is met, assuming that the concentration itself is not the stopping condition.

In order to deal with this, we have augmented the MOSSA algorithm with an ability to "forget" the maximum concentration yet encountered. If c_m is the maximum concentration encountered by the agent thus far, the update at each iteration is

$$c_m = ac_m \tag{4.1.1}$$

where

$$0 < a < 1 \tag{4.1.2}$$

The value of this a will be examined. Simulated plumes will allow us to examine the variation in optimal values of a under the effect of different concentration plumes.

4.1.4.3. *Wind Direction Sensitivity.* In order to track the plume upwind, an agent must first be able to identify the wind direction. In the natural world, as discussed briefly above, this is rather quick and easy. However, for real robotic systems, this is a difficult task, and requires significant time and work to determine the wind direction, particularly in the low wind speed case.

Considering the amount of effort required to produce accurate, quick, low-speed wind direction sensors, it is interesting to understand the wind direction accuracy requirement.

4.1.4.4. *Data Sets.* A plume is typically a very nonlinear construct, consisting of discrete packets of odorant which vary from extremely small in physical size to extremely large relative to the size of the source. Moreover, the packets of odorant have widely different concentrations, varying over many orders of magnitude. The behavior of each packet is also quite difficult to predict, as it is often subject to eddy currents and nonlinear motion (Grasso et al. 1996, Belanger and Willis 1996, Farrell 1999, Grasso et al. 1996a, Grasso et al. 1998, Grasso et al. 1998a, Ishida 1996, Ishida 1994, Kazadi 2000, Kuwana 1995, Murlis 1992, Zanen 1996).

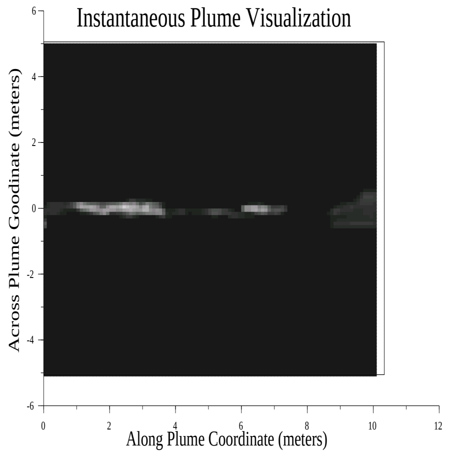


Figure 4.7: This is an example of a plume. This data is obtained by digitizing a plume made up of fluorescing dye and illuminated with a planar laser. In this figure, the odorant concentrations are linearly dependant on the brightness. Note that the plume is extremely concentrated in small packets. The properties of these packets may be very different from instant to instant.

We perform experiments on two different data sets. The first data set (Figure 4.7) is provided by a group at Georgia Tech (Cowen and Chang, 2000). This data is digitized from an active plume of fluorescent dye, imaged by videotaping the brightness of light resulting from fluorescing of the dye in response to a planar laser. The second data set is provided by a plume simulator based on a simulator made by Jay Farrell of the University of California at Riverside (Farrell et al. 1999)

(Figure 4.8). This simulator represents plumes as aggregates of packets of odorant. Each packet is individually created, tracked across a predefined area, and destroyed.

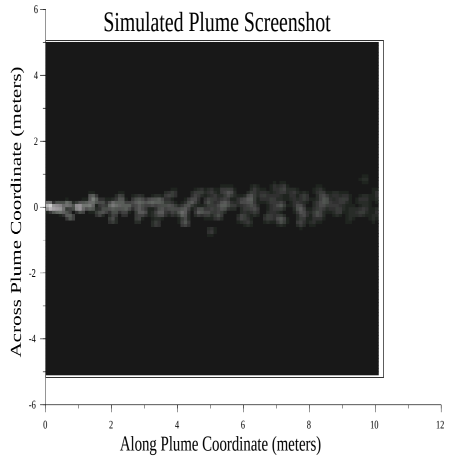


Figure 4.8: This is an example of an instantaneous screen shot of the simulated plume used in this study (Farrell et al. 1999). Each packet is evolved on the far left-hand side of the search area, and tracked across the area, pushed by a simulated wind. Packets are spherical with a gaussian intensity. The packet slowly expands as it travels downstream, making a more dispersed plume of lower intensity.

4.1.5. Performance Measures. In plume tracking, the central concern is progress made upstream. Thus, we report now on the algorithm's ability to track upstream. We carry out one hundred independent runs of a simulator in which an agent moves toward the plume source, basing its movements on one of the algorithms given above. The agent has a maximum speed of 17 m/s, the minimum time that an agent may take to reach the source (judged as 10 m from the source along the plume axis) is $\frac{x}{17.0}$ where x is the distance measured in meters. This means that the performance may be judged as a function of the ratio in the average time to source and distance to source vs. the minimum time to source and distance to source. As described in the caption to Figure 5, we scale our results by the minimum time and distance to source, generating two dimensionless numbers. These numbers, the distance decrease factor (DDF) and the time decrease factor (TDF) are defined as

$$DDF = \frac{d_{min}}{d_{actual}},\ TDF = \frac{t_{min}}{t_{actual}}\ . \tag{4.1.3}$$

Note that these numbers must be smaller than one in all cases.

4.2. Olfactory Sensor Sensitivity

Many different technologies exist for making wind direction sensors. As discussed in Chapter 2, these range in design from saline solution measures, to devices that use changing resonances, to conductivity measurements. In each case, the properties of the sensor are different from those of other sensors. However, the response of a particular sensor may be characterized in many cases as the response of two competing charging/discharging processes, each of which has a different time constant. The relationship between these can be a serious affector of the success of the algorithm, as we shall now investigate.

We consider the effect of the sensor properties on the WBSA. Generalization of the general properties found in a study of this algorithm are expected to apply to the other algorithms, as the question here is the interaction of the agent's sensors with the plume. We note, however, that once a plume has been contacted, the goal would seem to be to move upstream as quickly as possible. A trade-off would seem to be in play between the speed of interaction (rise and decay) and the need to stay near the plume. While the details for MOSSA and BSA are likely to be somewhat different, the general design paradigm is thought to be similar to what is found here.

We first assume that the behavior of the olfactory sensor may be modelled as

$$S = \left(f\left(c\right) - S_0\right)\left(1 - e^{-\alpha t}\right) + S_0 \tag{4.2.1}$$

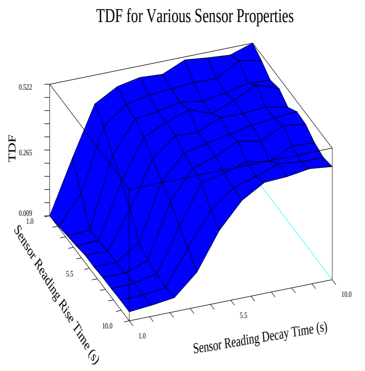
where S represents the sensor reading, f represents some function of concentration indicating the equilibrium response of the sensor, S_0 represents the instantaneous reading of the sensor, and α gives the time constant of increase of sensor response. By the same token, we may assume that decrease is given by

$$S = \left(f\left(0\right) - S_0\right)\left(1 - e^{-\beta t}\right) + S_0 \tag{4.2.2}$$

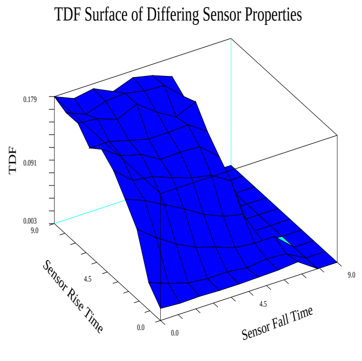
where β is the constant giving the rate of decrease of the sensor response.

If we assume that sensors behave in this way, which is a good approximation for many sensors including the Caltech Silicon Nose Project's polymer sensors (Lewis et al. 1996), we may inquire as to the effects of changing the time constants relative to one another.

We simulate the search for the source as described above, using one hundred independent runs for each set of sensor properties. We model the sensors as given in equations (4.2.1) and (4.2.2) above. Sensor time constants range between 0 and 9 seconds in length. We report the TDF measure of performance, the DDF measure of performance, and the average percentage of agents which find their way home.



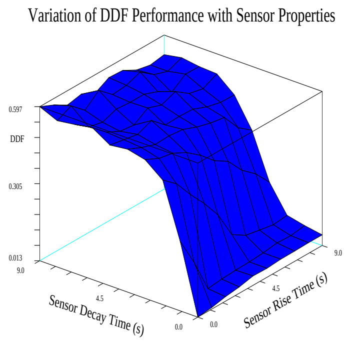
(a)



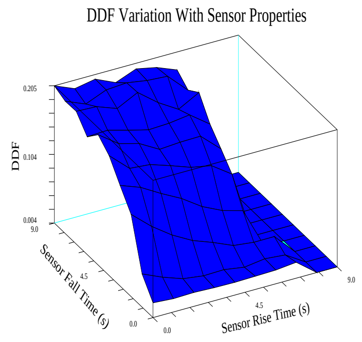
(b)

Figure 4.9: These are the TDF measures of performance for both the simulated (a) and real (b) plumes. In both cases, the performance is worst when the sensor properties are such that the rise time is much longer than the fall time. The properties are more strongly favorable when the sensor rise time is quite short, and the sensor reading decay time is quite long.

Figure 4.9 gives the TDF measures of performance of simulated and real plumes. Both cases are similar in that they indicate that short sensor rise times are favorable, and long sensor decay times are favorable. Figure 4.10 also indicates this, though more strongly.

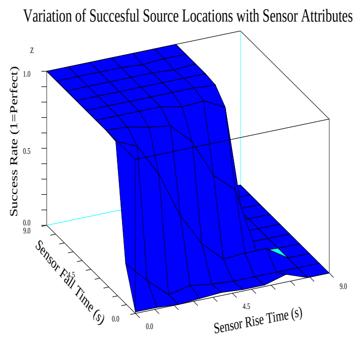


(a)

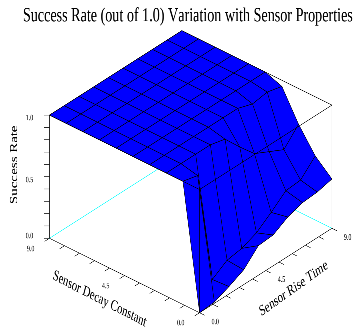


(b)

Figure 4.10: These are the DDF measures of performance for both simulated (a) and real (b) plumes. These have the same general performance indications as those found by examining the plots given in Figure 9.



(a)



(b)

Figure 4.11: These figures illustrate the actual success rate (out of 1.0) of agents tracking the plume to its source. Interestingly, the two models differ in their predictions about optimal behavior.

Figure 4.11 gives the performance of one hundred independent runs on the tracking in terms of the fraction (1.0 is all agents) of agents that actually find the source. The simulated data set indicates that the value of the rise times can vary largely when the decay time is long, providing intermediate performance, while the real data set implies that this is not the case. This discrepancy arises from the differing plume properties, indicating that the requirements of differing plumes will not be identical, but rather will vary with the plume properties.

Empirically, we have found that the simulated plume has a larger gradient along the plume axis than the real plume, exhibiting a change in the concentration of the plume along the plume axis. The real plume is much more uniform than the simulated plume, exhibiting a very small change in the concentration of the plume in the downstream axis. This may be the reason for the differing sensor attribute effects, though a more thorough study of these properties is beyond the scope of this paper. However, in both cases, the best performance occurs when the sensor properties are such that the rise time of the sensor is small and the decay time is long. We consider this to be an important attribute of successful sensors. Note that the decay time need not be a function of the sensor, but rather may be simulated in either discrete electronics or in software.

4.3. Memory Length Sensitivity

In the MOSSA, no wind information is needed to track the plume upstream. In theory, this algorithm may work as in Figure 4.5 in the perfect laminar steady state plume. However, in practice, there exists no gradient (other than a probability gradient for concentrated packets) to base the behavior on. Thus, the actual track is much more variable than that given in Figure 4.5. Rather, we have the situation given in Figure 4.12.

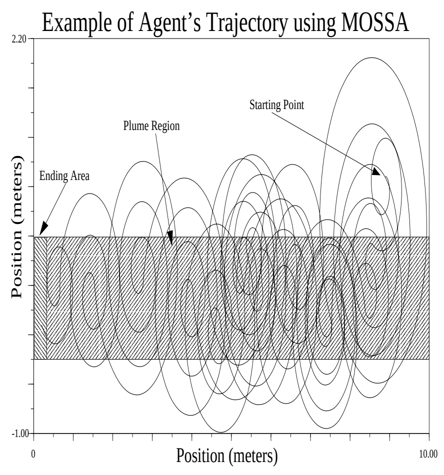


Figure 4.12: A single run of the MOSSA algorithm in the presence of a real plume. Note that the track tends to overlap itself, owing to the nonzero probability of finding a higher concentration of odorant downstream from the current spiral initiation point.

In general, it is empirically determined that one critical parameter in MOSSA is the length of time for which a particular high density measurement is retained in memory. More concretely, we model the behavior of the measure of the greatest previous measurement m_0 occurring at time t_0 to be

$$m = m_0 a^{-t} \tag{4.3.1}$$

where a is some positive number between 0 and 1. The use of this in a real plume comes from the uniformity of the plume, and the existence of high density packets throughout the plume. For instance, if the agent encounters a high density packet downstream, there is absolutely no way it can track upstream outside of executing a spiral and eventually stumbling on the source without ignoring or somehow reducing the measurement it previously made.

The difficulty in utilizing this method derives from the fact that if the rate at which the maximum measurement decays is excessive, the agent will simply retrace its steps indefinitely. On the other hand, if it is too small, the agent will never adapt to the new levels, and will experience one large-intensity packet, spiralling indefinitely from that point forward.

We investigate this parameter by initiating several independent MOSSA runs with differing parameter settings and reporting the performance measures. Each point is defined by one hundred independent runs, with all independent runs initialized 9 meters downstream, and one meter from the center of the plume.

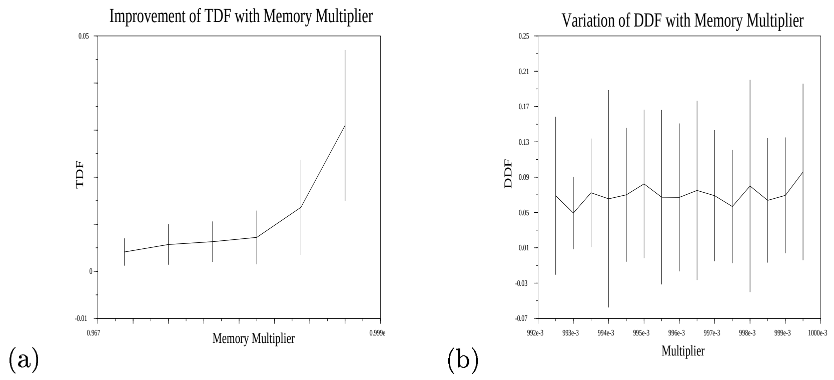


Figure 4.13: The TDF performance of the two models differs greatly. The performance on simulated data(a) is very strongly positively affected by increasing a. The performance on real data (b) does not seem to be affected by changing a.

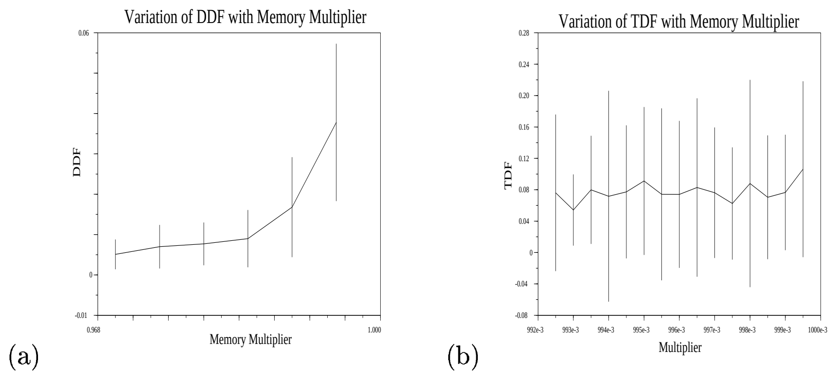


Figure 4.14: The DDF performance of the two models. As before the simulated data sets do much better.

Interestingly, we obtain very different behaviors using the two models. In Figure 4.13, we give the TDF performance, and in Figure 4.14, the DDF performance on these runs. In both cases, we see that increasing the parameter a improves the performance. That is, the longer the simulated runs remember what they've experienced, the better they seem to perform. However, this seems to have no effect for the real-data runs.

Of course, this makes sense because of the profiles of the different data sets. In the simulated data, there is a measurable spreading out of the signal downstream, while in the real data, there is no such large effect, with the only variation being extremely slight; remembering the maximum in an isotropic environment has little effect while remembering it in a variable environment has a large effect.

4.4. Wind Direction Sensitivity

A plume is generated by moving odorant carried by wind or other mass transport. It stands to reason that the determination of the direction of travel of the wind is enough information to cause the agent to travel to the source of the plume. Indeed, this is certainly the case in the natural world. Many (perhaps most) macroscopic mobile life forms seem to have the capability to track the wind or fluid flow; certainly most life forms that tracks odor have this capability. However, while animals have the ability to track the wind upstream, the level of sensitivity vs. behavioral averaging is currently unknown. Since the determination of direction can only be done approximately in any real instantiation, it is interesting and important to understand how the sensitivity of the wind direction sensor will improve the performance of the algorithm. This provides both clues to the reasons that animals might or might not have explicitly sensitive sensors, and motivation for generating wind sensors of a given sensitivity in the robotics world.

In this Section, we investigate the role of increasing the sensitivity of the agent's wind direction sensor. We do this by first explicitly increasing the sensitivity. We assume that the agent knows two facts:

1. The wind is coming from the front, or
2. The wind is coming from the left or right.

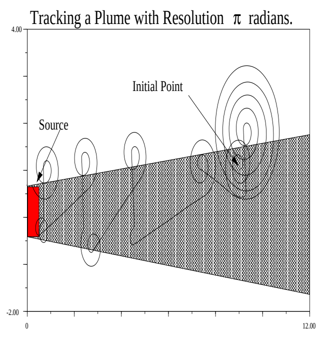
This knowledge is tempered by a specific angular sensitivity. The agent is assumed to believe that the wind is coming from the front if the direction differs from directly ahead by less than some resolution θ. We can vary the resolution and view the effect on the agent's capabilities.

We secondly investigate the role of noise in the determination of wind direction. We motivate an understanding of why one might want to have noise, even artificially generated, when determining wind direction, and demonstrate this empirically. This would seem to indicate that, rather than evolving highly sensitive wind direction sensors, Nature may have opted simply for increasing the noise on a given wind sensor. This would seem to be a design paradigm easily adopted in the development of plume tracking agents.

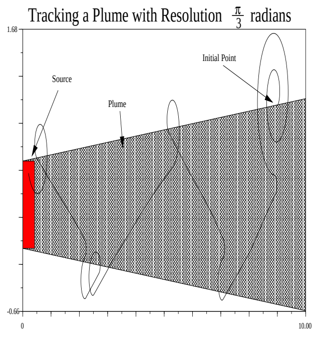
4.4.1. Simulation Description. We carry out one hundred independent simulations on the two data sets previously described. Each agent is endowed with only very basic capabilities, and carries out the search in complete isolation from all other agents. Each agent has an olfactory sensor and a wind direction sensor. The sensitivity of the wind direction sensor, and its inherant noise, are variable.

Each agent data set consists of one hundred independent agent runs. All agents are initiated at 9 meters downstream and one meter above the center of the plume. This allows the plume to be quickly located, giving a good measure of the affect of the change in the agent's performance due to the use of wind information. Note that since the agent will not move toward the source until it has located the plume, the maximum score is contrained to below 1. All agents have a uniform random noise in their movements. Each wheel is noisy in its determination of speed, with absolute magnitude of the noise constrained to 10% of the desired speed. This tends to produce spirals that are initially reasonably perfect, but later become very mishapen.

4.4.2. Explicit Sensitivity. As described above, the sensitivity of the (virtual) wind direction sensor used in this study is the free parameter. This sensitivity can be thought of as being the maximal deviation from directly ahead that will cause an error in the determination of direction – making the agent believe that it is going directly upwind when it is not.



(a)



(b)

Figure 4.15: The progression upwind of an agent endowed with a wind sensor with resolution (a) π and (b) $\frac{\pi}{3}$ radians. The former travels over a significantly longer distance than the latter.

If the sensitivity of the agent is low, as in Figure 4.15a, the agent spends a great amount of time casting back and forth across the plume, while greater sensitivity provides more direct travel to the source, as in Figure 4.15b.

We examine the question by running virtual agents in the real and simulated data sets. The agents are initialized near position located 9 meters downstream and one meter from the center of the plume. Robots are considered to have arrived at the source when they come within twenty centimeters of the source.

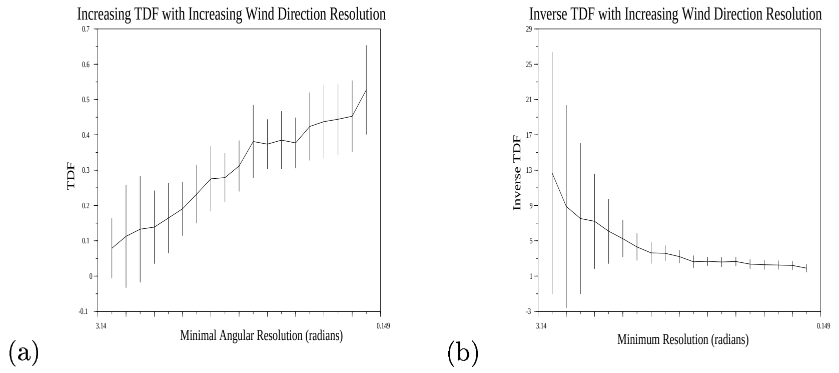


Figure 4.16: The TDF and TDF^{-1} performance measures of the agent performing on the real plume data as the directional resolution is increased. The performance is monotonically increasing, but seems to level off (b) at between eight and ten directions. After this, the improvement seems quite small, indicating that eight directions, or a resolution of 0.40 radians, is sufficient for the majority of the gains to be had by increasing the angular resolution of the sensor.

In Figure 4.16 we plot the performance measure TDF and its inverse. As the inverse plot is linear, this indicates that the the gains from increasing the number of resolvable directions is a monotonically decreasing function. However, it also indicates that at no time will the improvements end. This means that the only constraint in the design of a wind sensor is its cost, as improvements in sensor resolution provide ever improving performance.

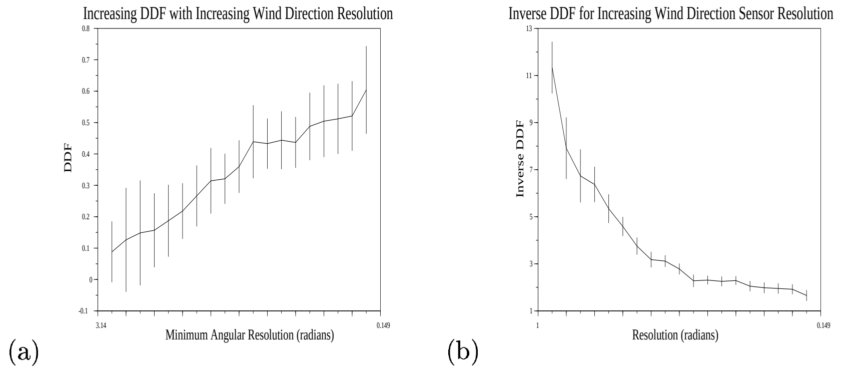
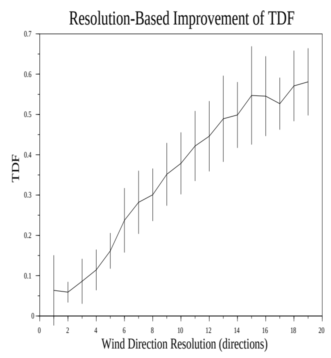


Figure 4.17: Plots analogous to Figure 16 using DDF rather than TDF. Notably, the plot in (a) is linear, indicating continuing rewards for increased sensitivity to direction. Plot (b) indicates a reduction of the returns as the number of directions increases.

Figure 4.17 also underscores this, providing the analagous plots using the DDF performance measure. Again, the inverse plot is linear, indicating decreasing returns. Moreover, the simulated plume reinforces this result, as the behavior of the agent is similar in the simulated plume to the behavior in the real plume.

(a) 

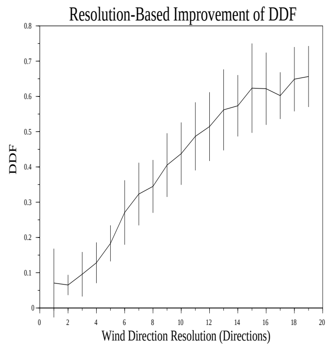
(b) 

Figure 4.18: TDF and DDF measures of performance in a simulated plume. The behavior is similar to that of the robot in the real plume, indicating that the plume is a good approximation of a real plume and that the improvement in performance is not dependant on the real plume data set.

4.4.3. Gaussian Distributed Noise. In general, the measurement of the wind direction by use of a single hair or multiple hairs is problematic, as it can be affected by the very turbulence that causes the nonlinear chaotic behavior in a plume. This means that determination of the exact direction is not particularly reliable, and that the given direction is an approximation to the true direction. In this SubSection, we investigate the effect of noise on the ability of a robot to find its way upstream.

While tracking a plume upsream using wind direction information, a choice of direction is made based on the wind direction sensor data, and possibly other olfactory data. Once the searcher believes it is travelling upstream, it will make a beeline toward the source. This is as we saw in Figure 4.15. However, if the determination of wind direction is continuous, and the searcher is convinced that it is not moving upstream, it will attempt to correct its direction, which in our case means rotating. The result is that the robot in question would seem to require more time to get upstream. However, at the same time, it gains precision in moving upstream, yielding an overall improvement, as can be seen in Figure 4.19.

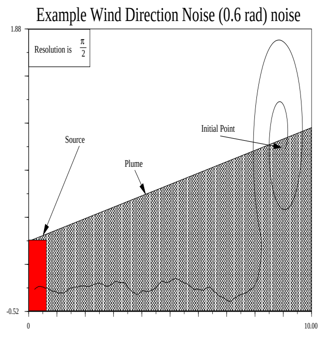


Figure 4.19: This gives the effect of adding a large amount of noise to the instantaneous determination of wind direction. The constant correction of direction causes an average upstream direction, despite the limitation in the resolution inherant in the sensors.

This yields two competing tendencies. The first is that as the noise increases, the accuracy of the path increases, decreasing the path length. The second is that the price for accuracy is an increase in the time, as the time to correct the path becomes increasingly long. We plot these trends in Figure 4.20. In these graphs one may note that the increase of noise initially improves performance, but later damages it, as the path takes a prohibitively long time to complete.

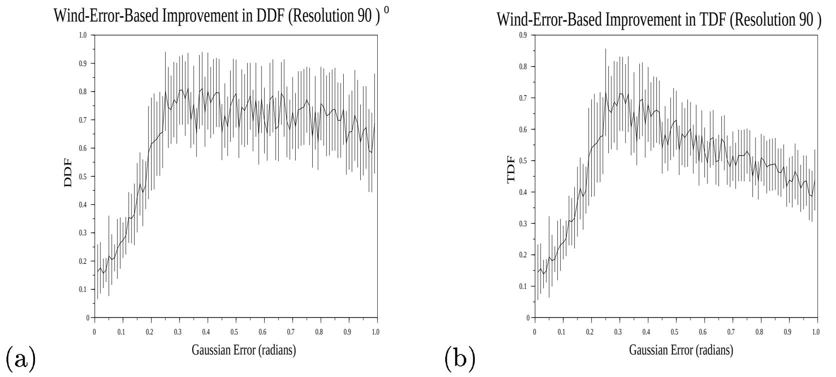
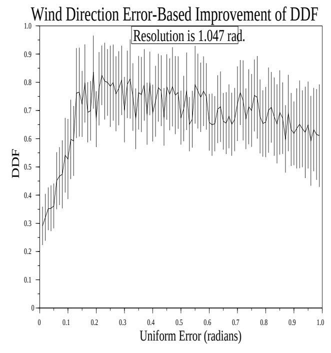


Figure 4.20: In (a) we see the effect of increased noise on the distance travelled. This measure of performance is somewhat constant despite the ever increasing noise. This indicates that the distance travelled, which does not include distance travelled while turning, is relatively constant with the increase in noise. (b) indicates that the time initially decreases, and then increases, as the time spent making corrections increases.

What these graphs seem to indicate is that the plume tracker will benefit from a limited amount of noise. Moreover, increasing the base resolution of the robot from $\frac{\pi}{2}$ to $\frac{\pi}{3}$ does nothing more than shift the graphs, yielding the same overall behavior.

(a) 

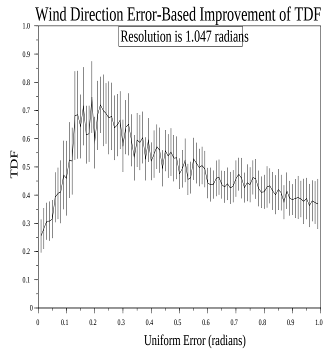
(b) 

Figure 4.21: These graphs are identical in form to Figure 4.20 above, indicating that the performance in the face of noise, while dependant in the position of the peak performance on the noise, is identical in form in both cases. Thus, any given resolution would seem to have an optimal noise level which is nonzero.

Thus, we conclude that differing resolutions will tend to have optimal noise levels, despite the deviation in precise noise level. The functional dependance of the noise level on the resolution would seem to be highly dependant on the plume, and the odor sensor properties.

4.5. Robot Trials

The true test of any robotic algorithm is its development on a robotic platform. While it is not possible to properly implement these algorithms and generate meaningful statistics on a robot outside of a continuous flow system, we generated anecdotal data about the implementation, yielding information about its applicability, and information about required variations to the basic strategy.

The test bed for our robotic implementation is a robot developed by Owen Holland at the University of West England, now called the Moorebot. Its fabrication and development of control software were done in the Caltech Microsystems Lab by the author, Adam Hayes, and Ladd Van Tol. Its basic system is a 386 processor produced by AMPRO^{TM} Computers, and a custom motherboard and chassis developed as indicated above. The robot is powered by a sealed 12V lead acid battery, and moves via two powered wheels and a castor wheel at one end of the robot. Custom hardware developed here at Caltech include proximity sensors and communication beacons, as well as the various hardware items designed to measure odor concentration and wind direction.

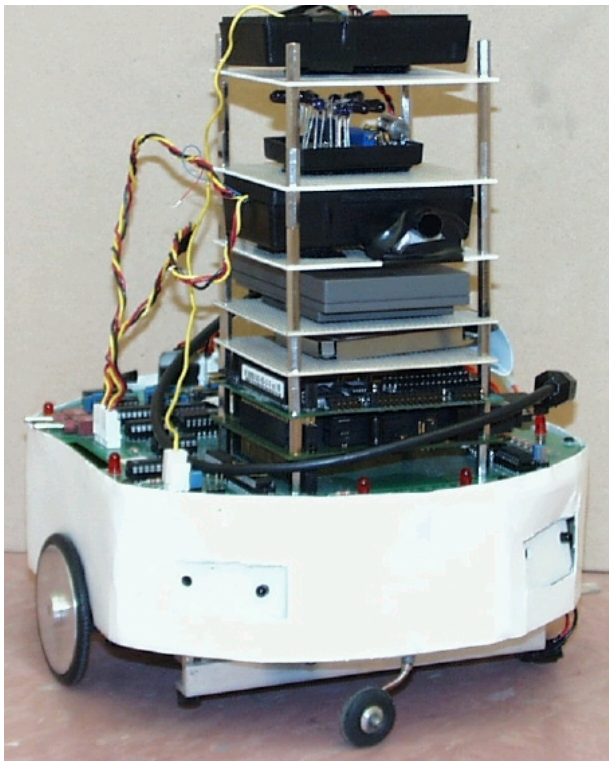


Figure 4.22: The Moorebot.

Along with the AMPRO CoreModule ^{TM}, the PC104 stack contains a PCMCIA card connected to a radio LAN. The system is a stripped version of RedHat 6.0 Linux, and all system files are fully contained on a 124 Mb solid state hard drive.

Both BSA and MOSSA have been implemented on this test bed, the latter simply as a result of the failure of the first. The plume system is similar to that described in Chapter 3, although it was necessary to increase the number of steamers to four, due to the insensitivity of the polymer sensors. All runs were conducted in an arena of size 20'$\times$20'. Because the arena in which these trials were conducted is not adequately ventilated, water vapor quickly saturated the arena (within 15 minutes), making any measurements with the sensors impossible. As a result, we only decribe our success with the two algorithms, and do not offer any systematic experimentation or statistics.

All trials of both BSA and MOSSA conducted with new polymer sensors (see Appendix A) and in an arena with little or no water vapor were successful. The trials were typically started approximately twenty feet downstream from the source. As the plume was in reality a jet, its extent was not known. However, the runs in low saturation conditions typically employed large spirals at the distal end of the plume, and employed alternating tight spirals (casting) in the proximal plume.

As the arena became more saturated, it became necessary to alter the algorithm. This was the origin of the MOSSA algorithm. This algorithm was successful only in low concentration conditions, and failed entirely as the olfactory sensors became completely saturated. In this regime, the MOSSA algorithm completed the task, while the BSA algorithm became hopelessly lost downstream in every run.

4.6. Conclusions and Generalizations to Swarms

The strength of the spiral algorithm in carrying out a plume tracking behavior derives from its robustness in localizing a source from any given initial position. If the source of the plume and its associated plume is of physical extent equal to or greater than the distance between any two successive spirals, the search algorithm will find it.

The spiral algorithm is affected by many parameters which help to define its instantiation. These include the sensor properties (which are generally fixed by the physical attributes of the sensor used), the decay time of a maximum sensor reading, and the sensitivity and inherant noise of the wind direction sensor. We by no means limit the parameters to this list, but have investigated these as we believe that these are of primary importance. Perhaps the most suprising result of this study is the advantageous property of wind direction sensor noise in localizing the plume.

We have not found, in this Chapter, any properties that may be improved by a swarm. Thus, we do not expect a swarm to be able to improve any of the aspects of plume tracking discussed here. Specifically, it is unlikely that a swarm will positively affect the speed in which a plume may be tracked, *if the plume can be reliably detected throughout the search.* This last condition, however, is not satisfied in the low density plume limit, and we expect that the only place where a swarm might be useful is in this limit. The next Chapter will focus on this aspect of plume tracking, and demonstrate that it is precisely this regime of plume tracking to which we may apply our swarm engineering. Moreover, we will demonstrate that there is little advantage to the use of a swarm in any other regime.

We have also noted that the behavior of our simulated agents in simulations utilizing both simulated data and real data is essentially identical. The behavior due to the “patchiness” in the data sets produced similar behaviors, as did all parameter variations with the exception of that resulting from the use of the MOSSA algorithm. In algorithms which used wind information, though the quantitative behavior differed, the qualitative behavior was extremely similar. Thus, we conclude that in algorithms which make use of wind direction information, the use of the simulated data set will not differ significantly from the use of real plume data, and so we adopt it as our source of plume data, whenever the analagous real data is not available.

CHAPTER 5

Swarm Extension of Plume Tracking

In the previous Chapter, we demonstrated that a simple spiral-based search algorithm can locate an odor plume, and a perturbation of a spiral can reliably track the plume to its source. The approach is virtually incapable of losing a plume *when the plume is available in large concentrations.* However, the weakness of this approach, indeed of all single robot approaches, is that when a plume is particularly diffuse, the robot will often lose track of it. If lost for a long enough time, the robot may never recover the plume.

One approach to dealing with the loss of a plume is to give the robot some intelligence and have it search the local area thoroughly taking into account the direction of the wind and the relative position of the last hit. This approach would seem to be feasible if three conditions are met:

1. The robot has a large memory.
2. The robot is capable of localizing itself and other landmarks.
3. The robot is capable of reasoning the probable position of a plume given the previous hit, previous wind direction, and current wind direction.

If we assume that the robot has a good idea of the wind direction, its current direction, and enough memory to remember the last direction, the last condition is currently capable of being met. However, the first two conditions are problematic. The question once the first condition is posed is exactly how large the memory should be to accommodate whatever facts are required. The second condition requires that one answers the question of how landmarks may be recognized from multiple directions, a difficult problem from computational vision that has yet to be solved.

As we have previously stated, the approach we take here is a rather simpler approach. We propose to take a number of different robots of identical design, endow them with communication, and allow the *swarm* to localize the source. The use of multiple robots means that one robot may receive an initial smell and recruit other robots to the area, allowing the other robots to search the local area while the first marks the spot of the first encounter with the odor. This allows mutiple roaming sensors to be used simultaneously in an identified region of search, providing good coverage, and a larger probability of finding either the plume or the source.

5.1. Experimental Plume Simulation

In this study, we are interested in exploring the regions of sensitivity of single simulated robots, and their swarm counterparts. We utilize a plume simulator created by Jay Farrell of the University of California at Riverside (Farrell et al. 1999) in all of these simulation experiments [15]. This simulator delivers odorant to the search space in the form of packets. Each packet is spherical in shape, has a well-defined size, and has a density which drops off as the distance squared (r^{-2}) from the center of the plume. Each packet is carried by a simulated wind downstream. As the packet ages, it widens, lowering its peak concentration, and increasing the area over which it has some influence. The wind is designed to emulate natural wind, and to produce plumes with meander. This allows one to investigate the effectiveness of plume strategies in variable wind conditions.

Many parameters are important in specifying a plume. We summarize these in Table 5.1.

Quantity	Value
Packet Initial Size	0.1 cm
Packet Growth Rate	10.0%/s
Wind Speed	1 m/s
Plume Length	10 m
Wind Meander Parameter	2.5

Table 5.1: This table gives several of the parameters associated with the plume. The values given here reflect the condition of the plume in use. The wind meander parameter is a characteristic of the plume's inherant spectral noise [15]. It is beyond the scope of this paper to thoroughly discuss this parameter.

A simple variation of the original simulator allows the concentration of the plume to be greatly varied, making study of the behavior of different algorithms under widely variable conditions possible. In Figures 5.1-5.3 we illustrate the various concentration levels, spanning three orders of magnitude. Notably, in the most sparse plume, the probability of encounter is exceedingly small.

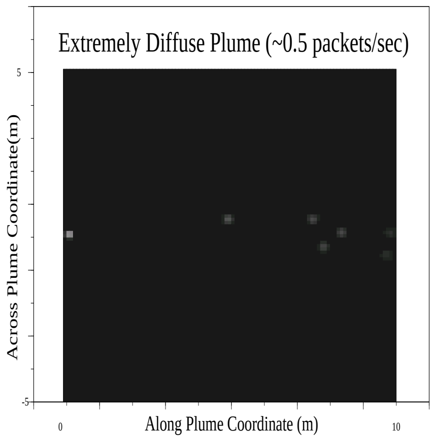


Figure 5.1: A simulated plume with 1 packet/second released into the simulated wind.

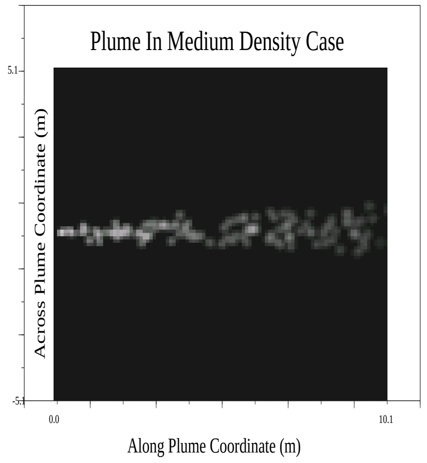


Figure 5.2: A simulated plume with 10 packets/second released into the simulated wind.

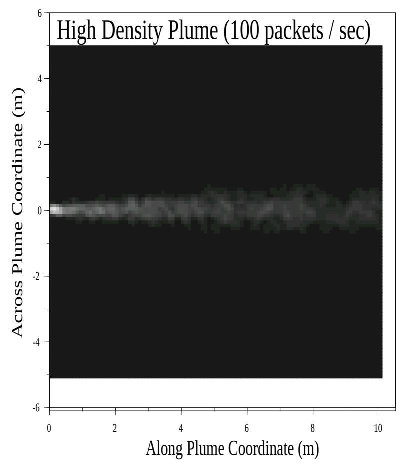


Figure 5.3: A simulated plume with 100 packets/second released into the simulated wind.

The main question, then, is how the performance under these extreme conditions may be improved by use of a swarm. In the next section, we illustrate the sensitivity limitations of single agents. The intention is to use this as a baseline for measuring the performance of minimally communicating swarms, and of explicitly communicating swarms. Careful examination of these questions may yield strategies for dealing with very sparse plumes, such as those produced by explosives or nonvolatile chemicals.

5.2. Sensitivity Measurements of the Solitary Spiral Algorithm

We choose as a baseline performance that of the solitary spiral algorithm in which a single agent performs a plume tracking behavior independently. In this simple case, the agent, as illustrated in Chapter 4, will be completely self sufficient, neither using information from other robots, nor affecting other robots.

Our measures of performance are scaled performance measures, giving the relative performance of the robot using the given algorithm to one with perfect knowledge which may turn directly toward the target and move right to it. These are defined by

$$TDF = \frac{t_{min}}{t_{actual}} \tag{5.2.1}$$

and

$$DDF = \frac{d_{min}}{d_{actual}} \tag{5.2.2}$$

where t_{actual} and d_{actual} represent the time and distance expended and travelled, respectively, and t_{min} and d_{min} represent their minimal counterparts.

Moreover, we report on the arrival times and distances of the first agent to arrive given a particular plume concentration and number of agents. As we increase the number of solitary searchers, we expect this performance to improve, on average. The amount that it improves as well as the numerical values of these average runs provides us with a view of how different methods of communication may be exploited to more quickly locate an odor source.

Finally, we report on the performance of the group of robots. For each parameter setting, we report on how many robots make it to the source in a span of ten seconds. All robots are initiated in a random initial position centered around twelve meters downstream. As the plume extends only ten meters, the robots must first locate the plume, and then track it upstream. The robot is considered to have arrived if it is located to the left of the origin and is between -1 m and 1 m from the origin. It must also have detected some odorant in order to have arrived. A sample track is given in Figure 5.4.

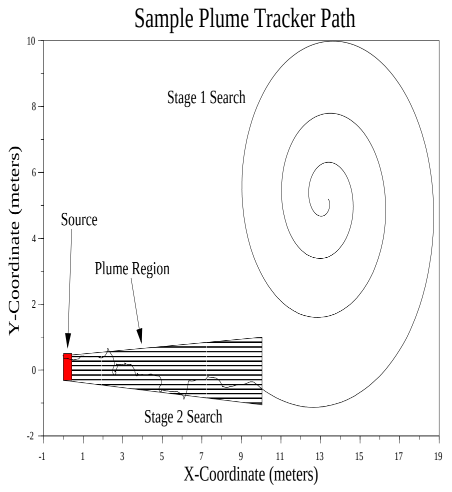


Figure 5.4: This is a typical robot track, illustrating the two regions of search. In the first region of search, the robot is locating the plume. In the second region of search, the robot is using the plume to find the plume source.

The robots have a maximum speed of 17.0 meters per second, an angular resolution on the wind sensor of 90^o, noise on each of the two wheel motors causing a variation in the speed equalling up to 10% of the commanded speed, and noise on the wind direction sensor with a guassian distribution and standard deviation 45^o. This robot configuration is used throughout this paper.

We first observe the success rate of the group of robots in obtaining the source. Figures 5.5-5.7 give the sensitivity curves for three group sizes.

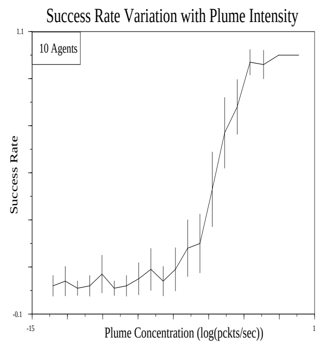


Figure 5.5: This figure gives the success rate of 10 robots searching for the source of the plume. The turning point, located at a density of ˜1.8 packets per second (−4 log scale) is the turning point of sensitivity for the robots, after which more than half the maximum performance of the robots is obtained.

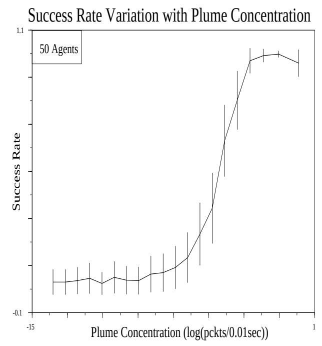


Figure 5.6: This figure gives the success rate of 50 robots searching for the source of the plume. The turning point is located at a density of ˜1.8 packets per second (−4 on this log scale). This performance is identical to that given above. This is to be expected, as each robot is performing the same behavior.

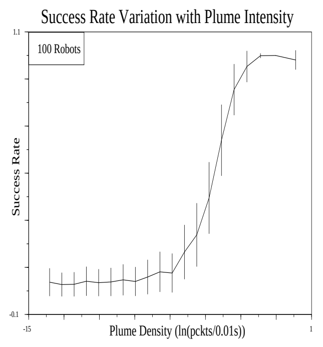


Figure 5.7: This figure gives the success rate of 100 robots searching for the source of the plume. The turning point is located at a density of ˜1.8 packets per second (−4 on this log scale). This performance is identical to that given above. This is to be expected, as each robot is performing the same behavior.

These figures illustrate a design limitation of the basic spiral algorithm. This limitation is that at densities lower than emmissions of 1.8 packets per second, fewer than half of the maximum number of robots will reach the source. Interestingly, the percentage of robots arriving does not drop absolutely to zero at lower concentrations. This means that sometimes the robot is capable of finding its way to the source despite the low concentration. This is illustrated in Figure 5.8.

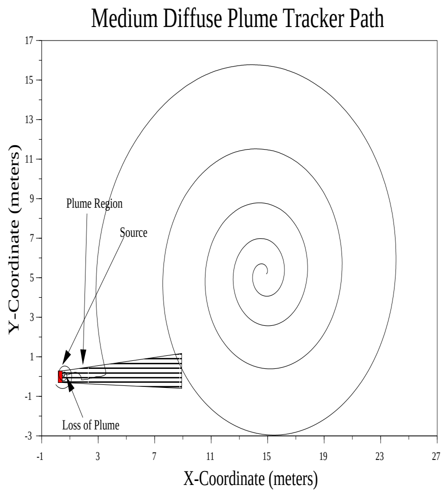
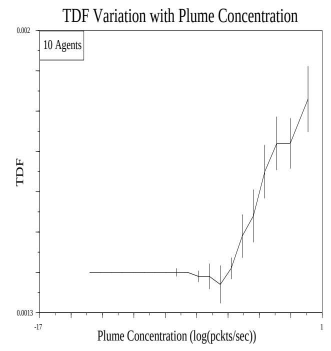


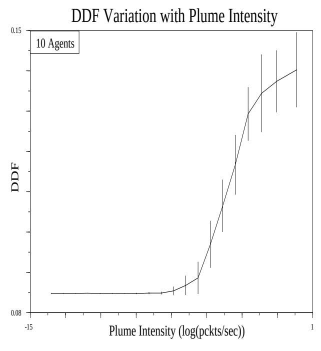
Figure 5.8: In this figure the trail a robot follows in tracking a plume to its source is illustrated. The robot in question encounters a packet, travels upstream for a short distance, and initiates a second spiral. This second spiral is successful in finding the source of the plume.

The probability of this occurring is not zero, but it typically occurs near the source, and requires only a small final search for the source.

We may investigate the TDF and DDF performance measures to obtain a more complete view of the group performance. These are given in Figures 5.9-5.11.

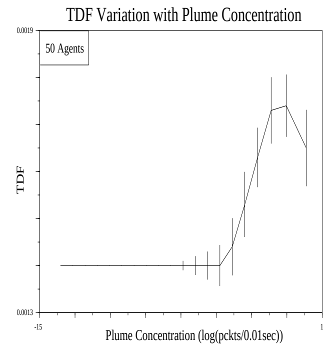


(a)

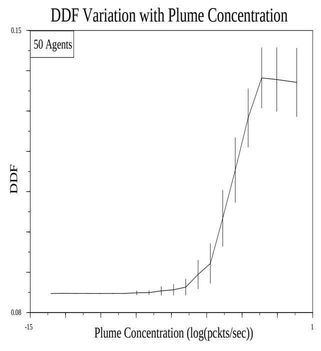


(b)

Figure 5.9: This gives the TDF (a) and DDF (b) performance measures for a group of ten noninteracting robots searching independently. The small TDF value indicates that the robot moves at a net speed that is much lower than the maximum speed. However, the value of DDF is much larger, indicating that the waste is not in the distance travelled, but rather that the average speed over the entire track is small. Note the turning point is at a higher concentration than that given above. This would seem to indicate that the point at which the time and distance-based measures turns over is at a higher concentration than that at which the robots are merely likely to find the source in the specified time. The effect would seem to be identical to that of reducing the search time from 10 seconds to some smaller number.

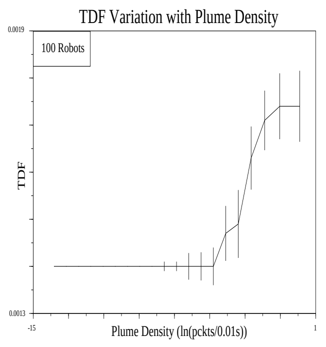


(a)

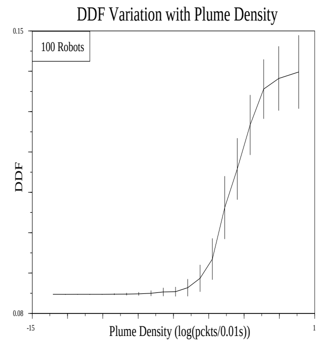


(b)

Figure 5.10: This gives the TDF (a) and DDF (b) performance measures for a group of fifty robots. The performance is very close to that of the 10 robot case.



(a)



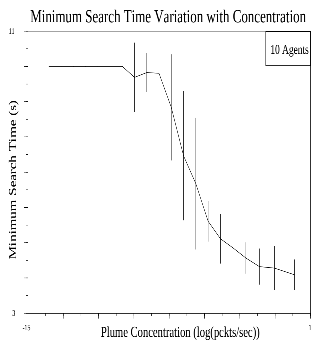
(b)

Figure 5.11: This gives the tdf (a) and DDF (b) performance measures for a group of one hundred robots. The performance is also very close to that of the 10 robot case.

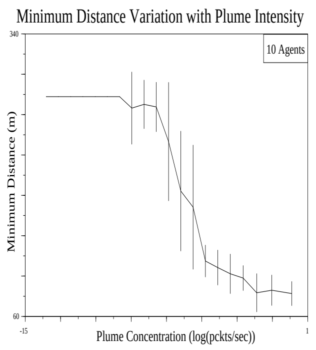
The small value of TDF in Figures 5.9-5.11 indicates that the robot tends to move at a net speed that is much lower than the maximum speed. However, the much larger value of DDF indicates that the waste is not in the distance travelled, but rather that the average speed over the entire track is small. One would expect this in a spiral-based algorithm, as linear progress in any particular direction is progressively slower as time passes and each loop becomes larger. Note the turning point, or point of inflection, in the TDF and DDF graphs is at a higher concentration than that given in the success rate graphs. This would seem to indicate that the point at which the time and distance-based measures turns over is at a higher concentration than that at which the robots are merely likely to find the source in the specified time.

Note that the performance measures do not change with group size. This is expected, as the performance of any one agent, upon which the performance measures are based, is not dependant on the size of the group. Thus, the average performance should be fairly constant across group sizes, tending to a finite limit as the group size increases.

Finally, we report the minimum distance and time to source location as a function of the group size.



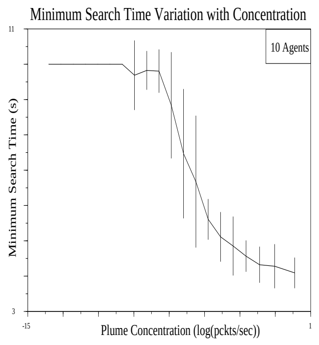
(a)



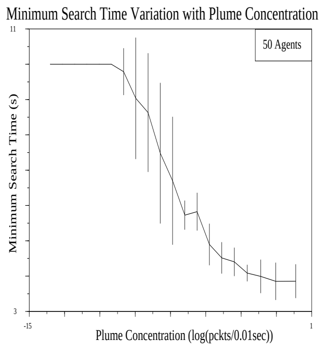
(b)

Figure 5.12: The minimum search time (a) and distance (b) are monotonically decreasing functions of plume concentration.

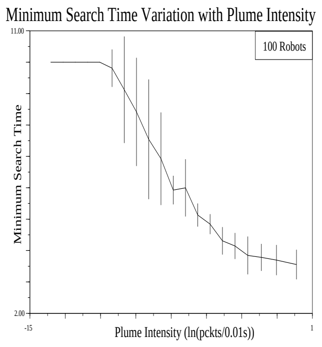
As one might expect, this is a monotonically decreasing function of plume concentration. However, this reflects only the fact that any agent is more likely to interact with the plume when it is dense than when it is sparse. Another expected characteristic of the performance is that each of these measures is a decreasing function of group size (which we illustrate with the minimum search time and minimum search distance in Figures 5.13 and 5.14). This simply reflects the greater probability that at least one member of the group will interact with a plume when the group is large than when it is small.



(a)

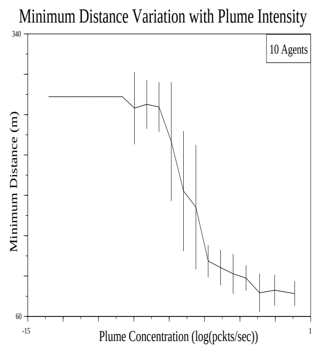


(b)

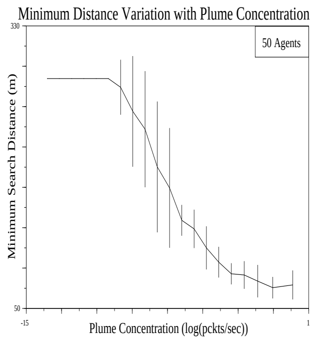


(c)

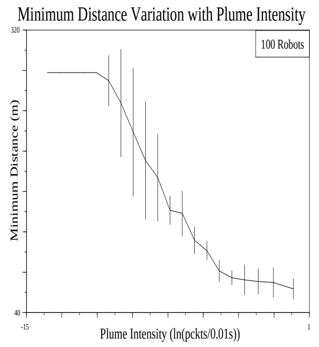
Figure 5.13: Both the turning point and absolute minimum of the arrival times are decreasing functions of group size, as is illustrated here with three minimum time paths.



(a)



(b)



(c)

Figure 5.14: Both the turning point and absolute minimum of the minimum search distance are decreasing functions of group size, as is illustrated here with three minimum time paths.

5.3. Swarm Engineering in Plume Tracking

Swarm engineering is the process of designing a system in which the agents of the system have a simple behavior which when combined produces a global behavior of interest. This behavior need not be an explicitly designed behavior of the system, though swarm engineering does not preclude this. Global group behaviors not explicitly designed in the creation of a system are *emergent behaviors*. We now turn to the investigation of the use of formal swarm engineering techniques in the generation of robust plume tracking behaviors capable of locating plume sources of diffuse odor plumes.

5.3.1. Implicit Communication. We define the formal method of swarm engineering by two steps.

1. Generation of an appropriate group based swarm condition.
2. Generation of a behavior to be taken by each of the elements of the system which satisfies this condition and generates the appropriate behavior.

The combination of these two steps can be expected to yield a desired global behavior.

In Sections 5.1 and 5.2, we examined a plume tracking algorithm that may be implemented on a single robot. This algorithm has a number of attractive qualities including the ability to track plumes that are nearby or far away, as well as the ability to track nearly directly upstream in the presence of a well defined plume. The main weakness in the algorithm is the position of its point of inflection, a rather high concentration.

If we examine the behavior of a large number of agents carrying out the single plume tracking behavior we initially note that without any implicit or explicit communication, the robots tend to cover area in their search that other robots might have already gone over or be going over at the same time. This leads to inefficient searches of large areas, characterized by Figure 5.15.

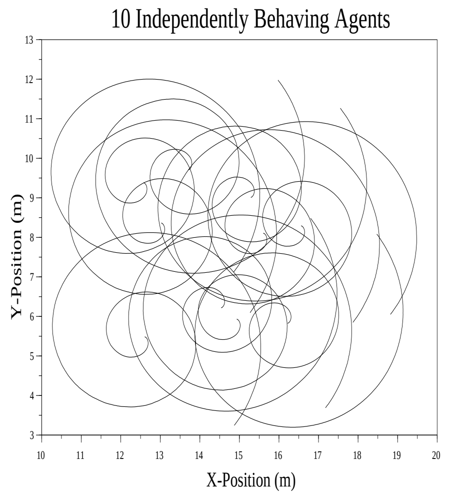


Figure 5.15: This figure gives the behavior of ten robots carrying out a general plume tracking algorithm. Most notably, the paths overlap greatly, reducing the efficiency of the search.

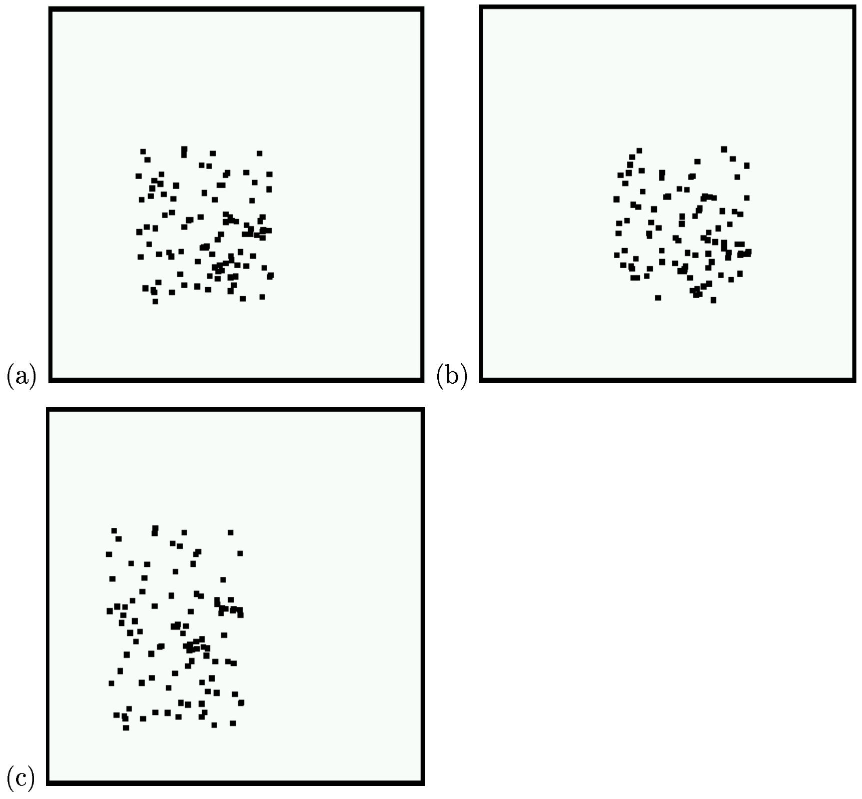


Figure 5.16: This figure gives three screen shots of subsequent plume tracking behaviors of fifty robots. Note that these points are highly clumped, and the robots seem to stay within the same area. This clumping behavior is highly undesirable.

One simple correction to this problem might be to initially distribute the robots around the search space. However, in open-ended search algorithms, this is not possible, as the search space is not known ahead of time. Any attempt to canvas the space would amount to nothing more than increasing the starting area, simply reducing the speed at which this fundamental problem needs to be dealt with.

We thus have a swarm condition. We require a swarm behavior that will increase the search space of the set of robots with only a minimal change to the algorithm. This allows us to more adequately cover large spaces, and to find the source quickly.

> **Swarm Criterion**: Local behavior must increase the search area, reducing the overlapping search spaces.

We now investigate one simple algorithm which will satisfy this swarm condition.

> **Swarm Design**: Robots located in high robot density areas will seek low density areas. Robots in low robot density areas will execute WBSA.

This simple design modification adds one level of complexity on top of the simple wind-based plume tracking algorithm discussed above. In this strategy, the robots execute an avoidance behavior when in the presence of many other robots. This is implemented in the following way: each robot has a beacon which is always on, and broadcasts to other robots. Each robot is aware of the signal and signal direction of the other robots. The broadcast intensity drops off as a function of the square of the radial distance between any two robots. If the sum of the broadcast intensity is above some threshold, the robot calculates its centroid, and moves directly *away* from the centroid as fast as possible. If the sum is below that threshold, the robot will carry out a simple plume tracking behavior. Figure 5.17 illustrates this behavior[1].

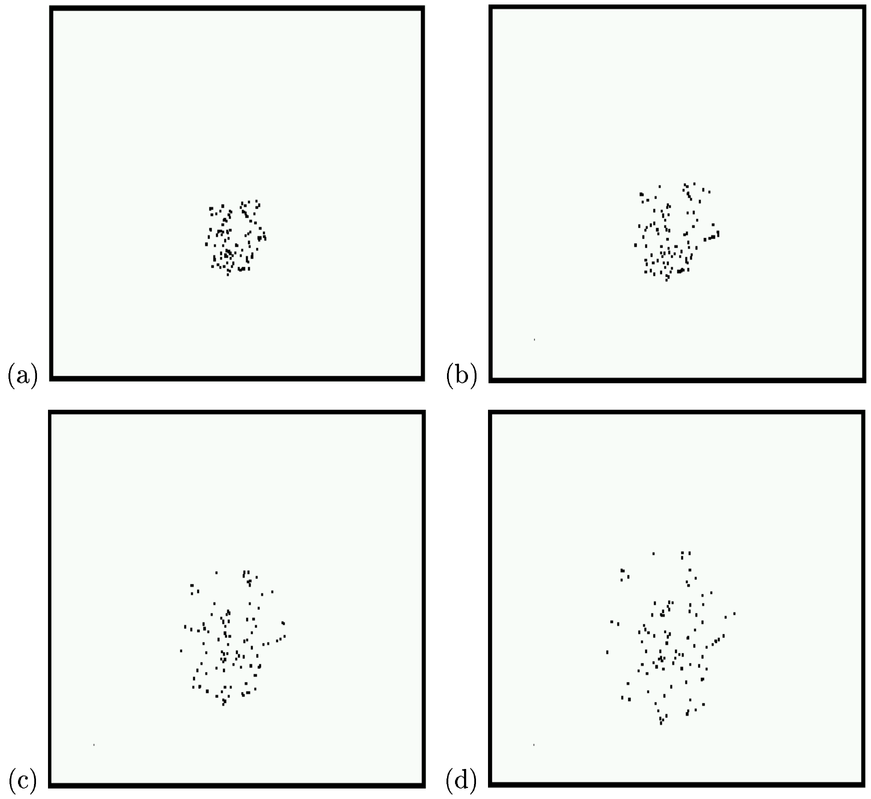
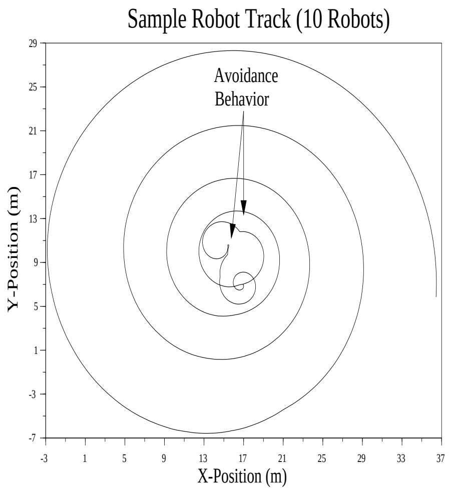


Figure 5.17: This gives the behavior of a self-avoiding swarm of robots. Each snapshot is separated by 0.3 seconds of simulated time.

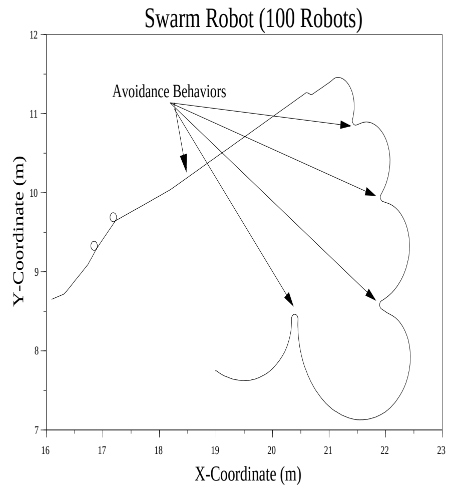
Note that this behavior need not be mediated by a beacon on each of the robots. Rather, it may be mediated using visual sensors, obstacle avoidance sensors, etc.

This algorithm produces robot paths similar to those given in Figure 5.18.

[1]See also distributed search, (Gage 1995).



(a)

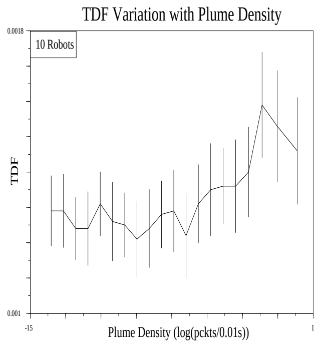


(b)

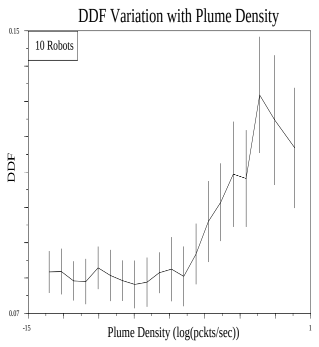
Figure 5.18: The behavior of the robots is quite different when utilizing the avoidance behavior in the presence of a small number of robots (a) and in the presence of a large number of robots (b).

Note that these behaviors are quite different. The behavior in Figure 5.18 (a) is that of a typical robot in the presence of a small number of other robots. In this case, a minor amount of avoidance is necessary. However, in the presence of a large number of robots as in Figure 5.18 (b), the behavior begins to be dominated by avoidance and exploration of what seems to be space on the fringe of the swarm. We can therefore expect large differences in performance by our performance measures in these two cases.

If we examine the behavior of the swarm in the small swarm case, we find comparable behavior to that found in Section 5.3.



(a)



(b)

Figure 5.19: The behavior of the TDF performance measure and the DDF performance measure is very comparable to the same measures given in the independently acting agent cases with similar group sizes.

The TDF and DDF performance measures given in Figure 5.19 correspond well to their counterpart measures in the independently acting robot case. At these levels of robot density, the effect is rather minimized. Interestingly, the range of values over the entire range of plume densities is slightly smaller than the same in the noncollaborating case. Moreover, the entire graph is actually shifted downward, in comparison to the noncollaborating case. This would seem to indicate that the collaboration impedes the search of at least one agent, and this impedance is sufficiently strong to offset any possible benefit. Indeed, from these two graphs, it is hard to see how there might be any benefit at all.

The performance measure of success rate is also quite comparable to the previous data sets, exhibiting comparable turning points and measures.

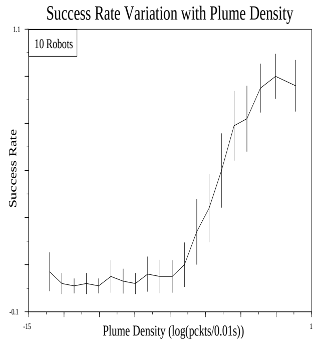
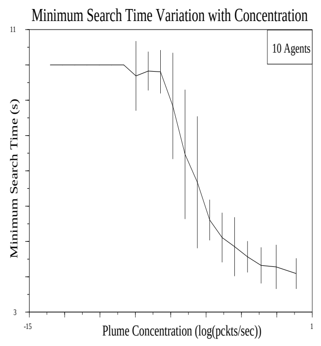
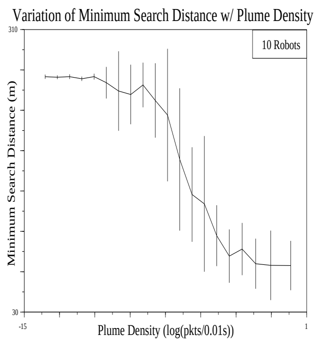


Figure 5.20: The performance measure of the success rate is also quite comparable to the same in the case of a small number of robots in the previous model.

Finally, (Figure 5.21) for a small number of robots, the minimum arrival time and minimum arrival distance seem not to be strongly affected.



(a)



(b)

Figure 5.21: These data for the minimum search time and minimum search distance for the avoidance low density swarm are quite similar to that given above in the non-swarming case. However, both data sets are shifted down a bit from the independent agent data. This indicates a positive effect of the swarm strategy is that the first agent will tend to arrive in a quicker time and with a smaller overall time.

Thus the data are comparable to that given above.

The situation radically changes as we increase the number of robots. This is to be expected from the difference in the trajectories of the robots. As the robots become increasingly mutually disruptive, their ability to explore becomes severely curtailed. This means that robots that are near the source may not find it because instead they are moving away from other robots. While the expansion will help a select few to find the source, this may be a small proportion of the group of robots. We see this reflected in Figure 5.22.

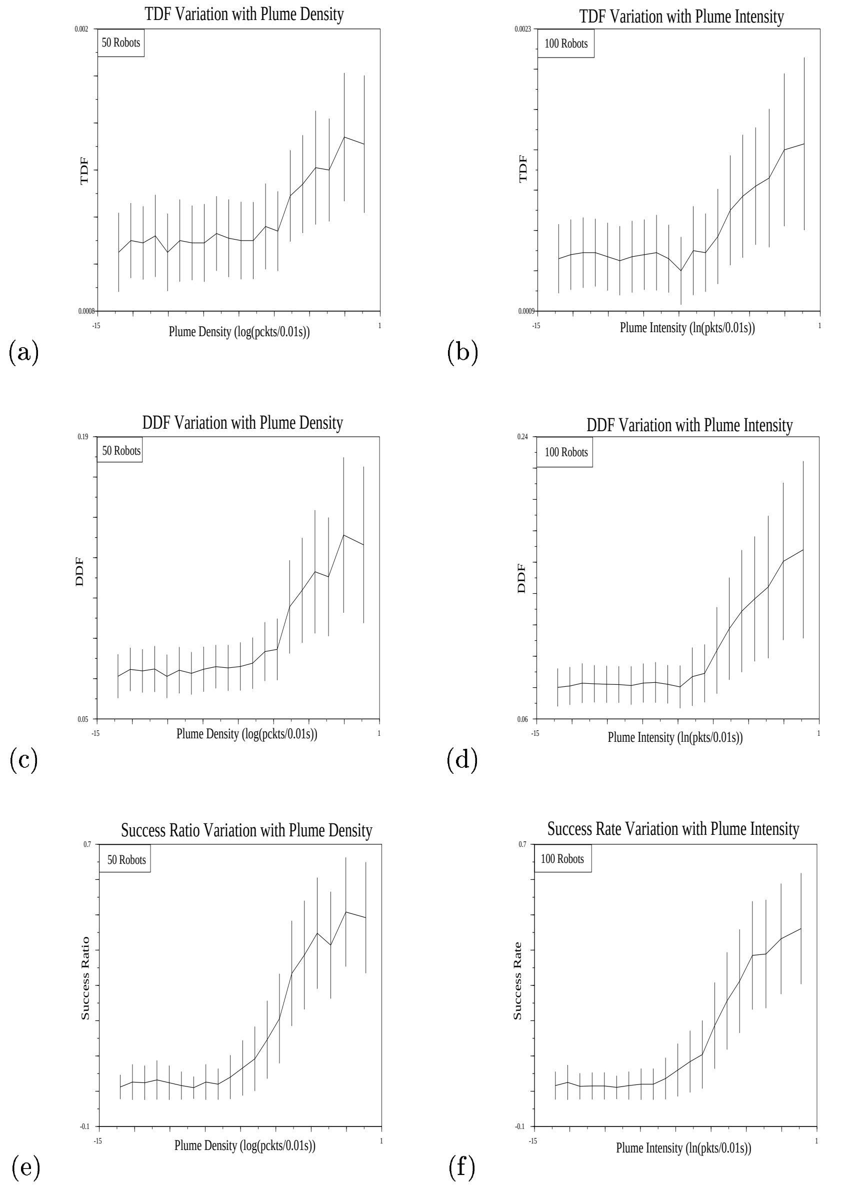
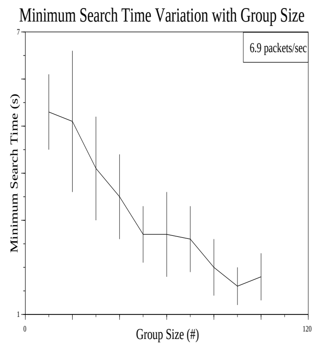
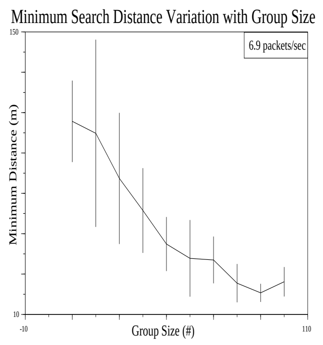


Figure 5.22: The behavior of the TDF performance measure ((a) and (b)), the DDF performance measure ((c) and (d)), and the success ratio performance measure ((e) and (f)) degrade sharply with increasing group size. This is the effect of the increased avoidance time in the spiral search algorithm.

One interesting effect, however, is that the minimum search time does not increase, but rather decreases, as does the minimum search distance. This is the average time and distance used by the first robot to obtain the target. This time is expected to improve as the effect of the group interaction is the increase in the size of coverage of the group by many times, as is apparent in Figure 5.23.



(a)



(b)

Figure 5.23: This figure illustrates the benefit of adding a deliberate interaction behavior. If the goal is to get one robot to the source in the minimum time, these data illustrate the the interaction behavior provides a mechanism to allow the swarm to obtain the minimum in a smaller time and distance than with non-interacting robots.

In very large fields, the interaction term of small mobile robots may be effectively zero by design. This line of inquiry indicates that it behooves a designer of a robot search system to implement an avoidance behavior of the scale of interest of the problem. Moreover, if the size scale of the plume is much greater than that of the robot, a deliberate avoidance behavior may be beneficial to the performance of the swarm.

5.3.2. Explicit Communication. Aside from the ability of the preceding swarm algorithm to find the source of the plume more quickly and using a searcher which requires less time, the performance was not particularly promising. All of the performance measures recorded a reduction in functionality as the group size increased. We regard this as a failure of the swarm design condition, as it seems not to have accounted for the increased time and effort requirements of the swarm. Thus, it would seem to be necessary to alter the swarm requirement to allow methods of overcoming the difficulties.

> **Swarm Criterion**: The swarm must, at the individual agent level, use the spiral search algorithm. It must expand in areas of little or no density of odorant, and contract in areas of large density, while making progress up the plume in measurable odorant density areas.

This condition does a number of things at once. First, it requires that the swarm be capable of implementing a standard WBSA at the individual level. In the absence of any other agents, a single-agent swarm will be virtually indistinguisable from any other single agent. Next, it requires that the swarm be a space-filling entity in areas of low odorant density. This behavior would seem to allow the swarm to efficiently fill a space and discover any source of odorant or plumes. The swarm must contract upon a plume or odorant source in order to concentrate resources in areas where the odorant may be found, but also be sparse. Finally, the swarm must progress upstream. This precludes any kind of communication that brings the robots back downstream.

We implement these conditions by making a few addendums to the previous low-level behavior described in Chapter 4.

> **Swarm Design**: Robots located in high density areas will seek low density areas. Robots in low density areas will execute WBSA. If encountering an odorant, a robot will turn on a beacon to call other robots, and continue to execute WBSA. Those robots downstream from the emmitting robot(s) will converge on that (those) robot(s). Convergence behavior will take precedence over expansion behavior. The beacon will stay on for a predetermined period after the loss of the signal.

This algorithm satisfies the design requirements of the swarm, and we now turn to an analysis of the swarm behavior.

Firstly, the algorithm uses the same expansion protocol as given above. This performs identically, and so will not be revisited.

As in the first swarm design, we expect the performance on the small data sets to be similar to that of the independently acting robots.

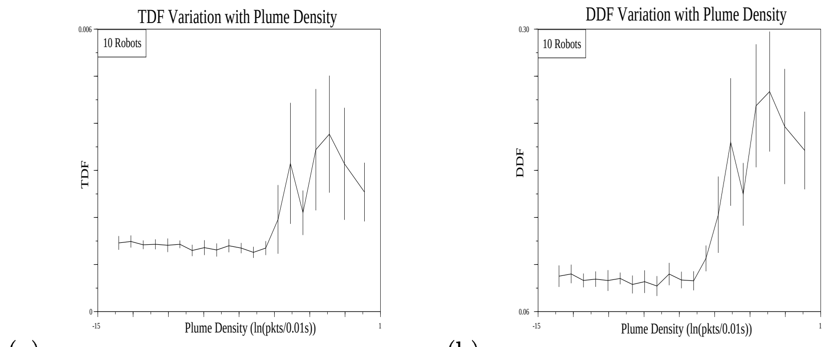


Figure 5.24: The small group (10 robot) behavior of TDF and DDF is similar to that of the independent robots, and the noncommunicating swarms. However, the range of TDF is roughly twice that of the other counterparts, changing from 0.00015 – 0.0016 to a range of 0.0015 – 0.0050. Likewise, DDF transforms from a previous range of approx. 0.08 – 0.12 to a range of approx. 0.08 – 0.24.

While the behavior is indeed similar to that expected, the range of performance measure values is greater than previous performance measure values (Figure 5.24). We can understand this result in the following way. The robots which interact with the plume will tend to redirect other searching robots to the plume. This means that these robots will be more likely to quickly interact with the plume. On average, this is a nontrivial effect, and creates the shift in the TDF and DDF performance measures even at this small number of robots.

The success ratio performance measure does perform as expected. However, the number of successful robots approaches 100% on average more quickly than either of the other two models. More concretely, the inflection point is toward the lower end of the concentration measure than the other two.

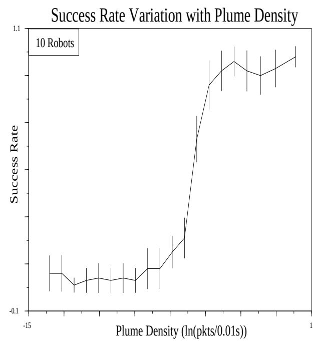
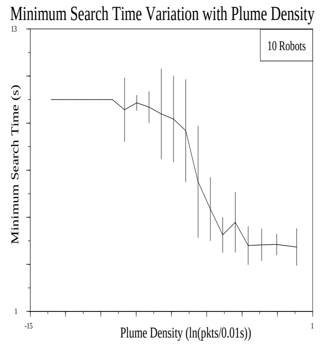


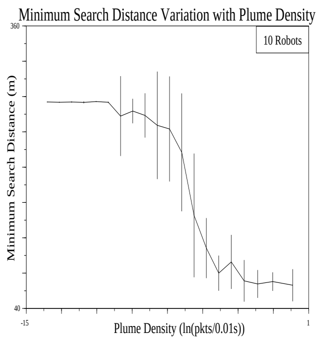
Figure 5.25: The behavior of the success rate performance measure is very similar to the other two models. However, the inflection point, located at -5.5, is at a lower concentration than that of the other models. This reflects the recruitment capability, yielding both agents which may find the plume, and agents that will recover losses of the plume.

Again, this is expected, as the recruitment both aids in the search for the source, and helps in the recovery of a lost plume.

The minimum arrival time and minimum arrival distance do not have any significant different character than their counterparts in the noncommunicating model. There is a slight improvement in the minimum distance, as one might expect since the swarm will expand. However, this is minimal.



(a)



(b)

Figure 5.26: The behavior of the minimum time and distance to the source in the small group collaborative model. This performance is nearly identical to that of the other two models presented earlier in this paper.

Thus, as expected, the behavior of the small group is nearly identical to that of the previous models.

The performance of the swarm changes greatly when the swarm size increases. Firstly, we examine the performance on the TDF and DDF performance measures. These are given in Figure 5.27.

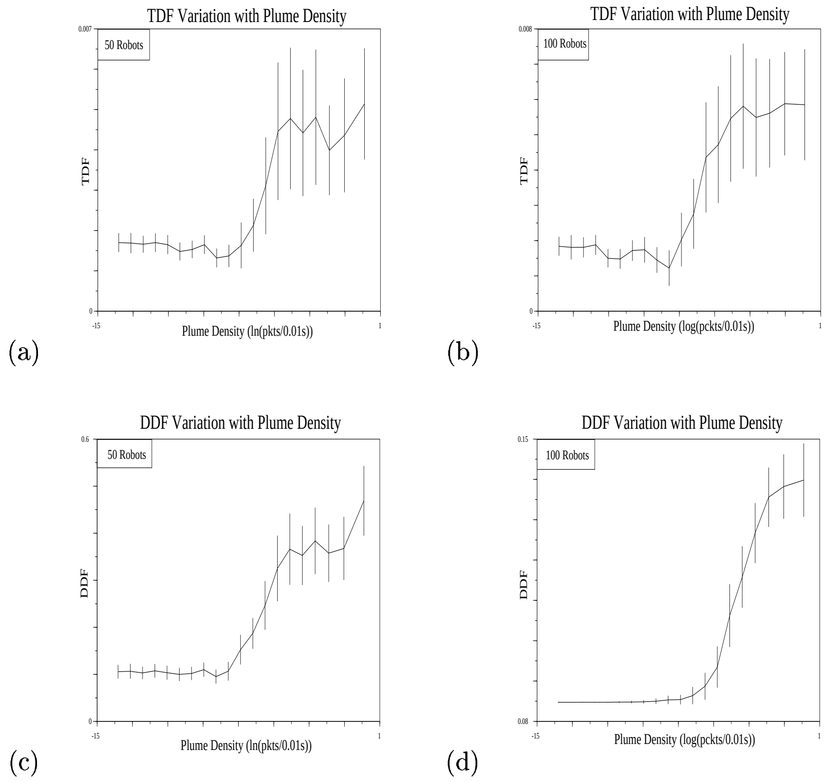
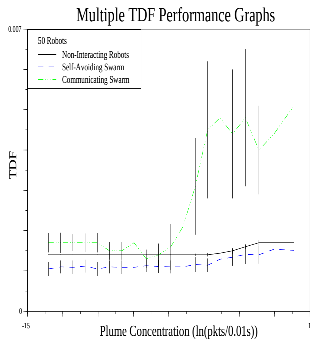
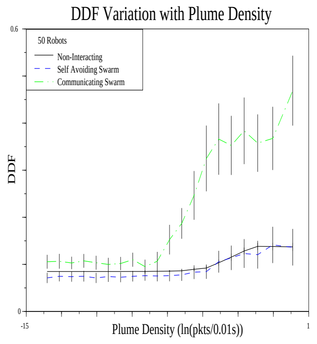


Figure 5.27: These are the TDF and DDF performance measures for 50 and 100 robots. The performance measures improve as the number of robots increases, with the both measures shifting to the left and up when going from fifty to one hundred robots.

In contrast to the performance of the previous two models, the performance improves as the number of robots increases. The improvement in performance is in addition to the overall improvement in these performance measures over their counterparts for the other two models. Numerically, the range of TDF for 50 robots is between 0.0013 to 0.0051, whereas its counterparts in the noninteracting and self-avoiding swarm models have ranges of 0.0014 to 0.0017 and 0.001 to 0.0015, respectively. Thus, while the lower end of the performances is comparable, the higher end is significantly different from the other two models. Similar results are apparent for the DDF performance measure. In this case, the non-interacting robots have DDF values ranging from 0.08 to 0.14, while the self-avoind swarm has DDF values ranging from 0.07 to 0.13. The communicating swarm has DDF values ranging from 0.11 to 0.49. (Note that 1.0 is completely impossible in this model, as described above, since the robots must first find the plume.) We plot comparative measures for 50 robots in Figure 5.28. Similar performance has been measured for larger numbers of robots, but is not presented here.



(a)

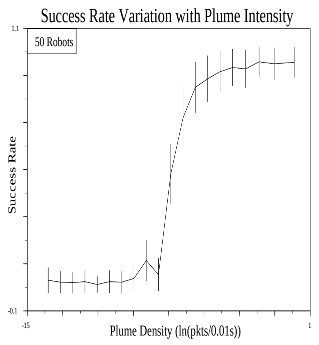


(b)

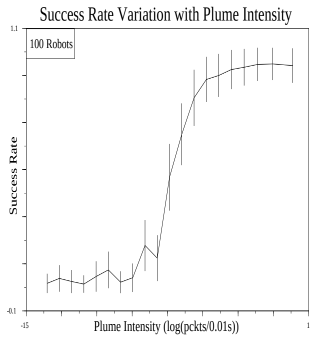
Figure 5.28: These give comparative performance measures for the three different models. Notably, the swarm has not only a better performance overall, but also a significantly better sensitivity, as measured by the inflection point of the curve.

The most important result here is that the performance of the swarm is significantly better than that of the other models. This improvement in performance is not restricted to the shift upward of the graph. It is also in the shifting to the left of the graph. This means that the point of inflection of the graph, which is a measure of the sensitivity of the swarm to the plume, is also shifted to the left. This indicates that some plumes which are not capable of being reliably located by single robots or self-avoiding swarms may be reliably located by this model of communicating swarm. This is an important result, as it indicates a significant benefit in the use of a swarm over multiple instantiations of a given robot.

Our next performance measure is the success rate of the algorithm. In Figure 5.15, we plot the success rate of the swarm for 50 and 100 robots.



(a)



(b)

Figure 5.29: These are the success rate performance measures of the swarming algorithm. Both the 50 robot and 100 robot are sharply increasing at a concentration of -4.5 (log scale). The performance of both algorithms is very similar, indicating that the sensitivity increase occurs by the swarm size of 50 robots.

In this graph, we find that the performance is has its inflection point at approximately -4.5 on a log scale of plume intensity. Moreover, the performance of the 50 robot swarm and that of the 100 robot swarm is extremely similar, and to experimental error, are overlays of one another.

One interesting attribute of this data is that the perfomance measure is not identically one at plume high concentrations. This is contrary to intuition. Further investigation reveals a slight error in the collaboration, which causes this unexpected result. This error causes

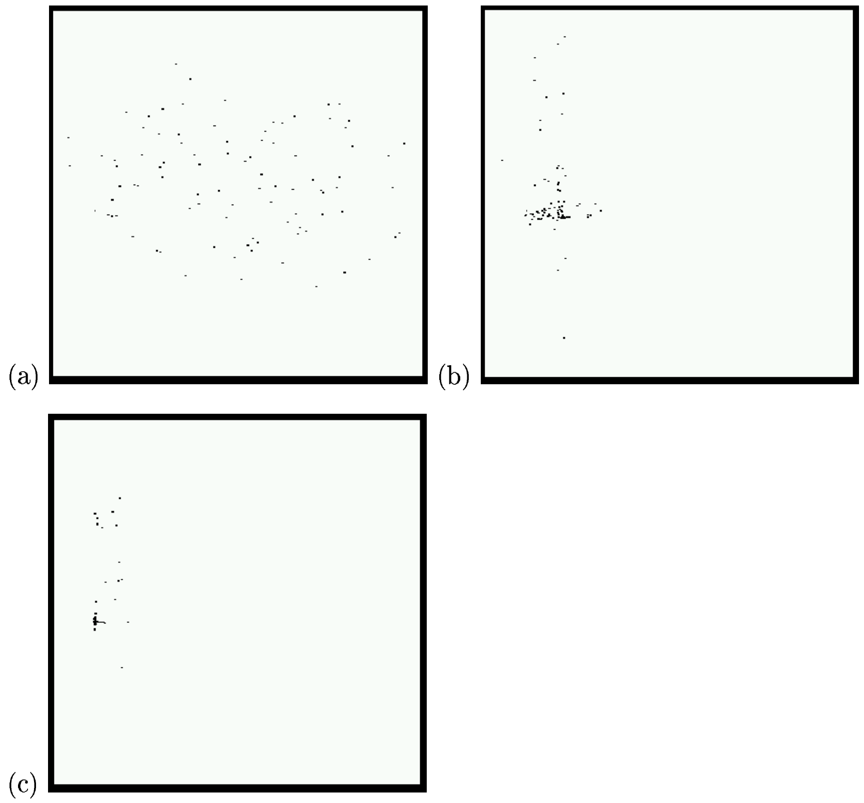


Figure 5.30: These screenshots of the swarming process provide an explanation for the variance in the performance. Some of the robots responding to the recruitment signal find their way to the source (b), but then wander away from the group (c).

some of the robots to wander away from a group of robots despite their effective recruitment. This is a correctable error, but has been left in so that we may illustrate the strength of the swarming approach.

Comparison of the performance of the performance on this performance measure with those of the non-interacting and self-avoiding swarm models yields a view of how much of an advantage this swarm has over the previous plume tracking model.

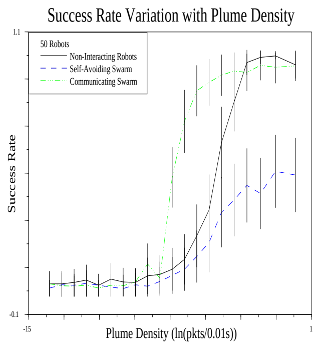


Figure 5.31: This is a comparative plot of the different performances on the success rate performance measure. Most notably, the performance of the communicating swarm has an inflection point significantly ahead of the other two models.

The inflection point of the performance measure of the communicating swarm is significantly to the left of the other two models. The difference in position is two log levels, indicating a significant difference in concentration between the points at which the self-avoiding swarm and non-interacting robots become reliable. This means that, as a reliable plume tracking entity, the swarm is more reliable at significantly lower concentrations than the other two models.

Finally, as we would expect, the minimum distance to the plume source and the minimum time to the plume source decrease as the number of robots increases. However, what is not expected is that the plots also shift leftward to lower concentrations. This behavior was not evident in previous models.

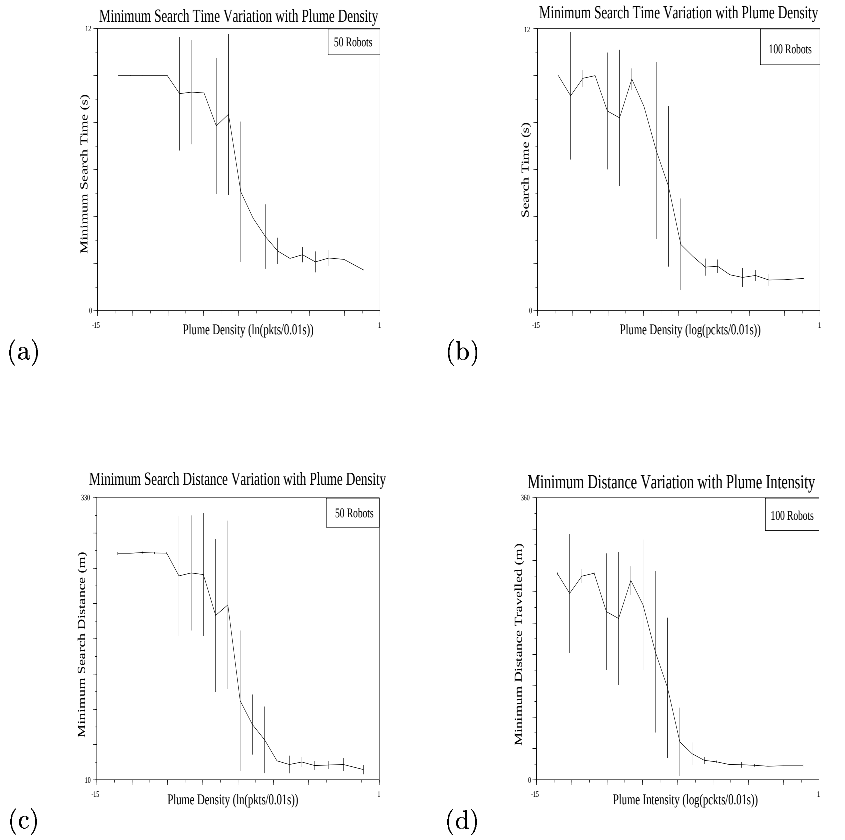
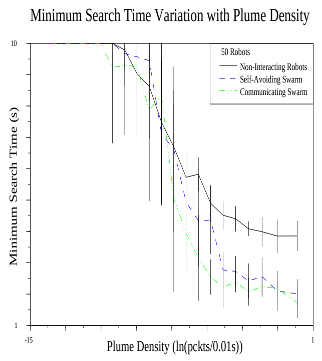


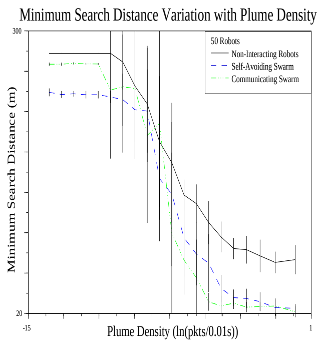
Figure 5.32: These plots give the average minimum arrival times and distances of robots in the communicating swarm model. The plots of a larger number of robots are shifted down and to the left of their 50-robot counterparts, indicating a secondary effect which helps robots to achieve the source initially.

This indicates that there is some degree of help in achieving the plume at low concentrations. In other words, one agent may be able to find some odorant, and recruit others who eventually find the source. This form of aid has been seen elsewhere (Hayes et al. 2000) and is the subject of study.

A comparison of the three models on these last two performance measures indicates that there is a no initial advantage, but that as the group size grows, the advantage increases.



(a)



(b)

Figure 5.33: These plots give a comparative view of the minimum search time as a function of plume density for the three models. Initially, all three are equivalent, in the low concentration limit. However, as the plume density increases, the the performances diverge, with that of the communicating swarm eventually winning out over the other two, though with no significant difference from the self-avoiding swarm at high concentrations.

Moreover, the performance of the communicating swarm eventually overtakes that of the self-avoiding swarm, again indicating that some second order effect of the communication is at work.

5.4. Discussion and Concluding Remarks

In previous Chapters we noted that the only information required for plume tracking is the direction of the wind and the local concentration of odorant. A swarm will add little information about the direction of the wind, but may be used to concentrate search effort on a location that has a verified likelihood of containing the source. What we have illustrated is that a swarm of interacting robots is capable of tracking plumes of very much smaller density than those that can be tracked by either independent or semi-independent swarms.

The distinction between tracking with swarms and tracking with independent or semi-independent swarms is important. It is, as our data would seem to indicate, not sufficient to cover an area with robots. Rather, the robots must share information about the area, and use this to focus the search. This means that the swarm, viewed as an entity, must have the ability to react to local conditions, and to rearrange itself based on these conditions. In this way, the swarm may accurately locate the source of the plume.

In fact, these requirements are exactly the requirements we specified at the outset of our swarm engineering effort. It would seem, then, that although we have not been able to analytically express the conditions required for swarm-based plume tracking, it was sufficient to state them clearly, and to generate a simple microscopic behavior that produced the desired macroscopic effect. In the next two Parts of this Thesis, we shall examine systems for which we can produce an analytic description of the desired swarm characteristics. However, it is worth noting that the two processes, that of generating an analytic expression or set of expressions to describe the system, and that of generating an empirical understanding of how the system works, are actually equivalent, in that their final goal is a concise description of the system. The key in both efforts is to produce a condition on the individual agent which, when satisfied, will generate the desired group behavior. This exactly what we have done, in a completely anecdotal way, determined by our experience with the problem generated in the work presented first two Chapters of this Part.

Part 2

Puck Clustering and Distributed Construction

CHAPTER 6

Introduction to Puck Clustering

6.1. Introduction to the Part

The past few years have seen a great deal of work on autonomous robots which cluster objects (Holland 1996, Franks et al. 1992, Karsai and Penzes 1993, Bonabeau et al. 1998, Camazine et al. 1990, Deneubourg et al. 1991, Melhuish and Holland 1996, Beckers et al. 1994, Maris and Boekhorst 1996, Kazadi 1997). In such a system, objects are manipulated by individual agents which are endowed only with local interaction rules. Despite this apparent deficiency, the agents in question are capable in many cases of causing clusters of objects to emerge, and eventually, to generate a single cluster. These systems exhibit *emergent behavior*, as their global behavior depends on the emergence of group behaviors not explicitly designed at the agent level rather than the explicit cluster-building design. This indicates that simple behaviors may be combined in such a way as to produce desired global behaviors, one of which might eventually be rudimentary construction behavior. One would like to understand what the minimal requirements of a system might be in order to generate a *predetermined complex global behavior*. Once these minimal requirements have been established, the system design may be done in any of a number of different ways that leads to the desired global behavior.

This Part develops a series of theoretical arguments which propose a set of conditions under which a system will result in the collection of all pucks into a single cluster from any set of initial conditions. We model the system under simple rules and ask what these rules imply about eventual equilibrium points. The movement of a puck from one cluster to another is modelled as a transport process which is mediated by the robots. In our model, robots are constrained to carry only one puck at a time. Therefore, a robot is able to deposit its single carried puck into a cluster or, if empty, is able to remove a puck from a cluster. If the deposit or removal of a puck is viewed as a random process, the probability that a puck is deposited in a given cluster is dependant on the number of robots in the system carrying pucks and the effective cross section of a cluster to the puck-carrying robot. Conversely the removal of a puck is dependant on the number of empty robots and the corresponding cross section of a cluster. The system can therefore be modelled as a diffusion process in which each cluster of radius R has different cross sections for puck evaporation $f(R)$ and condensation $h(R)$. The equilibrim point of the system will therefore depend on the nature of these cross sections.

We depart from previous methods of generating puck clustering swarms in an important way. This approach is to determine the condition(s) that must be satisfied in order for a particular final result to occur. Our goal is to illuminate a design requirement for robot systems undertaking this task, rather than to prove that any one single design will complete the task. The approach allows for the creation of many different strategies, all of which may lead to the final desired system. We do not make any attempt to address the design of efficient algorithms, deferring this to another later work.

6.2. Previous Puck Clustering Work

Puck clustering teams are an example of work done in the realm of minimalist design using robot groups and simple actuators (Holland 1996). These systems consist of robots which have very simple controllers including a small amount of memory (if any at all) and rigid heirarchical controllers. This view has gained increasing support as ever more interesting systems are designed using robots of this design paradigm. The allure of these systems stems from the ability to use very simple robots to achieve complex tasks, and obtain behavior that is similar to some intelligent behavior.

Initial biological studies of insect systems (Franks et al. 1992, Karsai and Penzes 1993) indicated that ants and termites are extremely simple creatures carrying out very simple algorithms. The behavior of ants and termites is extremely rigid and specific, yet their nests can be extremely efficient, flexible, and adaptive. Work on systems designed to take advantage of this newly discovered emergent property of systems of rigid agents has uncovered interesting design paradigms governing possible construction techniques and optimal resource exploitation algorithms (Bonabeau et al. 1998, Camazine et al. 1990, Deneubourg et al. 1991, Melhuish and Holland 1996). These studies indicated that simple rules and stigmergic actions were all that were required for the construction of complex structures. This formed some of the motivation behind the robotic test cases in which the first construction task, that of assembling a cluster of pucks, is undertaken.

One of the earliest attempts to design puck clustering robots came from Beckers et al. (Beckers et al. 1994). In this study, a group of robots is equipped with a curved passive gripper in the front of the robot. This gripper is capable of clustering "pucks" and holding them as the robot moves. When three or more are clustered in the gripper, it triggers a backup motion, which leaves the pucks where they are.

Average simulations of three robots take just over an hour and a half to complete. In the final stable state, a single cluster is formed. The system is capable of forming the same stable state with one robot, but this takes a concommittant longer time.

Maris and Boekhorst (Maris and Boekhorst 1996) present a system that is similar to that of Beckers et al. However, in this study, the robots do not have grippers, and so cannot recover pucks that have been lost near a boundary. In their case, pucks are pushed until a barrier (which may be another puck) is encountered, at which time the puck is left behind and the robot turns and goes in another direction. Once a cluster is big enough it becomes stable, as the robot cannot any longer approach the cluster and take off pucks. In this study, many clusters of size one or two were created near the boundaries, though other larger clusters were found in the interior of the arena. There was no global cluster formed, though one or two large ones were formed.

Kazadi (Kazadi 1997) constructed a single-robot system capable of forming clusters of pucks. In this system, the robot was designed with a gripper, similarly to that of Beckers et al., and sensed the pucks with infrared sensors, rather than with mechanical motion. This allowed a smaller perturbation to the system when encountering existing clusters than the design of Beckers et al. The robot itself was designed with all of its actuators and electronics physically located within the circle for which the gripper was a diameter. In this system, the robot clustered the pucks along the walls of the arena, yet avoided losing pucks to the arena walls. A typical robot clustering experiment is depicted in Figure 5.1.

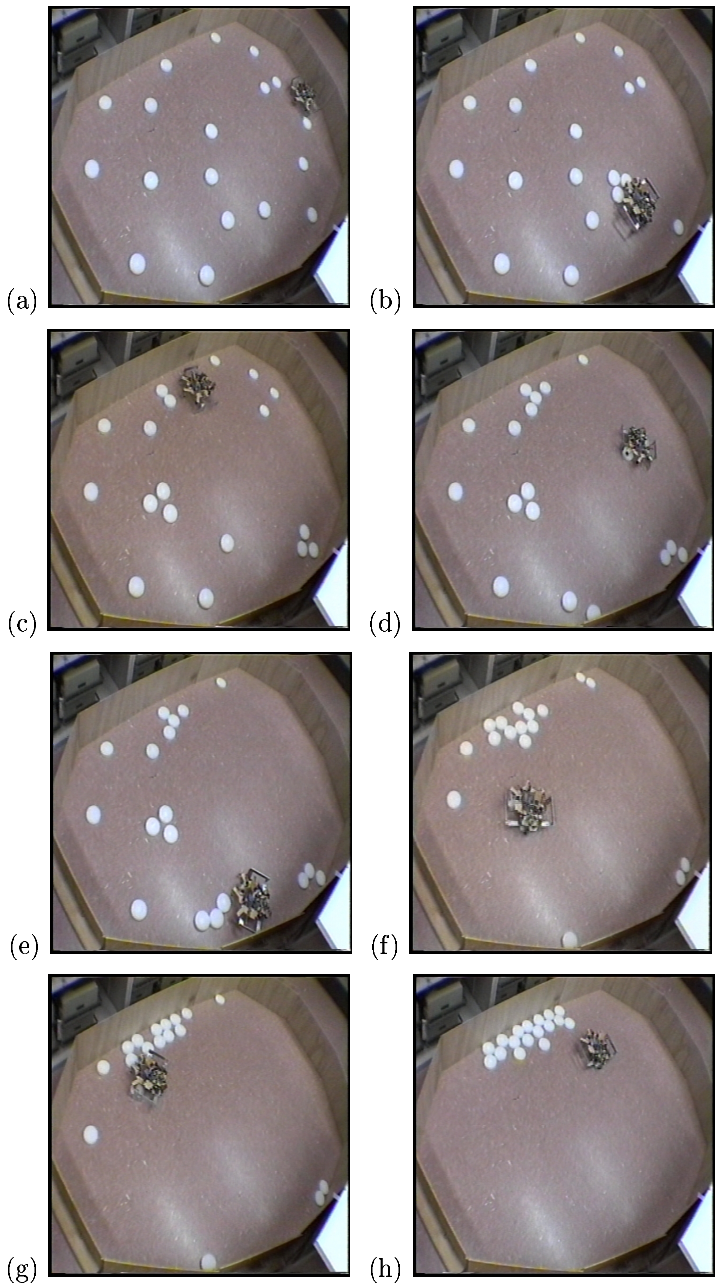


Figure 5.1: A puck clustering robot clustering randomly placed pucks into a single cluster.

We do not consider here the larger general field known as puck collection in which the requirement that the agents not originally know where the pucks are to be collected is relaxed. A large body of work has been done on this, including in applying learning algorithms to the group task (Mataric 1992, 1993, 1994).

6.3. Clustering Definitions

We take a moment here to make a few definitions. These definitions will help us to focus our discussion on a specific set of systems, rather than a large number of very different systems. A number of systems exist in the literature, all using the term "clustering". We draw disctinctions in the way in which these systems work, and so we describe them now as differing entities.

Firstly, any process in which a group of objects is clustered into a single pile is either a *clustering process* or a *pseudoclustering process*. The distinction between the two processes lies in the way in which the clustering occurs. In a clustering process, there is a time-averaged net flow of pucks toward a single final destination, or at least away from a given initial point. This may include components of flow that move in either of the two directions, producing a net flow as indicated. However, a pseudoclustering process depends on the removal of clusters to produce its clustering effect. That is, it is necessary that two or more clusters exist, and there is an equal probability of removal from either cluster. In this case, the clusters that are smaller will tend to be removed more quickly, and once removed, the constituents of these clusters will be added to the remaining clusters. In this case, there is no specific tendency to place the constituents on larger clusters rather than smaller clusters. This means that the creation of a single cluster is not driven by the system, but rather a result of a large number of random events.

Though we do not show this here, there is a further consequence of this distinction. The dynamics of the two paradigms under increasing numbers of clusters vastly different. In clustering paradigms, the rate of convergence is expected to be polynomially dependent on the number of pucks in the system. This is certainly not the case for a pseudoclustering paradigm in which the rate is expected to be an exponential function of the number of pucks. The difference in scalability is quite compelling, as it indicates that only true clustering may be extended to large numbers of pucks of the order of $10^6 - 10^{10}$, which may be required for realistic construction processes. The distinction is a possible source of further research, but is not discussed in detail here.

We can mathematically define clustering and pseudoclustering in such a way as to create a distinction between the two. Clustering is any process in which the average rate of change of size of the smallest cluster in the system is a negative polynomial function of the size of the cluster. Pseudoclustering is any other process. Note that pseudoclustering does not preclude the formation of a single cluster. Rather it speaks only to the scalability of the system to larger sized systems.

CHAPTER 7

General Theoretical Arguments

In this section, a system of two clusters is considered. This is the simplest possible clustering task. In such a system, two clusters have been formed, and the desired goal is the eventual clustering of all pucks in one of the two clusters. General conditions under which we may expect these clusters to evolve under the action of the robots to create a single cluster are derived. We do not make any assumptions in this or any later section regarding the physical dimension of the system, and our results may be generalized to any applicable system in any number of dimensions.

In this section, it is assumed that the robot medium is *perfectly mixing.* This means that the probability that any given robot will interact with any given cluster depends *only on the size of the cluster*, and not the history of the robot. A generalization of this result under imperfect mixing conditions is discussed in Section 7.4.

7.1. General Considerations

7.1.1. Two Cluster Dynamics. Consider a system of two clusters and n_c robots. Let the clusters C_1 and C_2 contain n_1 and n_2 pucks, respectively. Assume that the robot media is perfectly mixing, and that the probability of any one cluster having an interaction with a robot is dependant only on some monotonically increasing function of the size of the cluster. At each interaction with a cluster, a robot will determine whether or not the cluster should have a puck removed or deposited. If a puck is to be removed and the robot is not carrying a puck already, then the robot will remove a single puck from the cluster. If, on the other hand, a puck is to be deposited and the robot is carrying a puck, the puck will be deposited in the cluster. We assume that these are the only two interactions possible, and that each robot can hold only one puck at a time. Such a system behaves as a simple diffusive system. In this system the two cluster system can be likend to two resevoirs of pucks mediated by some transport environment containing n_c robots.

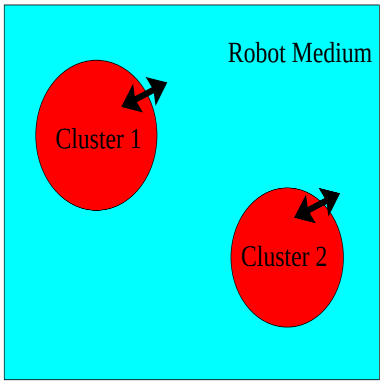


Figure 7.1.1: This is an abstract view of a puck-clustering system. In this view, the clusters are effectively resevoirs of pucks, and the robots are simply pathways for pucks to move between clusters.

If the number of robots holding a puck is n_m then, under the assumption that a robot may only carry a single puck, it follows that the number of robots available to remove a puck from the cluster is $n_c - n_m$. The rate of change of pucks in the two clusters C_1 and C_2 can be written as

$$\frac{dn_1}{dt} = -\left(n_c - n_m\right) f\left(n_1\right) + h\left(n_1\right) n_m \tag{7.1.1}$$

$$\frac{dn_2}{dt} = -\left(n_c - n_m\right) f\left(n_2\right) + h\left(n_2\right) n_m \tag{7.1.2}$$

where, f is the cross section for puck removal and h is the effective cross section for puck deposit.

If it is assumed that the number of pucks in the transport media is stationary (i.e., on average $\frac{dn_m}{dt} = 0$), it then follows that

$$\frac{dn_1}{dt} = -\frac{dn_2}{dt}, \tag{7.1.3}$$

From (7.1.1), (7.1.2), and (7.1.3) we obtain

$$\left(n_c - n_m\right) f\left(n_2\right) - h\left(n_2\right) n_m = -\left(n_c - n_m\right) f\left(n_1\right) + h\left(n_1\right) n_m \ . \tag{7.1.4}$$

Solving for n_m gives

$$n_m = \frac{n_c\left(f\left(n_1\right) + f\left(n_2\right)\right)}{f\left(n_1\right) + f\left(n_2\right) + h\left(n_1\right) + h\left(n_2\right)}. \tag{7.1.5}$$

This can then be substituted into (7.1.1) to produce the expression

$$\frac{dn_1}{dt} = n_c \frac{h(n_1) f(n_2) - h(n_2) f(n_1)}{f(n_1) + f(n_2) + h(n_1) + h(n_2)} \tag{7.1.6}$$

with an analogous expression for n_2.

The condition required for C_1 to grow given that $n_1 > n_2$ may be written as

$$\frac{dn_1}{dt} > 0;\ \forall\, n_1 > n_2. \tag{7.1.7}$$

If $h,\ f > 0$ then it follows from (7.1.6) that this condition is satisfied if

$$h(n_1) f(n_2) > h(n_2) f(n_1). \tag{7.1.8}$$

and so

$$\frac{f(n_2)}{h(n_2)} > \frac{f(n_1)}{h(n_1)}, \tag{7.1.9}$$

Thus for $C_1$to continue to grow

$$g(n_2) > g(n_1);\ \forall\, n_1 > n_2 \tag{7.1.10}$$

where $g(n) = \frac{f(n)}{h(n)}$. This condition holds if $g(n)$ is strongly monotonically decreasing and positive. Figure 7.1.2 shows an adequate form for g.

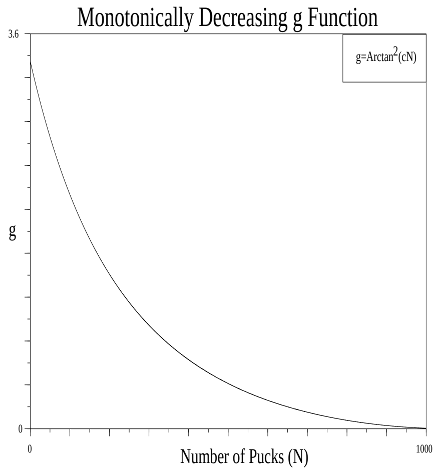


Figure 7.1.2: Any monotonically decreasing function will serve as a clustering condition.

Under the above assumptions, the condition for the growth of the cluster is that the ratio of the propensity to pick up pucks to that of the propensity to retrieve pucks be a strongly monotonically decreasing function.

7.1.2. Multicluster Dynamics. Again, the puck cluster system is as defined above, but with several clusters of pucks rather than only two. The question of whether or not a particular cluster will be reduced or increase in size depends on its interaction with the other clusters which may absorb evaporated pucks, as well as produce free pucks which may be clustered by the cluster in question. This problem may be approached by deriving conditions under which any given cluster will decrease monotonically in size.

The density of pucks in the transport media is again considered stationary. It is also again assumed that the robots are perfectly mixing. Then

$$\frac{dn_1}{dt} = -\left(n_c - n_m\right) f\left(n_1\right) + h\left(n_1\right) n_m \tag{7.1.11}$$

$$\frac{dn_2}{dt} = -\left(n_c - n_m\right) f\left(n_2\right) + h\left(n_2\right) n_m \tag{7.1.12}$$

$$\vdots$$

$$\frac{dn_q}{dt} = -\left(n_c - n_m\right) f\left(n_q\right) + h\left(n_q\right) n_m \tag{7.1.13}$$

and

$$\frac{dn_1}{dt} + \frac{dn_2}{dt} + \cdots + \frac{dn_q}{dt} = 0 \tag{7.1.14}$$

This gives

$$n_m = \frac{n_c \sum_i f\left(n_i\right)}{\sum_i f\left(n_i\right) + \sum_i h\left(n_i\right)} \tag{7.1.15}$$

which when substituted into (7.1.13) leads to the equation:

$$\frac{dn_q}{dt} = \frac{n_c \sum_{i \neq q} \left(h\left(n_q\right) f\left(n_i\right) - f\left(n_q\right) h\left(n_i\right)\right)}{\sum_i f\left(n_i\right) + \sum_i h\left(n_i\right)} \tag{7.1.16}$$

The rate of change in the size of the cluster C_q will be negative provided

$$\frac{dn_q}{dt} < 0$$

which leads to the condition that

$$g\left(n_q\right) = \frac{f\left(n_q\right)}{h\left(n_q\right)} > \frac{\sum_{i \neq q} f\left(n_i\right)}{\sum_{i \neq q} h\left(n_i\right)} \equiv g(n_{eff}) \tag{7.1.17}$$

where $g(n_{eff})$ is the effective cross section ratio of all clusters not belonging the C_q. In general, the determination of the effective rate must depend on the number of clusters and their relative sizes (as well as, eventually, their accessibility to the cluster in question).

Recall that if g is a monotonically decreasing function of cluster size then, for $n_1 > n_2$

$$f(n_1)\,h(n_2) > f(n_2)\,h(n_1) \quad . \tag{7.1.18}$$

If C_q is the smallest cluster then

$$f(n_q)\,h(n_i) > f(n_i)\,h(n_q) \tag{7.1.19}$$

$\forall\ i \neq q$. Summing over all $i \neq q$ it follows that

$$\sum_{i\neq q} f(n_q)\,h(n_i) > \sum_{i\neq q} f(n_i)\,h(n_q) \quad . \tag{7.1.20}$$

Rearranging, we find that

$$\frac{f(n_q)}{h(n_q)} > \frac{\sum_{i\neq q} f(n_i)}{\sum_{i\neq q} h(n_i)} \tag{7.1.21}$$

which is the condition for

$$\frac{dn_q}{dt} < 0 \tag{7.1.22}$$

Thus, irrespective of the forms of f and h, the time averaged behavior of the smallest cluster will be to decrease in size, as long as g is monotonically decreasing.

7.2. Clustering Systems

In this Section, we investigate several forms of g. We illustrate that these forms have the predicted behaviors in both the pair of cluster cases and the multiple cluster cases. We provide experimental evidence to support our approximation of the amount of variability in the number of pucks held by robots.

7.2.1. Simple Example.

A particularly simple example serves to give some idea of the requirements of the functional form of a puck system that converges.

Assume that the cross sections have the simple form

$$f(n) = cn^{\alpha} \tag{7.2.1}$$

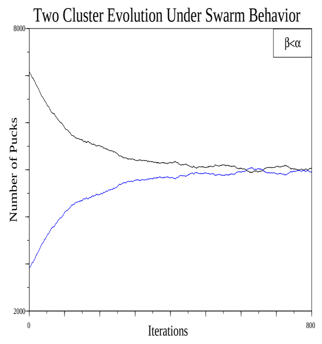
$$h(n) = cn^{\beta} \tag{7.2.2}$$

where c, α, and β are some constants. Then,

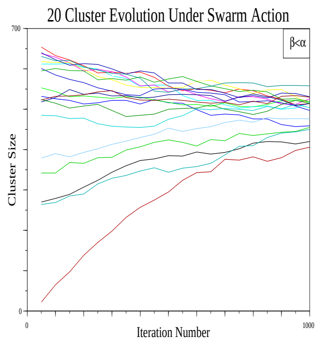
$$g(n) = n^{\alpha - \beta}. \quad (7.2.3)$$

Thus the cluster growth condition holds if $\beta > \alpha$. Where $\beta = \alpha$ then the system is static and the distribution of pucks in the clusters will remain the same on average.

Figures 7.2.1-7.2.3 present the results of simulations done in the non-embodied model (or instant transport) on a number of clusters initialized with nonzero sizes. Each of these simulations employs 50 robots, and the model is as given above, with various values for α and β. Each one runs to completion if clustering occurs or for a small number of steps, allowing the outcome to be clearly ascertained.

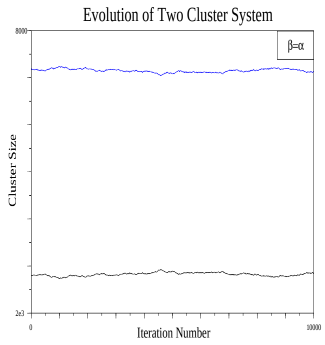


(a)

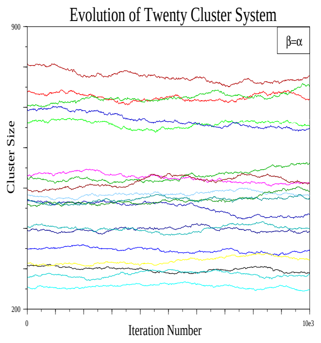


(b)

Figure 7.2.1: This illustrates the behavior of a system when $\alpha = 1.0$, and $\beta = 1.1$. $c = 0.00001$ for two-clusters, and $c = 0.0001$ for twenty clusters. As expected, neither system converges.

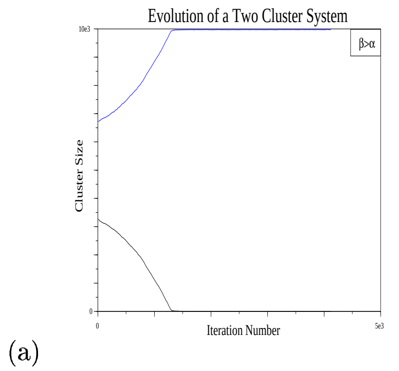


(a)



(b)

Figure 7.2.2: This illustrates the behavior of a system when $\alpha = 1.0$ and $\beta = 1.0$. $c = 0.00001$ for two clusters, and $c = 0.0001$ for twenty clusters. As expected, neither system converges.



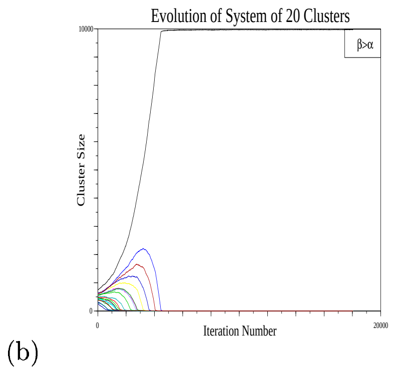


Figure 7.2.3: This illustrates the behavior of a system when $\alpha = 1.0$ and $\beta = 2.0$. $c = 0.00001$ for two clusters, and $c = 0.0001$ for twenty clusters. As expected, both systems converge to a system with one cluster containing all the pucks.

Figures 7.2.1 through 7.2.3 illustrate the convergence states of the system under the action of the robot swarm. The results are as expected. Only the case in which $\beta > \alpha$ produces a stable single cluster.

7.2.2. Physial Robot Model. Section 7.1 presents a general condition governing the condensation of two clusters. It is interesting to address the use of these conditions on real robots. Real robot systems have a limited sensory aperture. In the event that a given robot identifies the region of space based upon a line of sight, this aperture might be a active sensing device such as an IR transceiver or a passive sensing device such as a video camera. The physical design of the apparatus restricts the accurate identification of a region of space. This causes errors in the determination of the number of pucks in a particular region. It is these errors that allow the robot or robots to cluster the pucks into a single cluster.

In the given model the cross section for removal and deposit depend on two major factors. The first is the cross section σ of the cluster itself. The second is the angle θ at which the robot encounters the cluster. Based on this angle, the robot will classify the cluster as a high-density region or as a low-density region. In a high density region, this robot model will drop off pucks, and in low density regions, it will pick them up. Thus, the angle of approach has two mutually distinct classifications, each of which has a specific effect on the cross sections f and g. Thus, we may write $f = \sigma(1 - \rho)$ and $g = \sigma\rho$ and

$$g = \tfrac{1-\rho}{\rho} = \tfrac{1}{\rho} - 1 \ . \tag{14}$$

Recall that g must be a monotonically decreasing function of the number of pucks in order for the clusters to converge to a single cluster. This means that ρ must be a monotonically increasing function of the number of pucks in order to create a single cluster. That is, the probability of a cluster acquiring a puck must be a monotonically increasing function of the number of pucks. Conversely, the probability of losing a puck must be a monotonically decreasing function of the number of pucks.

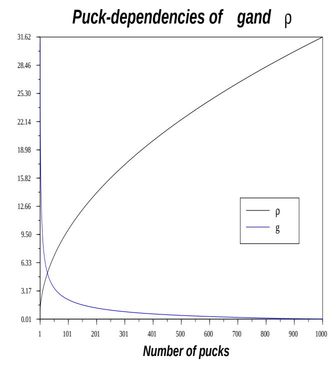


Figure 7.2.4: This figure gives the behavior of the function g with increasing ρ. The monotonic decrease is all that is required for g to lead to single clustering-behavior of the system.

It is instructive to derive the angular dependence, and hence dependence on the number of pucks, for a real robotic model.

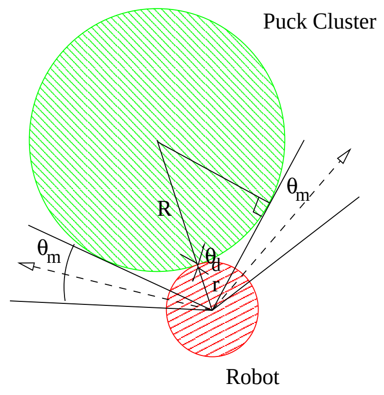


Figure 7.2.5: This figure illustrates the robot's interaction with the cluster. The robot will determine the density characterization for clusters whose pucks fall within the angle of interaction. All other interactions will be viewed as obstacle avoidance.

It is assumed that a robot has a cone of interaction which subtends an angle θ_m. In this model a robot will only deposit a puck if the direction of the robot is such that the cone falls completely within the cluster. If the cone only falls partly within the cluster, the robot, if it is able, will remove a puck. If follows, therefore, that the angular cross section for removal and deposit sum to the total angle of interaction of the robots cone $2\theta_m + 2\theta_d = \Theta$, where θ_m is the interaction angle and $\theta_d = \arcsin\left(\frac{R}{R+r}\right) - \theta_m$ is the deposit angle.

Using this construction gives

$$h = \sigma(R)\rho(R,r) \tag{7.2.4}$$

and

$$f = \sigma(R)(1 - \rho(R,r)) \tag{7.2.5}$$

where σ again is the cross section that the robot encounters a cluster and

$$\rho = \begin{cases} 0 \; if \; b < 0 \\ b \; if \; b \geq 0 \end{cases} \tag{7.2.6}$$

$$b = \frac{2\arcsin\left(\frac{R}{R+r}\right) - \theta_m}{2\arcsin\left(\frac{R}{R+r}\right) + \theta_m} = 1 - \frac{2\theta_m}{2\arcsin\left(\frac{R}{R+r}\right) + \theta_m}. \tag{7.2.7}$$

b and ρ both increase with R which, in turn, increases with the number of pucks n. Thus g for this model is monotonically decreasing and the system will form a single cluster.

Figure 7.2.5 gives a typical result of a simulation of this process. Initially, we have two clusters with nearly equal sizes. After many iterations, the migration of pucks from one cluster to the other causes the smaller of the two clusters to completely evaporate and the larger of the two clusters to absorb the liberated pucks.

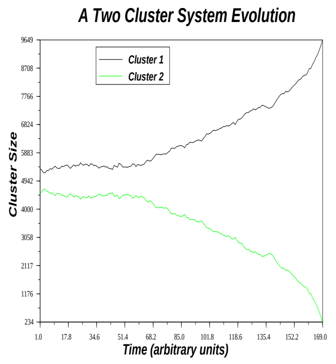
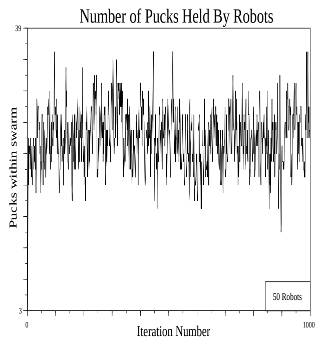


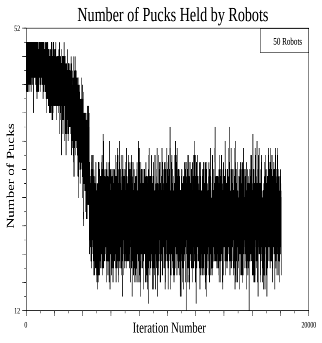
Figure 7.2.6: This figure gives the behavior of a two-cluster system under the influence of the robotic system described in this section.

7.2.3. Robot Interactions. In Section 7.1, we assumed that the robots would not absorb or evolve pucks on a large scale; the derivative of the number of pucks contained in the swarm could be ignored. We now turn to this assumption, offering empirical evidence that supports it.

We present data collected in the simulations given in Section 7.2.1. In these simulations, 50 robots make up the swarm.



(a)



(b)

Figure 7.2.7: These figures give the robot swarm puck occupancy. The swarm is at full capacity when the number of pucks in the swarm is 50. However, this never becomes the case in the (a) constant clusters size case, and once a single cluster is formed, this ceases to be the case (b) in the converging cluster case.

Figure 7.2.7 gives the evolution of systems of clusters in differing swarm conditions. In (a) the swarm does not change the cluster sizes appreciably, and so the occupancy of robots remains constant throughout the simulation. In (b), the swarm does converge, and the occupancy transitions from a full occupancy to a medium occupancy when the cluster is finally formed. This final occupancy is the equilibrium occupancy value which we return to in Section 6.

7.3. Cluster Evolution under the Influence of Differing Forms of g.

If we recall that g represents the effective behavior of robots over a large time, we can imagine that this is the effect of the swarm, and thus to step back from the system and ask questions about the design of the swarm around a specific goal. Let us examine this aspect of swarm engineering a bit more directly.

Swarm engineering, formally stated, requires that one creates a swarm condition and generates behaviors which satisfy this condition. Previously, we conditioned our swarm design on the desire to obtain a single cluster at the completion of the construction phase. This design requirement led us to the *local swarm condition* that

> *In a two cluster system, the largest cluster should increase in size, while the smaller decreases in size. In a many-cluster system, the smallest cluster should decrease in size.*

This condition led us to the general condition that g should be strongly monotonically decreasing. We now turn to the generation of further behaviors.

7.3.1. Increasing g. At the end of Section 7.1, we found that the condition required for creation of a single cluster of pucks independent of the initial conditions of the system was that g was a strongly monotonically decreasing function of N. In this Section, we investigate implications of the opposite condition on g. In the next Sections, we give the effect of these behaviors on various forms of g given various initial states.

First, we reverse the condition given in equation (7.1.9). Then

$$g\left(n_2\right)=\frac{f\left(n_2\right)}{h\left(n_2\right)}<\frac{f\left(n_1\right)}{h\left(n_1\right)}=g\left(n_1\right)\ . \tag{7.3.1}$$

This expresses the reverse condition. It is straightforward that this is true iff

$$f\left(n_1\right)h\left(n_2\right)-f\left(n_2\right)h\left(n_1\right)>0\ . \tag{7.3.2}$$

Since we have previously shown that

$$\frac{dn_1}{dt}=n_c\frac{h\left(n_1\right)f\left(n_2\right)-h\left(n_2\right)f\left(n_1\right)}{f\left(n_1\right)+f\left(n_2\right)+h\left(n_1\right)+h\left(n_2\right)}\ , \tag{7.3.3}$$

this together with (7.3.2) gives us that

$$\frac{dn_1}{dt}<0\ . \tag{7.3.4}$$

Moreover, since

$$\frac{dn_2}{dt}=n_c\frac{h\left(n_2\right)f\left(n_1\right)-h\left(n_1\right)f\left(n_2\right)}{f\left(n_1\right)+f\left(n_2\right)+h\left(n_1\right)+h\left(n_2\right)}, \tag{7.3.5}$$

we have

$$\frac{dn_2}{dt}>0. \tag{7.3.6}$$

This can be summed in the following Theorem.

> **Theorem 1 (Clustering Theorem)**: Suppose we have a two cluster system of pucks. If the ratio g as defined above is greater for cluster 1 than for cluster 2, then cluster 1 will lose pucks to cluster 2.

Note that this is also true for effective clusters in the same way as for real clusters, allowing this Theorem to be applicable to the multiple cluster system as well as the single cluster system. This means that any system in which one cluster will have a higher g value than the other will tend to lose pucks to the other, irrespective of the clusters' relative size. Now we turn to several rather specific peaked functions of g, in which the long time behavior is completely worked out.

7.3.2. Forms of g. We have established an interesting behavior of the swarm-cluster interaction under the action of differing g implementations. We now turn to the exhaustive investigation of some simple forms of g. Initially, we conditioned our investigation on the desire to have the system evolve to one containing a single large cluster independently of initial conditions at some time in the future. Moreover, we required that the system converge monotonically to this state. This produced g functions of the form given in Figure 7.3.1.

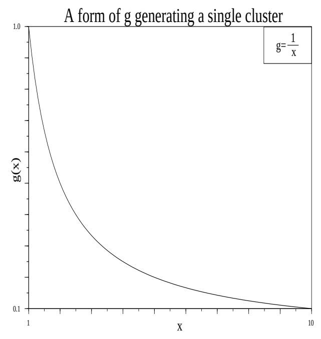


Figure 7.3.1: The function $g = \frac{1}{N}$ is a typical decreasing function of N which will yield a single cluster.

Suppose that g has a different form. For instance, suppose that g is a decreasing function of N for small N but an increasing function of N for large values of N (Figure 7.3.2).

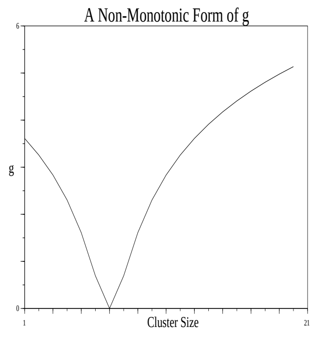


Figure 7.3.2: A form of g which has a minimum point, and an increasing behavior as time increases.

What then will be the long time convergence properties of the system? In general, this will be a function of the initial conditions and the form of g. We illustrate a complete set of possible outcomes in what follows.

We assume that g is made up of two linear regions, the relative slopes of which will be important to the eventual outcome of the system. We generally assume that the two linear regions meet at a point x_0 which is the minimum or maximum of these regions. We label the first linear region l_1 and the second l_2. Note that in order for these regions to produce a minimum, the signs of l_1 and l_2 are opposite. There are then three possible situations:

1. The slope of l_1 is greater than that of l_2.
2. The slope of l_2 is greater than that of l_1.
3. The slopes of $l_1$and l_2 are equal.

We shall see that these situations may lead to different end conditions given identical starting conditions. A general prediction of the outcome must include the relative slopes of the system, and the detailed initial cluster sizes.

7.3.3. Minimum with Equal Sloped Linear Regions. Let us suppose that we have a g made up of the juxtaposition of two linear regions which have the same magnitude of slope. Then we have three possible situations, with associated possible outcomes.

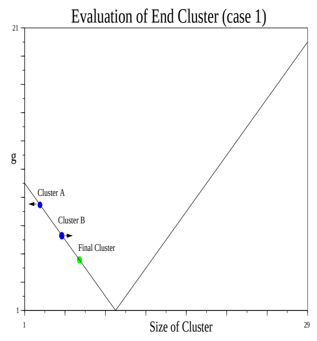


Figure 7.3.3: This gives the behavior of the pair of clusters when initiated on the decreasing region of g. The system evolves to one having a single cluster, as expected from above.

In Figure 7.3.3, we plot the evolution of a pair of clusters when initiated on the decreasing region of g. In this case, Cluster 1 will grow, absorbing Cluster 2, and eventually resulting in a single cluster of all the pucks. This is commensurate with the results of Section 7.1.

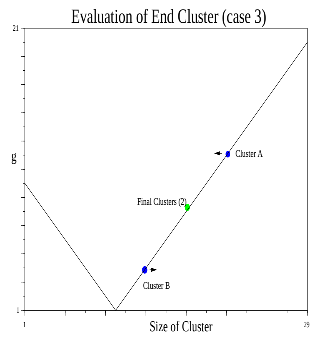


Figure 7.3.4: This gives the evolution of two clusters initiated on the increasing region of g. In this case, the clusters will evolve to the same size, equipartitioning the pucks.

If we instead put the clusters initially on the increasing region of g, the situation radically changes. Since the larger cluster now has the larger g value, it tends to lose pucks to the smaller cluster. Thus, the two clusters approach each other in size, and eventually become equal in size.

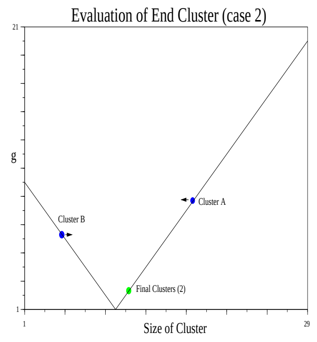


Figure 7.3.5: This figure illustrates the system evolution when the clusters are initialized on different sides of the minimum, with the larger cluster at a higher g value than the smaller.

Figure 7.3.5 illustrates the system evolution when the clusters are initialized on differing sides of the minimum with the larger cluster at a higher g value than the smaller cluster. In this case, the larger cluster will lose pucks to the smaller cluster. Since the slopes are equal, the loss of pucks is directly proportional to the relative loss of g. Thus, the larger cluster cannot "catch up" in g to the smaller. The result is that the process will continue until the initially smaller cluster passes the minimum of g and begins rising in g to meet the initially larger cluster. We end up with a system of two identically-sized clusters.

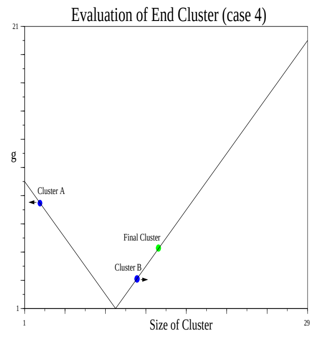


Figure 7.3.6: This Figure illustrates the evolution of a system of two clusters initially on differing sides of the minimum, with the smaller cluster at higher g.

In Figure 7.3.6, we illustrate the evolution of a system of two clusters initially on differing sides of the minimum. Initially, if the smaller cluster is at a higher g than the larger cluster, it will lose pucks to the larger cluster. This results in the gradual decrease in size of the smaller cluster until it is absorbed completely by the larger cluster.

Thus, if both regions have the same slope, there are two possible outcomes: the clusters merge to form one cluster, or the clusters equipartition the pucks.

7.3.4. Minimum with Large Region 1.

We now turn to one of two conditions in which the juxtaposition of linear regions forms a single minimum. In this SubSection, we assume that the magnitude of the slope of region 1 is greater than that of region 2.

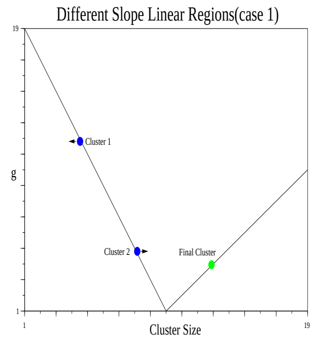


Figure 7.3.7: This gives the system evolution of two clusters initially on the decreasing region of g. As before, the system evolves to one containing a single cluster.

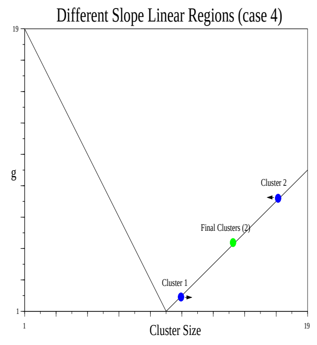


Figure 7.3.8: This Figure illustrates the system evolution of a system of two clusters initialized on the increasing region of g. The clusters evolve to two equal-sized clusters.

Figures 7.3.7 and 7.3.8 give the behavior of the system of two clusters initialized on the decreasing and increasing regions of g, respectively. In the first case, as in Section 7.1, the system evolves to one containing a single cluster of all the pucks.

As in Section 7.3.3, the second case evolves to one containing two clusters of equal size.

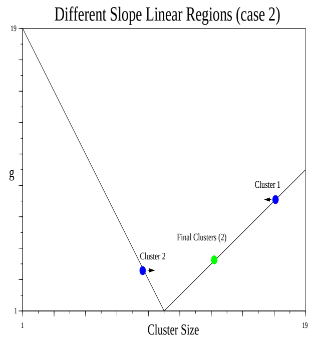


Figure 7.3.9: Evolution of a system of clusters in which two clusters are separated by the minimum, and the larger cluster is at higher initial g.

Figure 7.3.9 illustrates the evolution of a system of clusters in which two clusters are separated by the minimum, and the larger cluster is initially at higher g. The larger cluster will lose pucks to the smaller cluster, yielding two clusters of identical size.

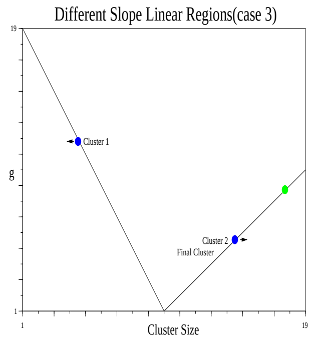


Figure 7.3.10: This Figure illustrates the evolution of a system of two clusters initially on opposite sides of the minimum in g with the smaller cluster at higher initial g than the larger cluster.

In Figure 7.3.10, the evolution of a two cluster system with two clusters separated by the minimum is depicted. In this case, the smaller cluster is at a higher initial g than the larger cluster. It therefore evolves pucks to the larger cluster, and the system eventually becomes one with a single cluster of all the pucks.

Thus, in this configuration of g the system has two possible final states. In one state, the system consists of a single cluster with all the pucks, while in the second, the system consists of two clusters of equal numbers of pucks.

7.3.5. Minimum with Large Region 2. In this SubSection, we assume that the magnitude of the slope of region 1 is smaller than that of region 2.

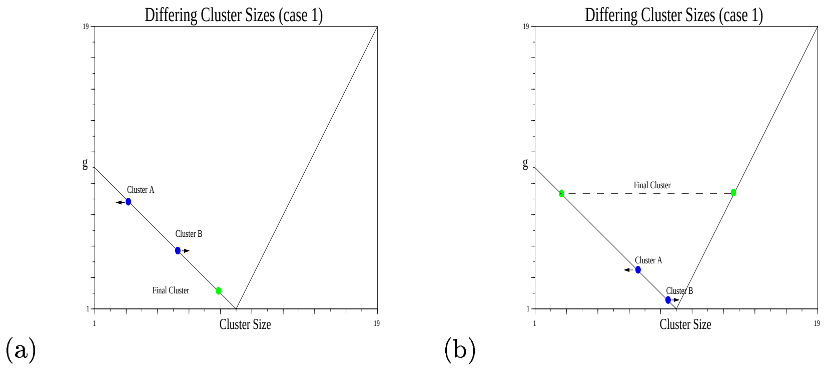


Figure 7.3.11: The evolution of a system of two clusters initialized on the increasing side of the g. This has the capability to evolve to a one cluster or two cluster system, with the two cluster system containing clusters of differing sizes.

Figure 7.3.11 illustrates the behavior of the system of two clusters both initialized on the decreasing side of g. In this case, the system has two possible stable outcomes. In the first, the pucks are completely absorbed by the larger cluster, yielding one final cluster (a). In the second, (b) the slope of the second region is sufficient for the g value of the larger cluster to "catch up" to that of the smaller cluster, yielding a stable configuration.

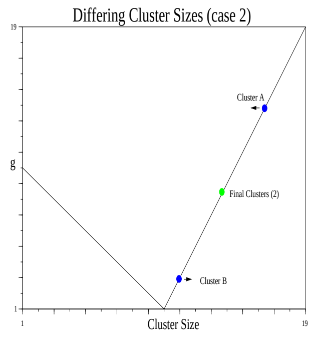
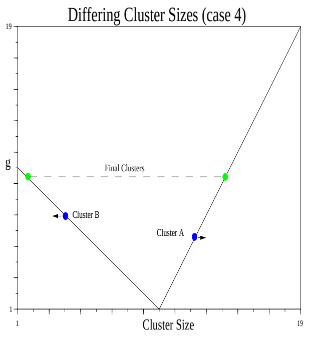
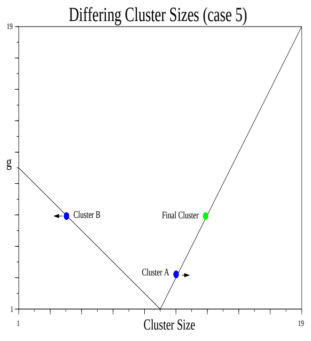


Figure 7.3.12: This gives the behavior of a two cluster system initialized on the increasing part of g. As before, the system forms a stable configuration of two clusters of the same size.

Figure 7.3.12 illustrates the behavior of a system of two clusters initially in the increasing region of g. As before, this system forms a stable configuration of two clusters with the same size.



(a)



(b)

Figure 7.3.13: This Figure illustrates the system evolution of a system of two clusters initialized in opposite sides of the minimum in g. Two outcomes are possible : a system of two *different sized* clusters, or one cluster.

In Figure 7.3.13, we give the evolution behavior of the system of two clusters initially on different sides of the minimum in g in which the larger cluster has a smaller g value than the smaller. This produces two different results depending on the relative slopes and positions of the clusters. In the first behavior, the slope on the increasing side is large enough that the g value of the larger cluster "catches up" to that of the smaller. At this point 7.3.13(a) the system forms a stable configuration of different sized clusters. The 7.3.13(b) second behavior is observed if the slope of the second region is insufficient for the g value of the larger cluster to "catch up" to that of the smaller cluster. In this case, a single cluster containing all the pucks is formed.

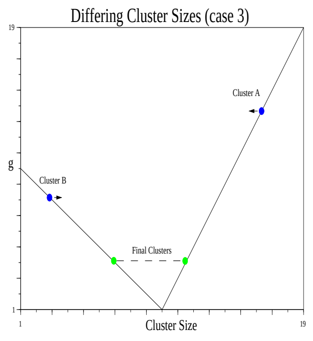


Figure 7.3.14: The evolution of a system of two clusters of differing sizes separated across the minimum in g in which the larger cluster has a larger initial g. The system evolves to a stable state containing two clusters of differing sizes.

In Figure 7.3.14, the system is initially comprised of two clusters of different sizes on opposite sides of the minimum in g. The larger cluster is at a higher g level than the smaller cluster, and evolves pucks which are absorbed by the smaller cluster. This process continues until the g levels become equal, which may happen before the clusters become the same size, as the slope of the second region is now greater than that of the first region. If the g levels do not become equal before the cluster sizes are equal, the system will produce two stable clusters of equal size.

Thus, there are now three possible outcomes if g is made up of two different regions in which the slope of the second has a larger magnitude than that of the first region. In some cases the system evolves to a stable state of one cluster, while in others the system evolves to a stable state containing two identical clusters. However, suprisingly, if the system is appropriately primed, it may evolve to a stable set of two differently sized clusters. This suprising result may be able to be exploited in order to create some interesting behavior.

7.3.6. Maximum with Equal Magnitude Slopes.

We have seen in the preceding three sections that the use of an increasing region of g tends to create two identically sized clusters in a stable configuration. This unexpected result is interesting to investigate further, and we do so by juxtaposing it with a second decreasing region of varying relative slopes. In this SubSection, we investigate the juxtaposition of the two regions of equal magnitude slopes.

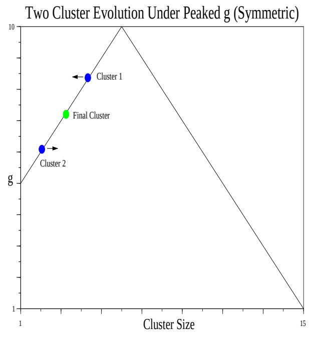


Figure 7.3.15: This figure illustrates the behavior of a system of two clusters with a peaked symmetric g. With the clusters initially on the increasing region of g the system evolves to one containing two identically sized clusters.

Figure 7.3.15 presents the behavior of a system of two clusters evolving under the action of a peaked symmetric g. In this case, the larger of the two clusters loses pucks to the smaller of the clusters, until the size of the clusters is the same.

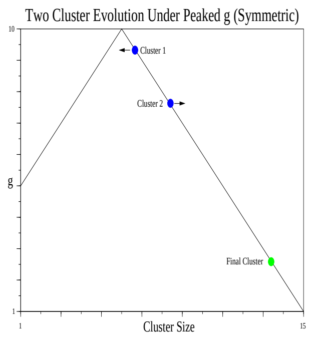


Figure 7.3.16: This Figure illustrates the behavior of a system of two clusters under the action of a symmetric g. A system with the clusters initially on the decreasing region of g evolves to one in which the smaller cluster loses pucks to the larger of the two until the larger absorbs the smaller.

Figure 7.3.16 depicts the evolution of a system of two clusters evolving under the action of a peaked symmetric g. In this case, the larger of the two clusters has the smaller g value, and so absorbs pucks from the smaller cluster. As the slopes are identical, the smaller cluster is not able to equal the larger cluster's g value even after it has passed the peak. Thus, the smaller cluster evaporates, leaving only the larger cluster.

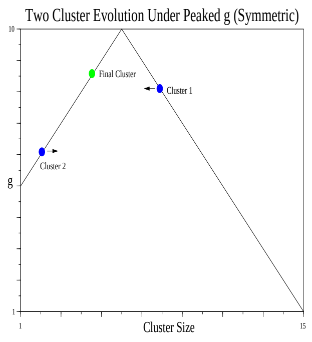


Figure 7.3.17: The evolution of a two cluster system under the influence of a symmetric peaked g. An initial configureation with each cluster on the opposite side of the peak, and with the larger of the two having the larger g value becomes a two cluster system with each cluster having identical size.

In Figure 7.3.17 is the evolution of a two-cluster system on a symmetric peaked g in which the larger of the two peaks is initially at a higher g value than the lower.

This will cause the larger cluster to evolve pucks, which are captured by the smaller cluster. The system then becomes a system with two clusters of the same size.

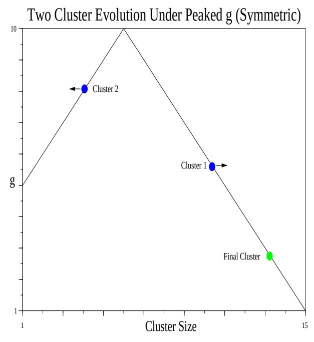


Figure 7.3.18: This is the evolution of a system of two clusters under the action of a single peaked g with the initial g of the smaller cluster larger than that of the smaller, and the two separated across the peak. The system becomes a single cluster system, with the larger of the clusters absorbing all the pucks.

In Figure 7.3.18 the evolution of a system of two small clusters under the action of a single-peaked g is plotted. Initially, the larger of the two clusters is located to the right of the peak, while the smaller is to the left of a peak. The smaller one, which has a higher g value, loses pucks, which are absorbed by the larger cluster. The system continues in this way until the larger cluster absorbs all the pucks from the smaller cluster.

Thus, the two behaviors of such a configuration are to form single clusters or two clusters of the same size.

7.3.7. Maximum with Left Asymmetric Slope Magnitudes. In the last SubSection, we investigated the symmetric peaked g case. In this SubSection, we investigate the asymmetric peaked g case with the larger slope magnitude in Region 1.

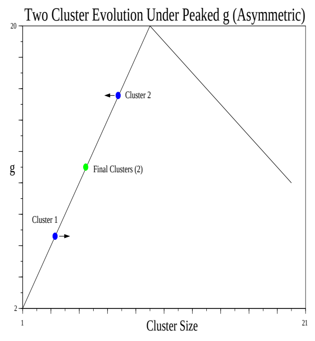


Figure 7.3.19: A system built on the increasing side of a peaked asymmetric g distribution evolves to a system of two clusters of equal size.

Figure 7.3.19 illustrates the behavior noted earlier in the increasing g region.

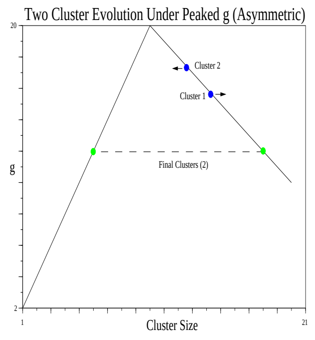


Figure 7.3.20: The behavior of the system when both clusters are on the decreasing region of g is different from that on the increasing regions of g, the system is able to form static systems of two clusters of differing sizes.

Figure 7.3.20 illustrates a rather unexpected behavior. If the system is initialized on the decreasing region of g. The smaller cluster will lose pucks to the larger cluster. This would seem to be similar to all cases of this kind discussed above. However, once the smaller cluster reaches the increasing region, as long as this region increases in g faster than the decreasing region decreases in g, the smaller cluster will catch the larger cluster in g and the system will fall into a state of dynamic equilibrium with two clusters of differing sizes. If $g(0) > 0$ then this may not form a static equilibrium state, but rather a single cluster of all the pucks.

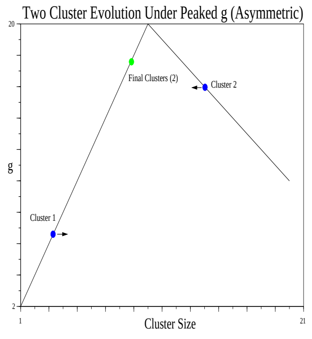


Figure 7.3.21: The behavior of the system when clusters are initialized on alternate sides of the maximum with the larger cluster at a higher g value than the smaller cluster.

Figure 7.3.21 demonstrates the behavior of a system of clusters initialized with the larger cluster in the decreasing region of g and the smaller cluster in the increasing region of g, with the value of g for the larger cluster higher than that for the lower cluster. In this case, the larger cluster will lose pucks to the smaller cluster, and the system will converge to a system of two clusters of equal size.

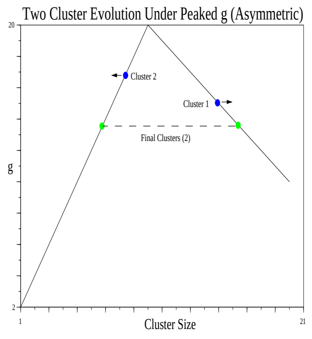


Figure 7.3.22: Evolution of a system of clusters initialized on opposite sides of the maximum, with g of the larger cluster lower than that of the smaller cluster.

Figure 7.3.22 shows the evolution of a system of two clusters initialized on opposite sides of the maximum, with g of the larger cluster lower than that of the smaller cluster. In this case, the smaller cluster will begin losing pucks to the larger cluster. However, this trend will continue only until the clusters' g values become equal. At that point, the system will form a second state of dynamic equilibrium in which the puck clusters maintain different sizes. This will happen, as in SubSection 6.4, only if the g value of the smaller cluster falls fast enough to avoid exhausting the cluster.

7.3.8. Maximum with Right Asymmetric Slope Magnitudes. The final case considered here is the asymmetric case in which the magnitude of the slope of g on the higher N side of the maximal point is higher than that on the lower N side of the maximal point. We begin again with the monotonic increasing and decreasing cases.

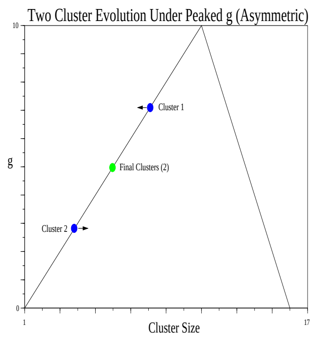


Figure 7.3.23: In this figure, we plot the evolution of a system of two clusters with g profile as indicated and initial sizes on the increasing region of g. As in earlier studies, we find that the system evolves to one containing two clusters of equal size.

Figure 7.3.23 gives the behavior of a pair of clusters evolving under the influence of a peaked asymmetric g with the magnitude of slope of the decreasing region larger than the magnitude of slope of the increasing region. In this case, as in previous cases, the system evolves to one containing two equal sized clusters with an intermediate size.

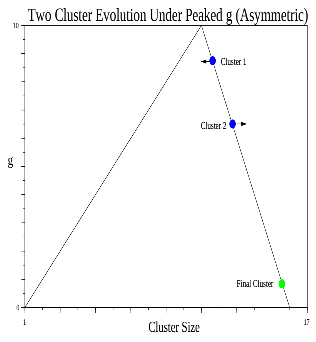


Figure 7.3.24: This figure gives the evolution of a system of two clusters under the influence of an asymmetric g with initial cluster sizes in the decreasing region of g. In this case, the smaller cluster is absorbed.

Figure 7.3.24 gives the behavior of a system of two clusters under the influence of an asymmetric g with initial cluster sizes in the decreasing region of g. In this case, the smaller cluster loses pucks to the larger cluster, as it's initial g is larger than that of the smaller. The process will continue until the smaller cluster disappears altogether.

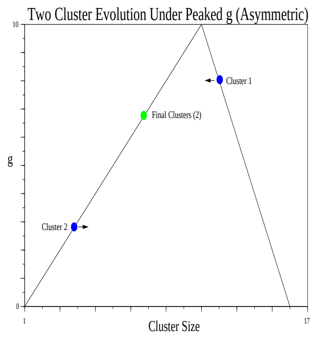


Figure 7.3.25: This figure gives the evolution of a system of two clusters under the influence of an asymmetric g with intial cluster sizes in different regions of g, and with a larger initial g for the larger cluster.

Figure 7.3.25 illustrates the behavior of a system of clusters under the influence of an asymmetric g with initial cluster sizes in differing regions of g, and with a larger initial large cluster g. The large cluster will lose pucks to the smaller cluster until it is brought into the increasing region, after which time the outcome becomes as detailed in Figure 5.8.1.

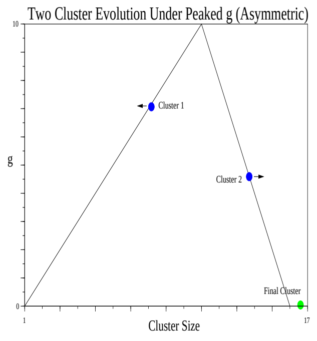


Figure 7.3.26: This figure gives the evolution of a system of two clusters under the influence of an asymmetric g with initial cluster sizes in different regions of g, and with a larger initial g for the smaller cluster.

Figure 7.3.26 presents the final behavior of our study – that of two clusters which begin on opposite sides of the maximum of g, with the smaller cluster having an initial g larger than the larger. In this case, as the magnitude of the slope of the descending region is larger than that of the increasing region, the smaller cluster cannot ever catch the larger one in g value, and it will eventually be absorbed.

Thus, the two behaviors of this type of asymmetric system are to form a single cluster and to form a set of two clusters of equal size.

7.3.9. Stability of Multi Points. We define *multi points* as those points at which the system settles in which two or more stable clusters exist in dynamic equilibrium with another. A multi point refers to the size of the cluster, rather than the value of g.

We wish to understand under what conditions we can expect these multi points to be stable. In order for them to be stable, there must be a restoring force when the system moves out of equilibrium. Thus, let us start with the condition for the existence of a multi point. We must have two clusters of sizes N_1 and N_2, which are not necessarily equal. We must have that

$$g(N_1) = g(N_2) \tag{7.3.7}$$

Then if we change the size of cluster 1 by some amount Δ, by our conservation of pucks condition, we must change the size of cluster 2 by the amount $-\Delta$. Thus we have that $g(N_1)$ becomes $g(N_1+\Delta)$ and $g(N_2)$ becomes $g(N_2-\Delta)$. Now, in order for a restoring force to exist, cluster 1 must start losing pucks. This can generally occur (Theorem 1) if

$$g(N_1+\Delta) > g(N_2+\Delta)\ . \tag{7.3.8}$$

If Δ is small, or these are linear regions, then we may approximate $g(N_1+\Delta)$ as

$$g(N_1+\Delta) \approx g(N_1) + \Delta\,\left.\frac{\partial g}{\partial N}\right|_{N_1} \tag{7.3.9}$$

and

$$g(N_2-\Delta) \approx g(N_2) - \Delta\,\left.\frac{\partial g}{\partial N}\right|_{N_2}\ . \tag{7.3.10}$$

Thus, noting that $g(N_1) = g(N_2)$, the condition becomes

$$\left.\frac{\partial g}{\partial N}\right|_{N_1} > -\left.\frac{\partial g}{\partial N}\right|_{N_2} \tag{7.3.11}$$

This means that the two regions must have two conditions met. First, the region in which cluster 1 is located must be increasing. Second, the region which cluster 2 is located must not be decreasing faster than the region around cluster 1 is located.

This gives the explanation for why we found multi points only in the situations outlined in Sections 7.3.5 and 7.3.7, and puts a strong bound on the appearance of multi points.

7.4. Imperfectly mixing media

Under the conditions that the media is rapidly mixing, it is possible to investigate the system using the number of pucks as the basis of investigation. This requires the assumption that the probability of encounter was either unrelated to or is some increasing function of the number of pucks. However, this assumption breaks down in imperfectly mixing media. In this Section, we consider densities of pucks rather than their numbers. This is motivated by a similarity between systems of moving robots and pucks and systems of gas molecules. Using this as a basis for our study, we develop an analagous condition to the previously derived condition for perfectly mixing systems. This is important because, while the use of only numbers requires perfect mixing, local densities do not, and so our analysis applies to real systems.

7.4.1. General Considerations. Previously the rate of evolution of a cluster was derived in terms of the number of robots in the system. The assumption that the robots are perfectly mixing allowed the formulation of equations solely in terms of the number of pucks in each cluster. However, in a real system, a cluster's evolution is determined in part by the local density of carriers rather than by the number of pucks in the cluster. It is necessary, then, in order to include real systems in our formalism, to reformulate the previous equations in terms of the local density of carriers and the number of pucks rather than the sheer number of pucks alone.

Two assumptions are made.

1. The average density of robots in the arena is constant.
2. Aside from the evaporation (and condensation) of pucks from (onto) a cluster, the behavior of pucks under the action of the robots will be to become on average equally dispersed throughout the space.

We consider the ability of the system to cluster under these assumptions.

Suppose that we have a space in which there exists a sustained gradient of puck density. In the simplest of cases, this gradient is linear in some direction $\widehat{\mathbf{x}}$. If we denote the distance along this axis by x, then we would assume that the density δ is a function of x, $\delta(x)$. Assume that δ is monotonically increasing from one end of the arena to the other. It is interesting to understand what might be happening here.

Recall that, all other things remaining equal, it may be assumed that the natural tendency is that the average density of moving pucks will tend to become constant. What this means is that there will be a net flow of pucks from high density areas to low density areas. This will tend to happen irrespective of any other considerations. However, this means that in order to sustain the density gradient, there must be a puck sink at the low density area of the arena. This means that there may only be a gradient if there is a net flow of pucks from the high density area to the low density area.

The opposite question is also important: what is the requirement for a net flow of pucks from one area of the arena to the other? If we do not let the robots exchange pucks, this means that a single robot carrying a puck must move from one region and drop it off in the other region. If the total density of robots is constant, this means that the number of empty robots on the low density side will be larger than that on the high density side. The end result is that a density gradient will occur and be maintained. This is an important consequence.

A net flow of pucks from one area of an arena to another may occur if and only if there exists a monotonic gradient in the density of robots carrying pucks between the areas in question, in which the density of robots carrying pucks is smaller at the end receiving the pucks.

In general, the rate of change of the size of a particular cluster, then, depends on two things. The first is the number of carriers interacting with the cluster. The second is the nature of the interaction of a carrier with the cluster in question. The rate of change of the size of a single cluster may be written as

$$\frac{dn_1}{dt} = -\delta_e F(n_1) + \delta_f H(n_1) \tag{7.4.1}$$

where the local density of empty robots δ_e and of robots carrying pucks δ_f are functions of position and time. In this equation, the functions F and H are analagous to previous functions, but now include the geometric properties of the arena and cluster position(s).

If we have two clusters, then the system is described by

$$\frac{dn_1}{dt} = -\delta_e(x_1) F(n_1) + \delta_f(x_1) H(n_1) \tag{7.4.2}$$

$$\frac{dn_2}{dt} = -\delta_e(x_2) F(n_2) + \delta_f(x_2) H(n_2) \tag{7.4.3}$$

If again, we assume that the number of pucks in the medium is stationary, then we have that

$$\frac{dn_1}{dt} = -\frac{dn_2}{dt} \tag{7.4.4}$$

or

$$-\delta_e(x_1)\,F(n_1) + \delta_f(x_1)\,H(n_1) = \delta_e(x_2)\,F(n_2) - \delta_f(x_2)\,H(n_2) \tag{7.4.5}$$

In general, the density of carriers is not constant throughout the region. However, if we consider only the density of robots in the arena, either empty or filled, if follows that this density is, on average, constant. If we let D denote the time-averaged density of robots in the arena, and δ_e and δ_f denote the time averaged local densities of empty and full robots, respectively, then

$$D = \delta_e + \delta_f \tag{7.4.6}$$

We assume that we are working in a state of dynamic equilibrium, in which the local density of empty carriers and of full carriers is approximately constant. This means that even though there may be a net flow of pucks between clusters in the system we now consider, we assume that the local densities are remaining constant.

The way that the clusters evolve depends on the equilibrium states. In general, the system will behave in such a way as to move toward the equilibrium states of the individual clusters. The interaction between clusters will occur through the robot media, and there will be two competing tendencies. The first tendency will be to have time averaged densities of robots become equal throughout the space. The second will be the tendency of robots nearby clusters to remove or put down pucks, moving the local densities toward the equilibrium state of the cluster. We will derive the following equilibrium densities in Section 7.4.2, as it is a significant aside from our current discussion. For now, we state without proof that the equilibrium relations for a single cluster are,

$$\delta_f = D\frac{1}{1+\frac{H}{F}}, \text{ and } \delta_e = D\frac{1}{1+\frac{F}{H}}. \tag{7.4.7}$$

Then, if we denote the local densities around cluster one by δ_{e_1} and δ_{f_1}, with analogous expressions for cluster two, substituting equation (7.4.6) into equation (7.4.5),

$$-\left(D-\delta_{f_1}\right)F(n_1) + \delta_{f_1}H(n_1) = \left(D-\delta_{f_2}\right)F(n_2) - \delta_{f_2}H(n_2) \tag{7.4.8}$$

This can be rearranged to read

$$\delta_{f_1} = \frac{D\left(F\left(n_2\right) + F\left(n_1\right)\right) - \delta_{f_2}\left(F\left(n_2\right) + H\left(n_2\right)\right)}{\left(F\left(n_1\right) + H\left(n_1\right)\right)} \ . \tag{7.4.9}$$

This will be smaller than δ_{f_2} iff

$$D\left(F\left(n_2\right) + F\left(n_1\right)\right) < \delta_{f_2}\left(F\left(n_2\right) + H\left(n_2\right)\right) + \delta_{f_2}\left(F\left(n_1\right) + H\left(n_1\right)\right) \ . \tag{7.4.10}$$

$$\frac{\left(F\left(n_1\right) + H\left(n_1\right)\right) + \left(F\left(n_2\right) + H\left(n_2\right)\right)}{\left(F\left(n_2\right) + F\left(n_1\right)\right)} > \frac{D}{\delta_{f_2}} = \frac{F\left(n_2\right) + H\left(n_2\right)}{F\left(n_2\right)} \tag{7.4.11}$$

giving

$$F\left(n_2\right) H\left(n_1\right) > F\left(n_1\right) H\left(n_2\right) \tag{7.4.12}$$

or

$$\frac{F\left(n_2\right)}{H\left(n_2\right)} > \frac{F\left(n_1\right)}{H\left(n_1\right)} \tag{7.4.13}$$

This is the same condition found above for the case of the perfectly mixing robots. That is, if the ratio $G = \frac{F}{H}$ is monotonically decreasing, then the equilibrium density around larger clusters will be smaller than that around smaller clusters. δ_{f_1} being smaller than δ_{f_2} would seem to indicate that there are fewer robots carrying pucks nearby cluster one than nearby cluster two. This indicates that the robots are more successful in removing pucks from cluster two than from cluster one, and more successful in depositing pucks on cluster one than on cluster two. In this case, cluster one will tend to grow while cluster two tends to decrease. Moreover, there will be a net flow of pucks from cluster two to one. Put another way, this indicates that under the action of the robots, the smaller clusters will pump pucks into the media trying to reach their equilibrium, while the larger clusters will absorb these pucks, trying to move to their smaller equilibrium.

Note that the relative density of empty or full carriers is a function of the design of the robot and the number of pucks in the cluster, as well as the cluster's detailed characteristics (eg. its geometry). Thus, as long as the design of a robotic system tends to create lower densities of pucks around larger clusters, the larger of two clusters will tend to monotonically increase in size on average.

Perhaps most importantly, this result is easily generalizable to the multiple cluster regime. The minimum condition required is that the local density of robots carrying pucks induced by the cluster is a monotonically decreasing function of cluster size. Thus, nearby any given small cluster, the density will increase over that closer to larger clusters. Moreover, the smallest cluster will have the highest equilibrium density, and therefore can be seen to monotonically decrease in size, as in the previous section.

Mathematically, this can be shown as follows. Suppose that cluster q is the smallest cluster in the system, and that there are n clusters. Then we have

$$F\left(n_q\right) H\left(n_i\right) > F\left(n_i\right) H\left(n_q\right) \tag{7.4.14}$$

for all $i \neq q$. Then, we have

$$\sum_{i \neq q} F\left(n_q\right) H\left(n_i\right) > \sum_{i \neq q} F\left(n_i\right) H\left(n_q\right) \tag{7.4.15}$$

or that

$$\frac{F\left(n_q\right)}{H\left(n_q\right)} > \frac{\sum_{i \neq q} F\left(n_i\right)}{\sum_{i \neq q} H\left(n_i\right)} \tag{7.4.16}$$

which indicates that the effective cluster built from the detailed design of the other clusters tends to absorb pucks from the smallest cluster, mirroring our previous investigation of this topic.

Of course, the *efficiency* is another matter. This result does not help us to understand whether or not the system will quickly approach the desired equilibrium state. We return to this point in Section 7.5.

7.4.2. Induced Densities.

In (7.4.7), the equilibrium densities of empty and full robots around a cluster of a given size and interaction dynamic were stated without proof. This subsection is devoted to deriving these relations.

We assume that we have a single cluster located in a container of some kind. The robots interact with the cluster according to the previously-defined functions F and H, and may carry at most one puck at a time. We assume that there is no sink or source either of robots or of pucks.

In order for a cluster to be in equilibrium with a robotic (or other) medium, it must be that

$$\delta_e F = \delta_f H \tag{7.4.17}$$

where H and F represent the cross sections for deposit and for removal for a cluster of size n. If $D = \delta_e + \delta_f$[1] then

$$\delta_e F = \left(T - \delta_e\right) H \ . \tag{7.4.18}$$

[1]This condition implicitly requires the arena in which the robots are behaving to be closed or otherwise bounded.

Thus

$$\delta_e = D\frac{H}{F+H}, \; \delta_f = D\frac{F}{F+H} \tag{7.4.19}$$

These results may be rewritten as

$$\delta_f = D\frac{1}{1+\frac{H}{F}}, \text{ and } \delta_e = D\frac{1}{1+\frac{F}{H}}. \tag{7.4.20}$$

In the event that $\frac{F}{H}$ is a monotonically decreasing function of n, the evolution of pucks from a given cluster will decrease as its size increases. The equilibrium density surrounding a large cluster will be smaller than that around a small cluster. Thus, the pucks will cluster into a single cluster as long as this condition is met. Happily, this is identical to our previous result, though it now contains no restrictions.

Suppose that we have two cross sections F and H given by

$$F = \sigma n^{\alpha} \tag{7.4.21}$$

and

$$H = 1 \; . \tag{7.4.22}$$

These together give us that

$$G = \sigma n^{\alpha} \; . \tag{7.4.23}$$

We may use this in our expression of δ_f . We have in a discrete model

$$\delta_f = T\frac{1}{1+\frac{H}{F}} = T\frac{\sigma n^{\alpha}}{\sigma n^{\alpha}+1} \; . \tag{7.4.24}$$

In Figure 7.4.1, we plot the prediction of this model and the measured occupancy of the robots for a nonembodied single cluster system with a varying number of pucks and 150 robots.

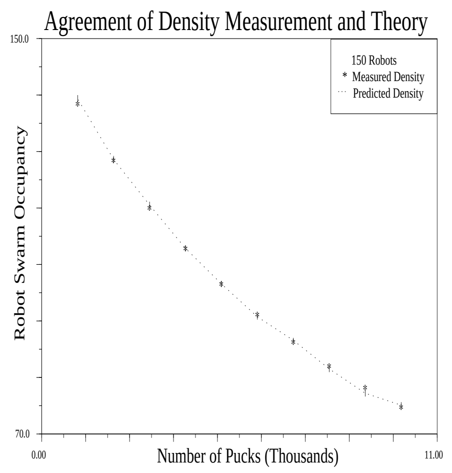


Figure 7.4.1: This figure gives the agreement of the predicted and measured values of the occupancy of robots in a swarm model. In this case, there is one cluster of varying size, the robots are nonembodied and quickly circulating, there are 150 robots, and the propensities are given by $F = \sigma n^{\alpha}$ and $H = 1$.

The agreement here is remarkable, and indicates that the equilibrium density of the robots is well modeled above.

7.5. Efficiency

So far, we have investigated the question of whether or not a single cluster will form at all. This is compelling work, to be sure, but not completely satisfying to the engineer. The main question to an engineer, after verification that the algorithm will work eventually, is how to increase the speed of an algorithm. In this section, we discuss methods of providing algorithms that are at once capable of carrying out the task and of carrying it out efficiently.

7.5.1. Positive Induced Transport. In general the net transport between two clusters of differing sizes is an increasing function of the difference between their sizes. That is, if the difference between the sizes of two clusters is z then we expect the rate of transfer of pucks from one cluster to the other will be an increasing function of z. Mathematically, this may be expressed as

$$\frac{\partial F}{\partial z} \geq 0 \tag{7.5.1}$$

where F represents the puck flux around the smaller cluster.

Suppose that we have two G functions, say G_A and G_B corresponding to two different interaction schemes of simple puck clustering robots. Then around cluster

1, the scheme A will have an equilibrium density of empty carriers given by

$$\delta_e^A(N_1) = T\frac{H^A(N_1)}{F^A(N_1)+H^A(N_1)} = \frac{T}{G^A(N_1)+1} \tag{7.5.2}$$

$$\delta_f(N_1) = T\frac{F^A(N_1)}{F^A(N_1)+H^A(N_1)} = T\frac{G^A(N_1)}{G^A(N_1)+1} \tag{7.5.3}$$

while near cluster two, the equilibrium density will be given by

$$\delta_e^A(N_2) = T\frac{H^A(N_2)}{F^A(N_2)+H^A(N_2)} = \frac{T}{G^A(N_2)+1} \tag{7.5.4}$$

$$\delta_f(N_2) = T\frac{F^A(N_2)}{F^A(N_2)+H^A(N_2)} = T\frac{G^A(N_2)}{G^A(N_2)+1} \tag{7.5.5}$$

Thus, we find that

$$\Delta\delta_e^A = \frac{T\Delta G^A}{(G^A(N_1)+1)(G^A(N_2)+1)} \tag{7.5.6}$$

and

$$\Delta\delta_f^A = -\frac{T\Delta G^A}{(G^A(N_1)+1)(G^A(N_2)+1)} \tag{7.5.7}$$

where $\Delta G = G(N_2) - G(N_1)$. Thus, if the transport of one algorithm characterized by G_A is greater than that of another characterized by G_B, we must have that

$$\frac{T\Delta G_A}{(G_A(N_1)+1)(G_A(N_2)+1)} > \frac{T\Delta G_B}{(G_B(N_1)+1)(G_B(N_2)+1)} \tag{7.5.8}$$

or that

$$\Delta G_A > \Delta G_B\frac{(G_A(N_1)+1)(G_A(N_2)+1)}{(G_B(N_1)+1)(G_B(N_2)+1)} \tag{7.5.9}$$

Of course, this is dependant on both strategies using the same techniqe for moving pucks around the arena. Any change in basic strategy would change the formalism above.

Suppose first that

$$G_B = G_A + r(N) \tag{7.5.10}$$

We wish to find the minimal condition on the function r which creates a second function G_B that is less efficient than G_A. Now, we may assume without a loss of generality that $r(N_1) = 0$. This gives us

$$\frac{\Delta G_A}{(\Delta G_A - r(N_2))} > \frac{(G_A(N_2)+1)}{(G_A(N_2)+r(N_2)+1)} \tag{7.5.11}$$

Since $\Delta G_A > r(N_2)$, we have that

$$\frac{(\Delta G_A - r(N_2))}{\Delta G_A} < \frac{(G_A(N_2) + r(N_2) + 1)}{(G_A(N_2) + 1)} \tag{7.5.12}$$

$$1 - \frac{r(N_2)}{\Delta G_A} < 1 + \frac{r(N_2)}{(G_A(N_2) + 1)} \tag{7.5.13}$$

$$-\frac{r(N_2)}{\Delta G_A} < \frac{r(N_2)}{(G_A(N_2) + 1)} \tag{7.5.14}$$

which leads us to

$$r(N_2)\left(\frac{1}{\Delta G_A} + \frac{1}{(G_A(N_2) + 1)}\right) > 0 \tag{7.5.15}$$

or more simply that

$$r(N_2) > 0 \tag{7.5.16}$$

Thus, if we have two functions G_A and G_B, the latter will be less efficient if we simply have

$$G_B > G_A \tag{7.5.17}$$

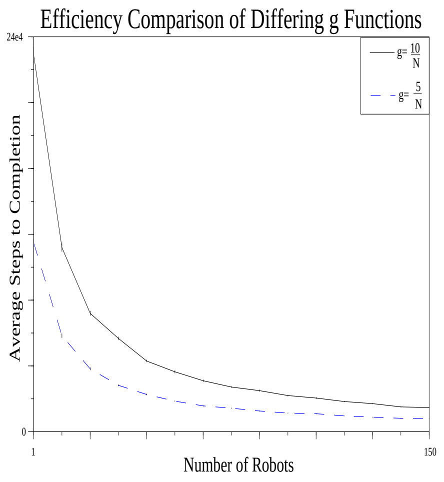


Figure 7.5.1: This figure illustrates the performance of the puck aggregation for a two cluster system with a number of different robot swarm sizes.

Figure 7.5.1 illustrates the performance of a several swarms of robots on a swarm aggregation task in which the g functions were given by

$$g_1 = \frac{10}{N} \tag{7.5.18}$$

and

$$g_2 = \frac{5}{N} \, . \tag{7.5.19}$$

Each performance measure for each swarm size was calculated using a perfectly mixing model of clustering in as described in Section 7.2. 10000 pucks were initialized in equal proportions in two clusters, requiring the system to randomly fluctuate out of equilibrium and into clustering behavior. As predicted above, the performance of the swarm with a smaller g is better than that of the swarm with the larger g in that the clustering time was significantly smaller, on average.

7.5.2. Induced Transport. Previously, we assumed that the induced transport was a positive increasing function of the difference in densities between two clusters. The analog for perfect mixing is that the induced transport is a positive increasing function of the difference in the number of pucks in the cluster. In this subsection, we verify this assumption in the imperfect mixing case. We show this in this section, completing our analysis.

Previously, we found that for a given cluster of n_1 pucks the rate of growth of the cluster is

$$\frac{dn_1}{dt} = -\delta_e(x_1)\, F(n_1) + \delta_f(x_1)\, H(n_1) \tag{7.5.20}$$

Recalling that in equilibrium

$$\delta_e = T\frac{G}{G+1}, \ \delta_f = T\frac{1}{G+1} \tag{7.5.21}$$

We wish to verify that the rate of transport of pucks is an increasing function of the difference of induced densities between clusters. First, we assume that we have two clusters of differing size, say of sizes n_1 and of size n_2, respectively. These induce densities (in equilibrium) of

$$\delta_{1,f} = \frac{T}{G_1+1} \text{ and } \delta_{2,f} = \frac{T}{G_2+1} \tag{7.5.22}$$

Now, we assume that the flow of pucks is linearly dependent on the rate of change of the density of pucks. That is,

$$\frac{1}{\eta}F = \frac{\partial \delta_f}{\partial x} + \frac{\partial \delta_f}{\partial y} = -\frac{1}{\eta}\frac{dn_1}{dt} \tag{7.5.23}$$

If the cluster is much larger than the number of robots, this is approximately constant flow. This gives

$$F = \eta \left(\frac{\partial \delta_f}{\partial x} + \frac{\partial \delta_f}{\partial y} \right) = C \tag{7.5.24}$$

The net flow from any area is a constant C, assuming that no pucks are allowed to cluster in any area of the arena. Solutions to this differential equation are

$$\delta_f = kx + ly + C_1 \tag{7.5.25}$$

Assuming that both clusters are on the $x - axis$, we may deduce that

$$k = \frac{\delta_{2,f} - \delta_{1,f}}{\Delta x} \tag{7.5.26}$$

Moreover, if the clusters are located within a bounded arena, we may conclude that $l = 0$, or that there is no gradient along the direction perpendicular to the radii connecting the clusters. Thus, we have

$$\delta_f\left(x\right) = \frac{\delta_{2,f} - \delta_{1,f}}{\Delta x} x + \delta_{2,f} \tag{7.5.27}$$

giving

$$F = \eta \left(\frac{\delta_{2,f} - \delta_{1,f}}{\Delta x} \right) = C \tag{7.5.28}$$

Clearly, if

$$\frac{\delta^1_{2,f} - \delta^1_{1,f}}{\Delta x} > \frac{\delta^2_{2,f} - \delta^2_{1,f}}{\Delta x} \tag{7.5.29}$$

where $\delta^1_{1,f}$ represents the induced density at cluster one for scheme one, $\delta^2_{1,f}$ represents the same for control scheme 2, and so on. Thus, as long as this difference is larger, for scheme two, then the induced flow is larger. The rest of our conclusions from above follow quite nicely.

7.6. Predictions of the Model

In this section, we present calculations based on the puck clustering models of Beckers et al. (Beckers et al., 1996) and Maris and Boekhorst (Maris and Boekhorst 1996). We show that, as observed, the first model is predicted to cluster, while the second is not.

Let us first consider the model of Beckers et al. Beckers and colleagues report that the robot has four possible interactions with the robot.

1. Pick up a single puck
2. Pick up two pucks
3. Drop a single puck

4. Drop two pucks.

A robot travelling in straight lines will experience cluster as in Figure 7.6.1.

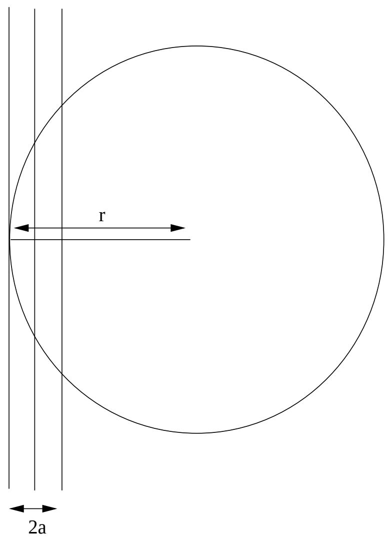


Figure 7.6.1: The modes of interaction of the robots and clusters in Beckers et al. clustering paper.

The robot has a probability of $\frac{a}{r}$ of pulling off a single puck, $\frac{2a}{r}$ of pulling off two pucks (if empty), and a probability of dropping of $\frac{r-a}{r}$ if full of one, and of 1 if it has two pucks. Thus, we may write an expression for h and f as

(7.6.1) $$h = \lambda\left(r-a\right)n_1 + 2\lambda r n_2$$

and

(7.6.2) $$f = \lambda a n_1 + \lambda a n_0 + 2\lambda a n_0$$

where n_0, n_1, and n_2 represent the number of robots carrying 0, 1, and 2 pucks, respectively, and λ is a proportionality constant. Then

(7.6.3) $$g = \frac{a}{r}\frac{(n_1+3n_0)}{\left(1-\frac{a}{r}\right)n_1 + 2n_2}$$

Now, if $r \gg a$ then

(7.6.4) $$g \simeq \frac{a}{r}\frac{(n_1+3n_0)}{n_1+2n_2}$$

This will generally be decreasing if n_0 is nearly constant (as in when robots are running in a large arena or with a small interaction probability). If this is the case, then $n_0 \gg n_1 \gg n_2$ and

$$g \simeq \frac{a}{r}\frac{3n_0}{n_1} \simeq \frac{a}{r}\frac{3}{\alpha} \tag{7.6.5}$$

where α is some linear proportionality constant, generally a function of a and n_0. This also explains why many clustering experiments failed when carried out in a very small arena, where the n_1 and n_2 are large, but not in larger arenas.

Martinoli, Ijspeert, and Mondada (Martinoli et al. 1999) report the use of a clustering experiment in which a khepera robot manipulates seeds and attempts to place them in long semi-linear "clusters".

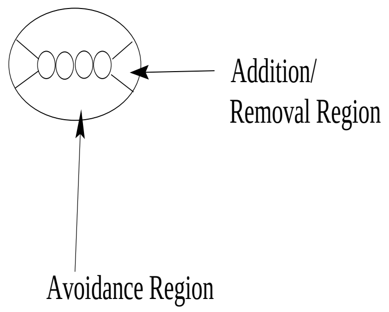


Figure 7.6.2: This gives the addition and subtraction regions of interaction of a khepera robot approaching a cluster of "seeds".

These clusters have two ways of being modified. These are having a seed added to the end of the cluster, or having a seed removed from the cluster. We plot the ratio of probabilities that these will occur reported in (Martinoli et al. 1999) in Figure 7.6.3.

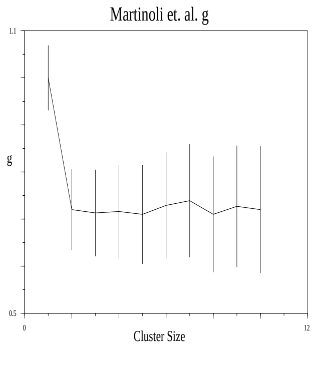


Figure 7.6.3: g as reported in (Martinoli et al. 1999). The error is a result of estimations of the numerical values for h and f at each point.

This is clearly not monotonically decreasing, and so therefore *does not* satisfy our condition for *monotonic clustering.* Indeed, this is precisely what is reported in

(Martinoli et al. 1999), as the authors note that the behavior is somewhat driven by a probability of clusters to disappear, after which time an irreversible system change has occurred. The performance of the system is expected to require long times to converge as the time required to remove the larger clusters and add their constituent parts to the smaller clusters would seem to be an exponential function of the length of the cluster. Indeed, this is what is observed. We would characterize this behavior as *pseudoclustering*, in which no direct clustering behavior is responsible for the clustering itself, but it is rather the effect of irreversible random events which may be built upon by further irreversible random events of the same type.

7.7. Proper Swarm Engineering

In this brief Section, we report on the use of a group of multiple robots in creating predefined distributions of clusters.

A particularly simple system to create is one of two clusters in which all the pucks are initially in one cluster and end up in a predefined proportion in a second cluster.

Suppose that we have a total of N pucks, and we wish to create a system of pucks which has N_1 pucks in the larger cluster and N_2 pucks in the smaller cluster. As initial conditions are easy to control, we assume that the initial distribution is an easily generated structure.

Now, we suppose that we have two linear regions l_1 and l_2 which meet at a minimum of g for the two clusters. We assume also that the robot has an accurate method of determining the size of the cluster, deferring investigation of limited ability. Without a loss of generality, we may assume that $g\,(N=1)$ while $g\,(N=0)=0.5$. If this is the case, and we wish to have the system settle at N_1 and $N_2 = N - N_1$ then we must have

$$\frac{1-g_{eq}}{N_1} = m_2 \tag{7.7.1}$$

and

$$\frac{g_{eq}-0.5}{N1} = m_1 \tag{7.7.2}$$

These yield the linear equations

$$l_1 : y = \left(\frac{g_{eq}-0.5}{N_1}\right)x + 0.5 \tag{7.7.3}$$

$$l_2 : y = \left(\frac{1-g_{eq}}{N_1}\right)x + \frac{g_{eq}N - N_2}{N_1} \tag{7.7.4}$$

giving a minimum at

$$\frac{N\left(1-g_{eq}\right)-0.5N_1}{\left(1.5-2g_{eq}\right)}=x \tag{7.7.5}$$

so that if $g_{eq}=0.45$ then

$$\frac{1.1N-N_1}{1.2}=x \tag{7.7.6}$$

Note that now the linear equations are given by

$$l_1 : y=\left(\frac{-0.05}{N_1}\right)x+0.5 \tag{7.7.7}$$

$$l_2 : y=\left(\frac{0.55}{N_1}\right)x+\frac{0.45N-N_2}{N_1} \tag{7.7.8}$$

Thus, we have a design condition. If we let the $f=1$ for all cases, which means that the robot will pick up every puck it encounters, and let the probability to drop be a function of size as prescribed in the above linear equations. We graph the evolution of a two-cluster system initialized with one cluster containing all the pucks and the second containing none.

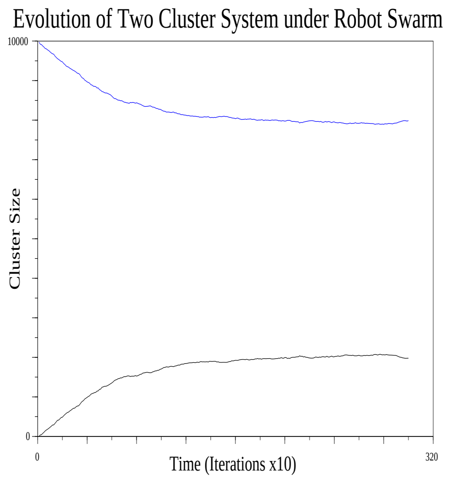


Figure 7.7.1: This figure illustrates the use of a swarm engineering technique designed to produce two clusters of a prespecified 4:1 ratio.

In this Figure, the sizes of the clusters are plotted as a function of time. Initially, all pucks are in the larger cluster. However, very quickly, the clusters begin exchanging pucks, and continue to have a net flow to the smaller cluster until the desired system is produced. Generalizations of this technique using a rough binary estimation of size, or a noisy size estimation produce similar results, with no significant degradation (up to 10% error). Most importantly, this can be easily generalized to multiple clusters, to clusters of differing size, and of a specific spatial design. These further possibilities require more investigation.

7.8. Discussion and Concluding Remarks

Groups of cooperating robots have the potential to carry out a wide variety of tasks which would seem to require intelligence in order to be completed, while utilizing little or no intelligence at the single agent level. In the natural world, these sytems have been shown to be capable of carrying out tasks which seem to be complex and require rudimentary intelligence. Indeed, they have been shown to be more than a simple sum of the parts, forming an entirely new whole.

In this work we have begun to turn the design paradigm on its head. What we've done is ask what the general conditions must be in order to design a system of puck clusterers which will cluster pucks into a single cluster. Once this condition has been satisfied, it is expected that any design paradigm can be shown to lead to the robust clustering of the system. This condition does not take into account the efficiency of the algorithm, but instead deals only with the final global design goal. Further work must be completed in order to verify these expectations with physical robots.

What this does is allow a great generality and freedom in our design. We may now use exceptionally simple designs to complete the same global design goal. This allows us to build robots with extremely simple algorithms which will complete our task. What we have then achieved is a general methodology for creating extremely simple algorithms which yield a desired global behavior. Indeed, we have begun to unravel the idea of *emergent properties* and have replaced them with *global design properties* which may be seen to be different, in many cases, from those simple properties built into the robot. As an example, we may see that there is nothing in the simple robot which picks up pucks and shuttles them from group to group which would indicate that this would yield a global clustering behavior. However, this is indeed the end result, despite the fact not being immediately obvious without the advantage of hindsight or deep insight.

Further work in this area may tackle the use of stigmergic communication as a minimal necessary means of completing tasks. In order for this to be required, however, it must be necessary to recruit more than one robot in order to complete a task. Any less than this amounts simply to an improvement in efficiency. Now that a good understanding of the minimal requirement for generating more efficient algorithms has been established, it is advantagous to undertake a study of the tradeoff between the efficiency of the collective, and the design complexity. This may yield optimally complex and efficient designs, using a minimum of collective processing, but yielding a maximally efficient collective. We do not tackle the problem here, but rather leave it as possible future research.

Part 3

Swarm Engineering for TSP

CHAPTER 8

Previous Swarm-Based Work on TSP and Combinatorial Optimization

The travelling salesman problem (TSP) is deceptively simple in its statement giving the illusion of having a simple proof. However, just as Fermat's last theorem is simply stated, but extremely difficult to prove, so too is TSP. Much of the work using software swarms has centered around TSP and related problems, such as routing problems in telecommunication systems. We review in this Chapter a small subset of the work done in this area.

Dorigo et al.'s seminal paper (Dorigo et al. 1996) on ant colony optimization provided a first attempt to use software swarms to solve a practial problem. In this paper, Dorigo et al. demonstrated the design of a system of agents travelling over a graph of nodes and edges which was capable of generating good solutions to small TSP instantiations. Moreover, it was demonstrated how one might generalize the system to a method of carrying out a variety of other optimizations.

Dorigo et al. (Dorigo et al. 1996) drew their inspiration from experiments done by biologists (Deneubourg et al. 1983, 1989, Goss et al. 1990). In studying ant systems it was noted that if a swarm of ants initially had created a stable path along which a group of ants was happily travelling back and forth, and an obstacle was placed in the way, as in Figure 8.1, the trail of ants would adapt by finding the shortest route around the obstacle.

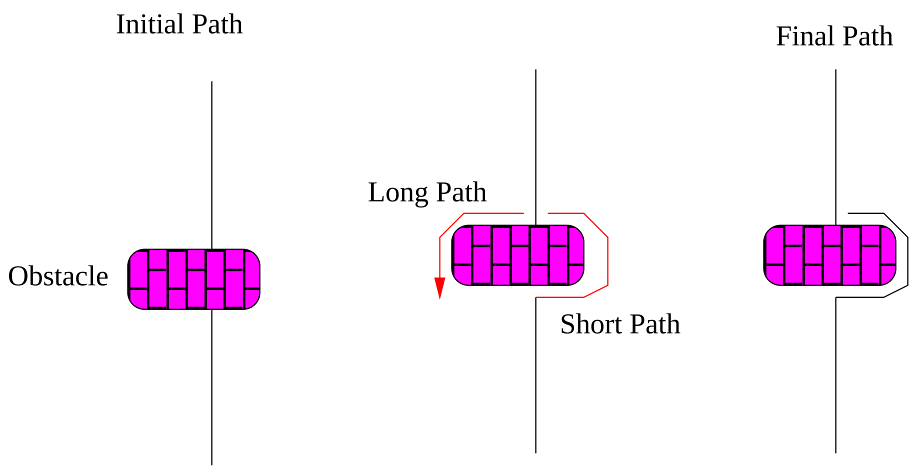


Figure 8.1: When an obstacle is placed in the way of a stable ant trail, the trail adapts to find the shortest route around the obstacle.

The question then is how this is accomplished by ants. Ants do not seem to have any ability to know whether or not one route or the other is longer, and their capability for communication seems to be limited by locality. However, the swarm of ants seems to be able to adapt in a way that indicates some measure of intelligence[1].

The way in which this seems to be handled is through a form of bidirectional feedback (Dorigo et al. 1991). That is, ants travelling in both directions seem to employ a behavior which contributes to small differences in signals obtained by temporal differences in the arrival of ants travelling along one or the other of the prescribed paths. Since ants travelling along both directions will generally equally take one or the other paths around the obstacle, the first one to make it to the other side will bias the ants making later decisions. This is an unstable equilibrium, and the initial bias will be increased as the addition of relatively more pheromone on one side of the obstacle will reinforce itself on both ends. Interestingly, the same mechanism leads to lock in of one or the other side if the obstacle is placed symmetrically across the line of ants, as a random fluctuation can be fed back to cause the domination of one path.

The ant system is based on this type of communication, but employs “enhanced ants”. These ants

1. have memory of where they’ve been
2. will not be completely blind

[1]One might argue that this is not intelligence at all, but rather dynamics. Evolution may have incorporated a dynamics into the system that allows it to deal with this commonplace occurrance.

3. will update the amount of pheromone deposited on the trail based on the distance they travelled.

Each ant is initialized to begin at a randomly chosen city. Each link between cities is initialized to have zero pheromone. As each ant moves, it calculates the total distance travelled, and will place a bit of pheromone the amount of which is inversely proportional to the whole distance on the path it travelled at the completion of a path. The probability of moving from city to city will be an increasing function of the pheromone and a decreasing function of the distance between the ant's current step and the proposed next step.

After each complete iteration, the pheromone is allowed to evaporate from each link, lowering the probability of those links not travelled. This allows long paths that have been travelled inadvertantly to become less probable as a path to take over many iterations.

The algorithm is shown to perform well on a variety of problems, including a 30 city problem known as Oliver30 for which the algorithm extends the current best to a new minimum distance path. Dorigo et al. provide an extensive investigation of the effects of the various parameters, but do not provide any informative theoretical analysis of the behavior of the system.

Extensions of this work are plentiful, typically coming from Dorigo and collegues (Dorigo et al. 1996a,1997,1997). These include several minor modifications to the basic system, meant to increase the capabilities of the system's search without the loss of the ability to find good solutions.

One particularly interesting extension of this work was created by Schoonderwoerd et al. (Schoonderwoerd et al. 1997). In this work a group of mobile "ants" were used to carry out dynamic load balancing on a telecommunication network. This was accomplished in an interesting way. In this work, the network is a directed symmetric graph with each node characterized by a capacity, a spare capacity, and a routing table, and each link having an infinite capacity. The routing table gives the probability that an agent will want to take a given link, in much the same way as the previous work used pheremones to bias the choices of ants. Ants are launched from random nodes and directed to random nodes. Each ant records the time required for its partial trip from one node to the next and updates the routing table entries at nodes it passes for those parts of the path going *the other way.* This is done by updating the pheremone levels on the routing table, in such a way as to preserve the normalization of a probability of 1 of choosing a given path. This and related work has laid a promising foundation for what one might hope will be a significantly more robust and dynamic standard method of carrying out traffic routing on communications networks. Although swarm engineering techniques will fit nicely in this framework, we do not attempt this problem here.

CHAPTER 9

Swarm Engineering for TSP

As we briefly stated in Chapter 1, several researchers have been making progress in the area of global design using local design rules. This work is attempting to generate an understanding of how to attack problems using swarms while employing techniques that are less anecdotal and more general than those used previously. William Spears and colleagues (Spears and Gordon 1999, Gordon and Spears 99). have begun advocating the use of techniques based on *artificial physics*. This is a term borrowed from artificial life and refers to a brand of physics based entirely on the imagination, implemented in artificial environments that dominate "physical" artificial life work. In this paradigm, a set of behaviors are set up, which may be implemented by artificial agents using available actuators, and which must be obeyed by each element of the swarm. The approach seeks to generate a set of behaviors which may be implemented in both simulation and on real robots, based on a set of rules which may be easily simulated, and later uploaded to a group of real mobile robots.

In this Chapter we present the application of swarm engineering techniques to the solution of the travelling salesman problem. Our goal is to generate a swarm condition which will allow the completion of the task using a swarm of mobile agents. We demonstrate that although the basic algorithms of Dorigo et al. satisfy the swarm condition, they are unstable under random fluctuations. This means that with a small number of agents compared to the size of the problem set, the probability of generating a suboptimal tour is nontrivial. Finally, we present work on a novel method of handling TSP based on the swarm condition and designed in such a way as to avoid the problems inherant in the methods of Dorigo et al.

9.1. Swarm Criterion

In general, the goal of swarm engineering is to provide a scaffolding upon which to build a set of behaviors each agent will undertake. This scaffolding should provide enough structure so that any behavior which fits naturally in this design will accomplish the task.

9.1.1. Initial Definitions: One way of stating the goal of the travelling salesman problem is that one wishes to find a path through a set of locations such that the distance travelled is the smallest possible given the set of locations. The main weakness with this statement of the problem is that it does not yield to a solution using a swarm of agents. Rather, it is very much centered around the generation of a single path, a single piece of a vast puzzle.

Swarms, by definition, have an effect as a group. Thus, we must look for a second statement of the problem which is amenable to solution by swarms. We choose to restate the problem this way.

Given a set of N nodes $\{n_i\}_{i=1}^{N}$, there are $\frac{(N-1)!}{2}$ different paths possible which completely travel through each node, ending up at the beginning node again, without passing through any one node twice. There is an inherant order from shortest to longest of these paths. Thus, we seek a method of removing paths from the longest end of the list to the shortest end until the only path left is the shortest.

Thus, the group of path lengths is the interesting quantity here, and we wish to create a system under which this set of paths will evolve, eventually generating a single path which is the smallest path among those generated by the set of locations. Given the set of nodes, we have a set of connections $\{c_i\}_{i=1}^{N(N-1)}$ between nodes. These represent one part of a possible path. Each member of the swarm might be able to travel on these connections while travelling from node to node. We may model such travel as a stochastic process in which a particular node is chosen by an agent and travelled to with some probability p_{ij}, where i represents the agent's current node and j represents the next node. The object of the swarm's interaction with the network of nodes is to modify these probabilities in such a way as to effectively (or deliberately) remove all links except those contributing to the minimum path.

We restate this formally in the next section, and it becomes our swarm condition.

9.1.2. Swarm evolutionary conditions and actions. Each path has a total probability of being traversed equal to

$$p_\lambda = \prod_{i=1}^{N-1} p_{i(i+1)} \tag{9.1.1}$$

where $\lambda = \left\{n_{\lambda(1)}, n_{\lambda(2)}, \cdots, n_{\lambda(N)}\right\}$. We can therefore state the following set of swarm conditions.

Swarm Criteria

1. (Initialization) $p_\lambda < p_{\lambda'}$ for all paths λ and λ' with the distance of λ larger than that of λ'.
2. (Increasing dispersion) The swarm should affect the probabilities in such a way that as $t \nearrow$, $|p_\lambda - p_{\lambda'}| \nearrow$
3. (Absolute best probability) The swarm should also affect the probability of passing over the best path. As $t \nearrow$, $p_{\lambda_{min}} \nearrow$ where $p_{\lambda_{min}}$ is the probability of the minimum path being travelled.
4. (Robust against error.) The system must be robust to sampling error.

The swarm is capable of doing two things in its travels over a network of cities and paths.

1. A swarm is capable of carrying out the travel and evaluating the path it took.
2. A swarm is capable of modifying the probabilities of the links it passes over.

The way in which modification of the probability of travelling over a given link is done may be quite general. Many researchers have done this with using a link-specific marker, many times called a pheromone. We now illustrate formally the effect of the use of a paradigm that uses a marker or other probability measure.

When travelling over a trail, a real ant will leave pheromone scattered on the trail, in response to cues both internal and external. Simulated TSP algorithms also employ a pheromone-based update method. In these methods, pheromone is typically left on the trail as a function of the distance the agent has travelled.

$$\eta = f(d) \tag{9.1.2}$$

where d is the total distance travelled by the agent. Thus, the amount of pheromone left on a particular link is given by

$$ph = N \sum_c f(d_c)\, p_c \tag{9.1.3}$$

where p_c represents the probability of a given agent travelling over a circuit c of distance d_c and N is the number of agents passing through the system. The sum is assumed to run over all circuits c containing the given link.

Let

$$Z = \sum_c p_c \tag{9.1.4}$$

We define the average inverse distance travelled over the entire set of complete paths which use the link in question

$$\left\langle \frac{1}{d} \right\rangle = \frac{1}{Z} \sum_{c} \frac{p_c}{d_c} \; . \tag{9.1.5}$$

Then the amount of pheromone makes

$$ph = NZ \left\langle \frac{1}{d} \right\rangle \; . \tag{9.1.6}$$

where $f\left(d_c\right)$ is $\frac{1}{d_c}$.

In actuality, pheromone merely causes a change in probability of an agent passing through a given link. Thus,

$$\frac{dp_l}{dt} \simeq g\left(NZ \left\langle \frac{1}{d} \right\rangle\right) \tag{9.1.7}$$

where again d represents the average distance travelled through the given link and g represents the function of pheromone which indicates how a given amount of pheromone translates to a given probability.

Example: For instance, this may be given by

$$\frac{dp_l}{dt} = \alpha \left(\left\langle \frac{1}{d} \right\rangle - \left\langle \frac{1}{d_0} \right\rangle\right) \tag{9.1.8}$$

where α is some proportionality constant and $\left\langle \frac{1}{d_0} \right\rangle$ is the average of the inverse distances travelled by agents through the whole system, as opposed to those travelling through the link.

Now, suppose that we have a four city TSP problem as given in Figure 9.1.

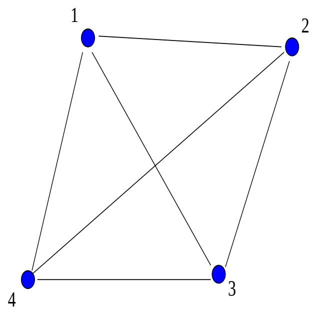


Figure 9.1: This figure gives an extremely small TSP problem.

We start this in the state that the probability of taking any one path between cities is equal. If this is the case, then we have the following equation for the amount of pheromone deposited on the link $1-2$.

$$ph_{12} = Np_{12}\left(\sum_{jk}\sigma_{jk} f\left(d_{12jk}\right) p_{2j}p_{jk}p_{k1}\right) + C \tag{9.1.9}$$

where

$$\sigma_{jk} = \begin{cases} 1 & \text{if } j \neq k \text{ and } j,k \in \{3,4\} \\ 0 & \text{otherwise} \end{cases} \tag{9.1.10}$$

This 2-d *tensor* describes the paths which may be considered. This gives us, if the level of pheromone is proportional to the rate of change of probability

$$\frac{dp_{12}}{dt} = Np_{12}\left(\sum_{jk}\sigma_{jk} f\left(d_{12jk}\right) p_{2j}p_{jk}p_{k1}\right) + C \tag{9.1.11}$$

giving stationary solutions or solutions to this in the approximation that ps are not changing much

$$p_{12} = e^{\alpha t} \tag{9.1.12}$$

where

$$\alpha = N\left(\sum_{jk}\sigma_{jk} f\left(d_{12jk}\right) p_{2j}p_{jk}p_{k1}\right) + C \tag{9.1.13}$$

with C possibly being some evaporation term or normalization term.

The most compelling part of this study is that such a system will generally increase or decrease depending on how the terms in equation (13) work out. If f is a nonnegative function, then the probabilities will never decrease, and will saturate (become 1) quickly. The resulting system is not selective. On the other hand, if the function f is negative, the probabilities will decrease, making the set of paths impassible by an agent.

We note also that the evolution of any given probability does not depend on the probability p_{12} itself, but merely on the ensemble of paths going through it.

We choose to model our f as

$$f = \frac{1}{d} - \frac{1}{d_{ave}} \tag{9.1.14}$$

where d_{ave} is the average distance of the complete paths taken by agents moving through the grid. Thus,

$$\alpha = N\left(\sum_{jk}\sigma_{jk}\left(\frac{p_{2j}p_{jk}p_{k1}}{d_{12jk}} - \frac{p_{2j}p_{jk}p_{k1}}{d_{ave}}\right)\right). \tag{9.1.15}$$

As long as this is negative, the link will decrease in probability. If it is positive, it will increase in probability. If there is a single path, the probabilities will remain stable.

9.1.3. Sampling Constraints. In general, it is very difficult, if not impossible to generate d_{ave} or d_{12}. The calculation of these quantities requires that one obtains the result for a large number of paths, comparable to the number of possible paths in the search space. Practical approximations are nonetheless approximations, and are limited by definition. Thus, any system attempting to approximate the previous design will be limited in its ability to faithfully reproduce the equations presented.

We return to this point later in this work, illustrating how our results are affected.

9.2. Swarm Engineering Case Studies

Dorigo et al's seminal work on swarm based optimization initially centered around the use of computational agents he called 'ants' to solve the travelling salesman problem. This work provided both inspiration and a basis for later work on the use of ant systems in the solution of combinatorial problems. In this section, we apply our previous theory to the solutions initially obtained in an effort to begin to understand why these algorithms performed as they did.

9.2.1. Initial ant-based optimization. Dorigo's initial ant-based optimization system makes use of discrete agents which travel over a graph made up of nodes and symmetric links between nodes. Each node represents a location, and each link represents the distance between nodes. Nodes are infused with a quantity of a "substance" called pheromone which influences the ant's likelihood of travelling along a given link. Ants are assumed to continually travel, keeping a record of the path they take. Ants are not allowed to retrace their path, and so cannot take a link that leads to a node they have already visited.

At each node, an ant must decide which path to take. This is done according to the amount of pheromone on each of the available links. The probability of an ant taking a particular node is given by

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \tag{9.2.1}$$

where τ_{ij} represents the amount of pheromone on a link between node i and j, η represents a *visibility* parameter, the sum l runs over all links from node i, and α and β are adjustable parameters. Note that the sum of these is set to one, meaning that the ant will always take *some path*, a condition we relax in Section 4.

Each complete path of all ants constitutes a complete iteration. At the completion of each iteration, each ant updates the pheromone levels on each link as indicated by equation (9.2.2).

(9.2.2)
$$\Delta\tau_{ij} = \begin{cases} Q/L^k(t) & \text{if } (i,j) \in T^k(t) \\ 0 & \text{if } (i,j) \notin T^k(t) \end{cases}$$

$T^k(t)$ is the tour undertaken by the ant at iteration t, and $L^k(t)$ is its length. Thus,

(9.2.3)
$$p_{ij}(t+\Delta t) = \frac{\left[\tau_{ij}(t) + \sum_{c_{ij}} \frac{Q(\Delta t)}{L_{c_{ij}}}\right]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_l \left(\left[\tau_{il}(t) + \sum_{c_{il}} \frac{Q(\Delta t)}{L_{c_{il}}}\right]^\alpha \cdot [\eta_{il}]^\beta\right)}$$

where c runs over all tours passing through this link. We assume that Q may be modelled as a function of time and that Q is continuous and $Q(0) = 0$. Thus, the difference between the two is

(9.2.4)
$$p_{ij}(t+\Delta t) - p_{ij}(t) = \frac{\left[\tau_{ij}(t) + \sum_{c_{ij}} \frac{Q(\Delta t)}{L_{c_{ij}}}\right]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_l \left(\left[\tau_{il}(t) + \sum_{c_{il}} \frac{Q(\Delta t)}{L_{c_{il}}}\right]^\alpha \cdot [\eta_{il}]^\beta\right)} - \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_l \left([\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta\right)}.$$

Now, if $Q \ll \tau$ then

(9.2.5)
$$p_{ij}(t+\Delta t) - p_{ij}(t) \cong \frac{\alpha \sum_l \sum_{c_{ij}} Q(\Delta t) [\eta_{il}]^\beta [\eta_{ij}]^\beta \left(\frac{[\tau_{il}(t)]^\alpha}{L_{c_{ij}}} - \frac{[\tau_{ij}(t)]^\alpha}{L_{c_{il}}}\right)}{\sum_l \left(\left[\tau_{il}(t) + \sum_{c_{il}} \frac{Q(\Delta t)}{L_{c_{il}}}\right]^\alpha \cdot [\eta_{il}]^\beta\right) \sum_l \left([\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta\right)}.$$

This leads us to

$$p_{ij}(t+\Delta t) - p_{ij}(t) \approx \alpha \sum_l \sum_{c_{ij}} Q(\Delta t) [\eta_{il}]^\beta [\eta_{ij}]^\beta \left(\frac{\tau_{ij}(t)[\tau_{il}(t)]^\alpha}{L_{c_{ij}}} - \frac{\tau_{il}(t)[\tau_{ij}(t)]^\alpha}{L_{c_{il}}}\right)$$

which, in the limit gives us that

(9.2.6)
$$\frac{\partial p_{ij}}{\partial t} = \alpha \frac{\partial Q}{\partial t} [\eta_{ij}]^\beta \sum_l \sum_{c_{ij}} [\eta_{il}]^\beta \left(\frac{\tau_{ij}(t)[\tau_{il}(t)]^\alpha}{L_{c_{ij}}} - \frac{\tau_{il}(t)[\tau_{ij}(t)]^\alpha}{L_{c_{il}}}\right).$$

Focusing for a moment on

$$\sum_l \sum_{c_{ij}} [\eta_{il}]^\beta \left(\frac{\tau_{ij}(t)[\tau_{il}(t)]^\alpha}{L_{c_{ij}}} - \frac{\tau_{il}(t)[\tau_{ij}(t)]^\alpha}{L_{c_{il}}}\right)$$

, we can rewrite this as

$$\left(\sum_l \left\{[\eta_{il}]^\beta [\tau_{il}(t)]^\alpha\right\} \sum_{c_{ij}} \left(\frac{\tau_{ij}(t)}{L_{c_{ij}}}\right) - \sum_{c_{ij}} [\tau_{ij}(t)]^\alpha \sum_l \left(\frac{\tau_{il}(t)[\eta_{il}]^\beta}{L_{c_{il}}}\right)\right).$$

For simplicity, we may assign $\eta_{ij} = 1$ yielding

$$\left(\sum_{l}\left[\tau_{il}\left(t\right)\right]^{\alpha}\sum_{c_{ij}}\left(\frac{\tau_{ij}\left(t\right)}{L_{c_{ij}}}\right)-\sum_{c_{ij}}\left[\tau_{ij}\left(t\right)\right]^{\alpha}\sum_{l}\left(\frac{\tau_{il}\left(t\right)}{L_{c_{il}}}\right)\right).$$

In principle, if the paths going through a link were long compared to other paths, the first term would tend to be smaller than the second, while if they were short, the second would tend to be smaller. This means that this system, would tend to satisfy our swarm condition, and should lead to a system that creates a shortest path.

However in real systems, one cannot know the values of these sums accurately, and there will always be an inherant error. Thus, the system will evolve under conditions that are not quite accurately predicted by these equations. The amount of change of pheromone, then under an error will determine the stability of the system. In a real ant-based system, the amount of change will be determined by

$$\left(\sum_{l}\left[\tau_{il}\left(t\right)\right]^{\alpha}\sum_{a_{ij}}\left(\frac{\tau_{ij}\left(t\right)}{L_{a_{ij}}}\right)-\sum_{c_{ij}}\left[\tau_{ij}\left(t\right)\right]^{\alpha}\sum_{l}\left(\frac{\tau_{il}\left(t\right)}{L_{a_{il}}}\right)\right).$$

where a represents the agent in the system over which the sums are actually taken. Now, if a long path has an initially large estimate of the first term, τ will increase along that link. This increase will in turn bias the ants from the second term to the first term, and if this bias is large enough, will tend to cause a link that should not increase to increase in probability, locking in a poor path. Indeed, this is precisely what is reported. Thus, although the system obeys our first three swarm engineering criteria, it fails to satisfy the fourth, which in practice significantly reduces the capability of the system.

9.2.2. Evaporation. Dorigo et al. propose a modification to the system which includes the use of pheromone evaporation. This is an attempt to provide a restorative force which will satisfy condition 4. In this case, we have

$$p_{ij}\left(t+\Delta t\right)=\frac{\left[\tau_{ij}\left(t\right)\left(1-\mu\right)+\sum_{c}\frac{Q(\Delta t)}{L_c}\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}{\sum_{l}\left[\tau_{il}\left(t\right)\left(1-\mu\right)+\sum_{c}\frac{Q(\Delta t)}{L_c}\right]^{\alpha}\cdot\left[\eta_{il}\right]^{\beta}}. \tag{9.2.7}$$

Here, again, we may assume that μ is a function of the change in time Δt which goes to zero when Δt goes to zero. Then

(9.2.8)

$$p_{ij}\left(t+\Delta t\right)-p_{ij}\left(t\right)=\frac{\left[\tau_{ij}\left(t\right)\left(1-\mu\right)+\sum_{c_{ij}}\frac{Q(\Delta t)}{L_{c_{ij}}}\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}{\sum_{l}\left(\left[\tau_{il}\left(t\right)\left(1-\mu\right)+\sum_{c_{il}}\frac{Q(\Delta t)}{L_{c_{il}}}\right]^{\alpha}\cdot\left[\eta_{il}\right]^{\beta}\right)}-\frac{\left[\tau_{ij}\left(t\right)\right]^{\alpha}\cdot\left[\eta_{ij}\right]^{\beta}}{\sum_{l}\left(\left[\tau_{il}\left(t\right)\right]^{\alpha}\cdot\left[\eta_{il}\right]^{\beta}\right)}$$

if again $Q \ll \tau$, then

(9.2.9)

$$p_{ij}(t+\Delta t) - p_{ij}(t) \cong \frac{\alpha \sum_l \sum_{c_{ij}} Q(\Delta t)\, [\eta_{il}]^\beta\, [\eta_{ij}]^\beta \left(\frac{\tau_{ij}(1-\mu)[\tau_{il}(t)]^\alpha}{L_{c_{ij}}} - \frac{\tau_{il}(1-\mu)[\tau_{ij}(t)]^\alpha}{L_{c_{il}}} \right)}{\sum_l \left(\left[\tau_{il}(t)(1-\mu) + \sum_{c_{il}} \frac{Q(\Delta t)}{L_{c_{il}}} \right]^\alpha \cdot [\eta_{il}]^\beta \right) \sum_l \left([\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta \right)} .$$

In this case, the damping term depresses the change in probability, controlling the random fluctuations. This allows the system to be tuned with an appropriatly chosen μ to a rate of change in probability that is small enough to manage and to relieve random fluctuations. Dorigo et al. report that this is sufficient to limit the effects of random fluctutation. However, there is a tradeoff between the strength of the evaporation and the effectiveness of the algorithm. If the evaporation term is too large, the denominator in equation (3.9) will also become small, making the size of the fluctuation increase. If this is the case, the system will never be able to lock into the desired minimum distance, but rather will continue to endlessly fluctuate.

9.3. Probabalistic Agents

The main weakness of ant-based systems is that they are vulnerable to random effects. This has led to a variety of different algorithms designed to minimize these effects. The weakness stems from the fact that the agent must make a decision about which path to take at every iteration. This is in contrast to first making a decision about whether or not to take a link in the path at all.

One might imagine a *geddanken* in which electrons are fired through an absorptive medium from each of the cities toward each of the other cities. If one were riding along with these electrons, one would note that the number of companion electrons travelling with the electron providing the ride would decrease as an inverse exponential of the distance. Thus, we would have

(9.3.1) $$N = N_0 e^{-\lambda d}$$

where λ is some characteristic distance, and d is some distance. Were one to send the electrons along to the next city, one would find a similar phenomenon occurring. This would happen again and again, until the final electrons arrived at their destination, with numbers reflecting the distance travelled. Thus,

(9.3.2) $$N = N_0 e^{-\lambda d_t}$$

with d_t representing the total distance travelled.

We choose as model agents for our swarm agents as improved ‘electrons’. Each agent has a probability of travelling from one node to another given by equation

(9.3.1). At each node, each agent duplicates itself so as to provide enough total agents to travel to each node not travelled to on the path it has already taken. Each new agent will then pick up where the first agent left off, travelling to the new nodes in the same way as the previous agent did in the last iteration. Note that if one agent is initialized at each node, and the probability travelling is 1 (λ is 0), each possible path will be travelled. The behavior of an agent and its duplicates is depicted in Figure 9.2.

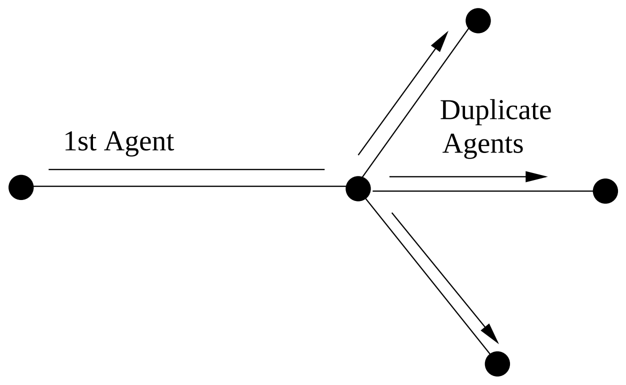


Figure 9.2: This depicts two parts of one iteration of the agent's motion on a map.

Note that in the first part of the iteration, one agent exists. In the second part, three agents exist. The number of agents equals the number of possible paths one might take while going through the first link.

It is straightforward to verify that the paths travelled by the agents in this system occur with an exponentially decreasing probability commensurate with equation (9.3.2). We illustrate this in Figure 9.3,

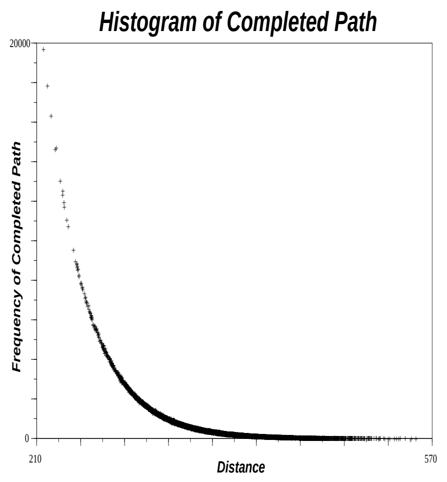


Figure 9.3: This illustrates the distribution of path lengths obtained from agents which completed a tour of ten randomly placed nodes.

in which 10^5 iterations of a sample graph have been run.

This method has two main flaws. Any attempt to handle large graphs immediately causes one of two outcomes:

1. A virtual explosion in the number of agents in the intermediate stages of the search.
2. No agents completing tours, as a result of the extremely low probability of any agent completing a tour given the low pairwise probabilities required to limit the explosion of agents.

Moreover, the method is not a swarm at all. Indeed, it uses the predefined system to find the smallest path, using a large number of trials only to verify that the correct path has indeed been found.

A swarm has been defined in Chapter 1 as a group of communicating agents exhibiting emergent behavior. We employ a swarm-based method which derives from these probabilistic agents. We initialize the system with one agent. This agent has a predefined initial path $\{a_1, ..., a_N, a_1\}$ with some distance d_a, which becomes the first estimated minimum distance.

All probabilities are initialized according to

$$p_{ij} = e^{-\lambda d_{ij}} \quad . \tag{9.3.3}$$

This biases the algorithm towards a small number of completing paths, with a small average distance. At each iteration, this probability is reduced according to

$$p_{ij} \mapsto \upsilon p_{ij} \tag{9.3.4}$$

where υ is some positive number smaller than 1.

In order to generate new tours for sampling, the current tour is shifted to the left by M elements, i.e.,

$$\{a_1, \dots, a_N, a_1\} \mapsto \{a_{M+1}, \dots, a_N, *, *, \dots, *\} \tag{9.3.5}$$

where $*$ indicates an unassigned node in the tour. The last M elements are chosen according to the guidelines given for probabalistic agents above, yielding many new paths in the process. Thus, the probability distribution of paths resulting from this method is

$$\{p_d | p_d = p_1 \cdots p_M\} \tag{9.3.6}$$

where the probabilities are the given according to choices for last elments. At every step, the new paths become truncated and duplicated, yielding a small number of new final paths.

If we define d_0 to be the current estimate of the minimum distance, we denote the update of transition probabilities by

$$\frac{dp_{ij}}{dt} = \alpha \left(\frac{1}{d_{e_{ij}}} - \frac{1}{d_0} \right) \tag{9.3.7}$$

where $d_{e_{ij}}$ is given by

$$d_{e_{ij}} = \left(\sum_c \left[\frac{1}{d_c} \right] \right)^{-1} \tag{9.3.8}$$

and the sum is over all tours c which travel through a given link. Let us examine this in terms of our swarm condition.

- The initial state is exactly as desired in the swarm engineering condition.
- As more iterations pass, two types of updates occur. First, the probabilities are updated by multiplying a factor (for instance 0.99) to all link probabilities. This lowers the probability of the group of paths. Next, when an agent completes the path, the links along that path are updated by adding the multiplying by quantity $\exp\left(\frac{1}{d}\right)$ to each of the link probabilities along the path. This increases the probability of going on the link, thereby increasing the probability of going along any small distance path going through this link. This satisfies condition 2.

- The probability of visiting a nearby link is limited by a small number of links, indicating that the previous problems of both explosion and low probability have been solved. Moreover, as shown above, the smaller paths will be visited more often than the larger paths. Thus, those paths with smaller portions contributing will be selected more often than those with large portions selected.
- The longer distance paths have a decreasing probability which decreases at a larger rate than the shorter distance paths. The links contributing to these paths will have a smaller number of signals completing, and a smaller positive controbution per signal. Thus condition 3 is satisfied.
- As the best path is always of distance smaller than or equal to the current estimate d_0, the probability of this and all other paths decreases steadily. However, the system is reinitialized with the new estimate of the minimum distance, increasing the rate of decrease of long paths, but holding steady that of the minimum distance path.
- Finally, errors in sampling are of a different kind than previous errors. It is unlikely to lock in a path that is suboptimal, unless the rate of probability alteration is extremely high, as new paths may then be found easily which are superior. These will lead to a reduction in the relative probability of travelling over the suboptimal path. Since

$$p\,(t+\Delta t) = p\,(t) + \Delta\alpha\left(\frac{1}{d_{comp}} - \frac{1}{d_{0_{est}}}\right) \tag{9.3.9}$$

  in practice, it is impossible have an estimated minimum distance $d_{0_{est}}$ smaller than the actual minimum. Thus, the probability of all paths will initially decrease, with the rate of decrease changing with each new better distance. The relative increases in rate of probability decrease is different for links of differing average distance, with those links which contribute to a small total path having a smaller acceleration than those links contributing to large paths.

We test these on three paths given in the TSPLIB, a standard set of travelling salesman test problems. In each case, the initial probabilities are given by $p_{ij} = e^{0.001 d_{ij}}$ and the initial path is $\{1, 2, \ldots, N, 1\}$ where the path has N nodes. In each case, the minimum is found with sufficient time. As a comprehensive analysis of the performance is beyond the scope of this chapter, we present only several minimum distances found on small paths. The results are given in Table 9.1.

TSP Problem	Minimum Distance	Distance Obtained	Iterations (Average)
Burma (14)	3323.0	3323.0	47.7
Ulysses (16)	6859.0	6859.0	1000.2
Ulysses (22)	7013.0	7013.0	2194.2

Table 9.1: This table gives the performance of the swarm based on probabilistic agents on a set of three standard TSP test problems[1].

In each case, the minimum distance was obtained, indicating that the swarm condition illustrated above does indeed produce the desired outcome.

Thus, the use of this type of swarm-based computation would seem to be promising, as reliable algorithms may be constructed simply by conforming to a given condition.

9.4. Conclusions

In this Chapter, we have examined the design of a swarm-based optimization system using a rational technique to initially design a swarm condition. Swarms are then engineered based on this condition. Those that satisfy this condition are expected to produce a desired final outcome for the system, while those that do not are not expected to be able to complete the task in a satisfactory and robust manner.

Investigating some of the previous work in this light has illustrated that, although most of the swarm condition imposed for the TSP are satisfied, they are not robust under the effect of noise. Addition of evaporation served to limit the amount of noise available for changes in probability. However, an evaporation rate that is too high would seem to produce paths with great variability, as is observed in practice.

This work represents the third and final illustration of a general procedure for producing swarms of a predefined desired behavior. The correct development and application of such techniques would seem to be a necessary step for creating a process whereby swarms can be applied to practical problems and used to produce useful results. Further work will address the developement of a swarm-based way of guaging efficiency, with the hope for such work lying in the generation of swarm actions of a particular efficiency.

[1]The average number of iterations for the Ulysses TSP algorithm is somewhat artificially low, as the runs were terminated at 10000 iterations, and these averages reflect the completing runs.

CHAPTER 10

Discussion and Conclusions

This work provides a first look into a potentially fruitful field in which many researchers currently work under a number of different names. Swarm engineering refers to a general approach to generating swarms of independent agents capable of completing specific global tasks, while using robotic or software agents which are relatively simple to individually construct.

Swarm engineering departs from the design of most[1] multiple-agent systems in that our inital focus is on the general global characterization of the problem, rather than the local rules of the problem. The creation of a general set of conditions which allow the global goal to be achieved makes the design of individual agents, indeed groups of agents, somewhat simpler. However, this set of conditions can take a great deal of effort to build, and need not require the generation of an analytical solution to the problem. [2] Thus, any behavior which satisfies these conditions can be thought of as a solution to the problem. While this approach is similar to the approach of traditional robotics, swarm-based robotics has departed from this approach.

We studied in this work three different systems, and the use of swarm engineering techniques in these three systems. The three systems were very different in the nature of the problem and in the type of solution. Plume tracking is correctly done in an embodied system, and as we have seen, the differing properties of the systems can lead to differing dependencies on parts of the solution. However, an analytic generation of a minimal condition for swarm-based exploration of plumes is both difficult and unnecessary. This is not to imply that generating a general condition for swarm-based exploration is unnecessary, simply that an attempt to do so analytically is unecessaily difficult. With puck clustering systems, the situation is quite different. We have generated a set of conditions which, when satisfied, will allow the system to accomplish the goal. In this case, analytic methods have

[1]We do *not* claim that we are the first to approach problems in this way.

[2]In fact, although analytic solutions are desirable in many conditions, as illustrated in two of our three studies, their generation in some cases is unnecessarily complicated, and may hinder the work to be accomplished. It is our belief, however, that whenever possible and practical, such methods should be applied, as their study can often times lead to innovations that cannot be found in any other way.

proved to be quite useful. Finally, in the travelling salesman problem, we have used analytic methods as well to investigate the strength of the basic systems used in ant-based systems. These methods proved useful in generating an understanding of the weakness of the Dorigo ant-based algorithms, a new method for attacking the problem with encouraging results.

10.1. Plume Tracking

A plume tracking algorithm is one in which a robot physically finds its way from a point downstream from the source of a plume to the source, using information available in the direction of motion of the surrounding fluid and in the intensity of the plume. This problem has its uses in both the natural world and in the unnatural world.

We discovered quite accidentally that a simple robot endowed with sensors with a directional response could be fabricated in such a way as to produce a behavior that carries the robot to the source of the plume. Interestingly, this behavior was discovered accidentally, and indicated that the solution of the general problem could be created quite easily. The many different solutions available in nature serves to underscore this point, indicating that artificial solutions need not be as complicated or involved as one might think.

A second algorithm for finding the plume source was investigated. This algorithm, unlike the first, has the advantage that it is unlikely to lose the plume once it has found it. Moreover, it is extremely efficient in tracking the plume to its source. We investigated the adaptation of the algorithm using and excluding the use of wind direction information in a configuration that used the direction to indicate whether or not the robot was facing upwind. The algorithm was investigated using both real experimental data obtained using a laser-scanned plume source (Cowen and Chang 2000) and simulated data using a simulator produced by Jay Farrell (Farrell et al. 1999) and adapted in house to produce tunable concentrations. The performance on both data sets was comparable except in the use of concentration information in producing tracking behavior. Overall, the algorithm was found to be quite robust over a large number of different pertubations of the system.

Finally, we applied swarm engineering to a swarm designed to extend the sensitivity of the system of robots. This swarm's design was based on the condition that the system should make progress upstream, and should focus its search in areas of interest discovered by agents who might otherwise lose the source. The extension of the sensitivity of the system was demonstrated using a variety of performance measures, though the extension of the first arrival was only slight.

10.2. Puck Clustering

The application of swarm engineering techniques to puck clustering systems is quite straightforward. We may analytically investigate this system, and this was done. This required the restatement of the puck clustering problem in a format that allowed the focus to be on single agents rather than on global parameters, but also allowed the global goal to be achieved.

The basic swarm condition was generated in two ways. The first way was the consideration of the number of pucks in a pair of clusters of pucks. The general condition on the form of the ratio of the propensity to lose pucks f versus the propensity to gain pucks h was determined. Using this condition, it was shown that in a system of many clusters, the smallest of clusters always decreased in size, a requirement for the eventual absorption into a single cluster[3]. The same condition was again derived using a statistical physics approach in which local densities were considered assuming a diffusive environment. The same condition as was previously obtained was obtained in this consideration, allowing the extension of the system to real robot systems.

A brief investigation of the method of determining the relative efficiencies of two algorithms was carried out. This consisted of determining the minimal condition required to create a system that is less efficient than a given system. We found that if the ratio g was being compared to another ratio g', the latter would be more efficient if $g > g'$.

We enumerated the ways in which a non-monotonic system of two clusters containing two linear regions and a maximum or minimum would evolve. We found that such systems would in many cases create two stable clusters of pucks. Moreover, we discovered that in many cases both clusters had identical size, but in some cases, some clusters had differing sizes. Finally, we demonstrated that we could determine the sizes of the clusters ahead of time. This has exciting possible applications in distributed construction.

10.3. Travelling Salesman Problem

Our final investigation centered around the use of software swarms in the solution of the travelling salesman problem. This problem has been extensively investigated by other researchers, and our main contribution in this area was to establish that this was an incidence of swarm engineering and to put the work on firm mathematical footing. Once this had been accomplished, we observed the weaknesses of the systems from the equations that describe them.

[3]Of course, the size to which we refer is a time-averaged size

We proposed a second method of solving the travelling salesman problem which circumvents this difficulty. This method employs tunnelling agents, capable of dying along the way and of reproducing themselves. Left to travel to each node and not capable of dying, these agents could travel to all nodes over all paths if initiated by a single agent. An adaptation of this method was shown to satisfy the swarm engineering condition, and found good results when applied to three standard test problems, although more work on more test problems is required to validate the algorithm.

10.4. Summation

This thesis has provided an example of a global-first approach to swarm engineering. The set of techniques employed here may be extended to a wide variety of possible applications. In the author's view, one of the most exciting possible applications is distributed construction in difficult to reach areas inaccessible to humans. The use of such techniques might make it realistic to create structures allowing the harvesting of energy in both sustained mountaintop air currents or undersea currents. Construction of enclosures both undersea and in extraterrestrial locations may be possible, allowing colonization of both regions. Finally, the exploitation of such methods may allow search and rescue teams of robots to perform tasks more quickly and efficiently than may be capable for conventional search and rescue teams, allowing the combination of many individual robots' capabilities in the right times.

Bibliography

[beckers1994] Beckers R., Holland O., and Deneubourg J.L. *From local actions to global tasks: Stigmergy and collective robotics.* **Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems**, MIT Press, 1994.

[beer1995] Beer R. *A dynamical systems perspective on agent-environment interaction.* **Artificial Intelligence**, 72: 173-215, 1995.

[belanger1996] Belanger J. and Willis M. *Adaptive control of odor-guided locomotion: behavioral flexibility as an antidote to environmental unpredictability.* **Adaptive Behavior**. 4 (3/4): 217-253, 1996.

[bertsimas1993] Bertsimas D. and Tsitsiklis J. *Simulated Annealing.* **Statistical Science**, 8(1): 10-15, 1993.

[bonabeau1997] Bobabeau E., Sobkowski A., Theraulaz G., and Deneubourg J. *Adaptive task allocation inspired by a model of division of labor in social insects.* **Bio Computation and Emergent Computing**. Lundh D., Olsson B., and Narayanan A., eds. World Scientific: 36-45, 1997.

[bonabeau1998] Bonabeau E., Theraulaz G., Fourcassie V., and Deneubourg J. *Phase-ordering kinetics of cemetary organization in ants.* **Physical Review E**, 57 (4), 1998.

[bonabeau1997] Bonabeau E., Sobkowski A., Theraulaz G., and Deneubourg J. *Adaptive task allocation inspired by a model of division of labor in social insects.* **Bio Computation and Emergent Computing**, Lundh D., Olsson B., and Narayanan A., eds., 36-45, 1997.

[brooks1997] Brooks R. *From earwigs to humans.* **Robotics and Autonomous Systems**, 20: 291-304, 1997.

[brooks1993] Brooks R. and Mataric M. *Real Robots, Real Learning Problems*, **Robot Learning**, Connell J. and Mahadevan S., eds., Kluwer Academic Press, 193-213, 1993.

[brooks1986] Brooks, R. *A robust layerd control system for a mobile robot.* **IEEE J. Robotics and Automation**, RA-2 (1), 14-23, 1986.

[brown2000] Brown M. and Gordon D. *How resources and encounters affect the distribution of foraging activity in a seed-harvesting ant.* **Behavioral Ecology and Sociobiology**, 47: 195-203, 2000.

[camazine1990] Camazine S., Sneyd J., Jenkins M., and Murray D. *A mathematical model of self-organized pattern formation on the combs of honeybee colonies.* **Journal of Theoretical Biology**, 147, 553-571, 1990.

[cao1997] Cao Y., Fukanaga A. and Kahng A. *Cooperative mobile robotics: antecedents and directions.* **Autonomous Robots**: 4, 1-23, 1997.

[cohen1996] Cohen W. *Adaptive mapping and navigation by teams of simple robots.* **Robotics and Autonomous Systems.** 18:411-434, 1996.

[cowen2000] Cowen E. and Chang K. *A single camera coupled PTF-LIF technique*, submitted to **Experiments in Fluids**, 2000.

[crutchfield1994] Crutchfield J. *Is anything ever new? Considering emergence.* Cowan G., Pines D., and Meltzer D. (eds), **Complexity. Metaphors, Models, and Reality**. Menlo Park, CA: Addison-Wesley Publishing Company, 515-537, 1994.

[dejean1986] Dejean A., Masens D., Kanika K., Nsudi M., and Gunumina R. *Les termites et les fourmis, animaux dominants de la faune ddu sol de plusieurs formations forestieres et herbeuses du Zaire.* **Actes des Colleques Insects Sociaux**. 3:273-283, 1986.

[deneubourg1983] Deneubourg J., Pasteels J., and Verhaeghe J. *Probabilistic beaviour in ants: a strategy of errors?* **Journal of Theoretical Biology**, 105, 259-271, 1983.

[deneubourg1989] Deneoubourg J. and Goss S. *Collective patterns and decision-making.* **Ethology, Ecology, and Evolution**, 1:295-311, 1989.

[deneubourg1991] Deneubourg J., Goss S., Franks N., Sendova-Franks A., Detrain C., and Chretien L. *The dynamics of collective sorting. Robot-like ants and ant-like robots.* **From Animals to Animats**. J. A. Meyer and S. W. Wilson, eds. Cambridge, MA: MIT Press / Bradford Books, 1991.

[deveza1994] Deveza R., Thiel A., Russell A., Mackay-Sim A. *Odor Sensing for Robot Guidance.* **The International Journal of Robotics Research**, 13(3): 232-239, 1994.

[dickenson1997] Dickenson T., Walt D., White J., and Kauer J. *Generating sensor diversity through combinatorial polymer synthesis.* **Analytical Chemistry**, 69: 3413-3418, 1997.

[dittmer1996] Dittmer, K., Grasso, F. and Atema, Jelle. *Obstacles to Flow Produce Distinctive Patterns of Odor Dispersal on a Scale that could be Detected by Marine Animals*, **Biological Bulletin**, 191: 313-314, 1996.

[dittmer1995] Dittmer, K., Grasso, F. and Atema, Jelle. *Effects of Varying Plume Turbulence on Temporal Concentration Signals Available to Orienting Lobsters*, **Biological Bulletin**, 189: 232-233, 1995.

[dorigo1991] Dorigo M., Maniezzo V., and Colorni A. *Positive feedback as a search strategy.* Technical Report No. 91-016, Politecnico di Milano, Italy, 1991.

[dorigo1996] Dorigo M., Maniezzo V., and Colorni A. *The ant system: optimization by a colony of cooperating agents.* **IEEE Transactions on Systems, Man, and Cybernetics B**, 26: 29-41, 1996.

[dorigo1996a] Dorigo M. and Gambardella L. *A study of some properties of Ant-Q.* **Proceedings of the Fourth International Conference on PArallel Problem Solving from Nature**, Berlin: Springer-Verlag, 656-665, 1996.

[dorigo1997] Dorigo M. and Gambardella L. *Ant colony systems: a cooperative learning approach to the travelling salesman problem.* **IEEE Transactions on Evolutionary Computation**, 1 (1): 53-66, 1997.

[dorigo1997a] Dorigo M. and Gambardella L. *Ant colonies for the traveling salesman problem.* **BioSystems**, 43: 73-81, 1997.

[dorigo1999] Dorigo M., Di Caro G., and Gambardella L. *Ant Algorithms for Discrete Optimization.* **Artificial Life**, 5: 137-172, 1999.

[farrell1999] Farrell J., Murlis J., Long X., Li W., and Carde R. *Filament-Based Atmospheric Dispersion Model to Achieve Short Time-Scale Structure of Odor Plumes*,

Technical Report, Department of Electrical Engineering, Univ. of California, Riverside, October 11, 1999.

[fittkau1973] Fittkau E. and Klinge H. *On biomass and trophic structure of the central Amazonian rain forest ecosystem.* **Biotropica**. 5 (1) : 2-14, 1973.

[floreano1999] D. Floreano and J. Urzelai. *Evolution of Adaptive-Synapse Controllers.* D. Floreano et al. (Eds.) textbfAdvances in Artificial Life - ECAL99, Berlin: Springer Verlag. 1999.

[floreano1994] Floreano D. and Mondada F. *Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot.* Cliff D., Husbands P., Meyer J.-A., and Wilson S. (Eds.), **From Animals to Animats III**, Cambridge, MA: MIT Press, 1994.

[franks1992] Franks N., Wilby A., Silverman B., Tofts C. *Self organizing nest construction in ants: sophisticated building by blind bulldozing.* **Animal Behavior**, 44, 357-375, 1992.

[freund1995] Freund M. and Lewis N. *A chemically diverse conducting polymer-based "electronic nose".* **Proc. Natl. Acad. Sci.**, 92:2652-2656, 1995.

[gabora1993] Gabora L. *Meme and variations: a computational model of cultural evolution.* **1993 Lectures in Complex Systems**, Addison-Wesley, 1995.

[gandi1995] Gandhi M., Murray P., Spinks, G., and Wallace G. *Mechanisms of electromechanical actuation in polypyrrole.* **Synthetic Metals**, 73: 247-256, 1995.

[gage1995] Gage D. *Many-robot MCM search systems.* **Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium**, 1995.

[goldberg1997] Goldberg D. ad Mataric M. *Interference as a tool for designing and evaluating multi-robot controllers.* **AAAI-97**, Providence, RI, pp. 637-642, July 27-31, 1997.

[gordon1999] Gordon D., Spears W., Sokolsky O., and Lee I.(1999). *Distributed Spatial Control, Global Monitoring and Steering of Mobile Physical Agents.* **Proceedings of IEEE International Conference on Information, Intelligence, and Systems**. November, 1999.

[gordond1999] Gordon D. *Interaction patterns and task allocation in ant colonies.* **Information Processing in Social Insects**. C. Detrain, J. Pasteels, J. Deneubourg, eds. Birkhauser Verlag, 51-67, 1999.

[goss1990] Goss S., Beckers R., Deneubourg J., Aron S., and Pasteels J. *How trail laying and trail following can solve foraging problems for ant colonies.* **Behavioral Mechanisms of Food Selection**, Hughes R., ed. Berlin: Springer-Verlag, 1990.

[grand1997] Grand S/, Cliff D., and Malhotra A. *Creatures: artificial life autonomous software agents for home entertainment.* **Autonomous Agents 1997**, Marina Del Rey, CA, 1997.

[grasso1996] Grasso, F., Consi, T., Mountain, D. Atema, J. *Locating Odor Sources in Turbulence with a Lobster Inspired Robot*, **Proc. 4th International Conference on Simulation of Adaptive Behavior,** Maes, Mataric, Meyer, Pollack, and Wilson (Eds), Cambridge, Massachusetts : MIT Press, 1996.

[grasso1996a] Grasso, F., Consi, T., Mountain, D. Atema, J. *Behavior of Purely Chemotactic Robot Lobster Reveals Different Odor Dispersal Patterns in the Jet Region and the Patch Field of a Turbulent Plume*, Biol. Bull.**,** 191: 312-313, 1996.

[grasso1998] Grasso, F., Consi, T., Mountain, D., Atema, J. *Biomimetic Robot Lobster Performs Chemo-orientation in Turbulence Using a Pair of Spatially Separated Sensors: Progress and Challenges.* Submitted to **Journal of Robotics and Autonomous Systems**, 1998.

[grasso1998a] Grasso F., Basil J., and Atema J. *Toward the convergence: robot and lobster perspectives of tracking odors to their source in the turbulent marine environment.* **Proceedings of the 1999 IEEE ISIC/CIRA/ISAS Joint Conference**, 1998.

[harvey1997] I. Harvey. *Artificial Evolution for Real Problems.* **Fifth International Symposium on Evolutionary Robotics**, Tokyo, April 1997.

[harvey1997a] Harvey I., Husbands P., Cliff D., Thompson A., and Jakobi N. *Evolutionary Robotics: the Sussex Approach.* **Robotics and Autonomous Systems**, 20: 205-224, 1997.

[harvey1997b] Harvey I. *Cognition is not computation: evolution is not optimization.* **Proceedings of the International Conference on Artificial Neural Networks**. Gerstner W., Germond A., Hasler M., and Nicoud J. (eds), Springer-Verlag, 1997.

[harvey1994] Harvey I. *Evolutionary Robotics and SAGA: The case for hill crawling and tournament selection.* **Proceedings of Artificial Life 3**, Langton C., ed. Reading, Massachusetts: Addison Wesley , 299-326, 1994.

[hayes2000] Hayes A., Martinoli A., and Goodman R. *Comparing distributed exploration strategies with simulated and real autonomous robots.* Submitted.

[helbing1999] Helbing D. and Schreckenberg M. *Cellular automata simulating experimental properties of traffic flows.* **Physical Review E**, 59: R2505-2508, 1999.

[holland1999] Holland O. and Melhuish C. *Stigmergy, Self-Organization, and Sorting in Collective Robotics.* **Artificial Life**, 5: 173-202, 1999.

[holland1996] Holland, O. *The use of social insect based control methods with multiple robot systems.* **UKACC International Conference on Control '96**. 1996.

[hol1997a] Holland O. and Melhuish C. 1997. *An interactive method for controlling group size in multiple mobile robot systems.* **International Conference on Advanced Robotics**, Monterey, CA; July 7-9, Hyatt Regency Hotel.

[holland1998] Holland, O. Private communication.

[holland1996] Holland O. and Melhuish C. *Getting the most from the least: lessons for the nanoscale from minimal mobile agents.* **Artificial Life V**, Nara, Japan, 1996.

[holldobler1990] Holldobler B. and Wilson E. **The Ants**. Cambridge, Masaschusetts : The Belknap Press of Harvard University Press, 1990.

[holldobler1994] Holldobler B. and Wilson E. **Journey to the Ants**. Cambridge, Masaschusetts : The Belknap Press of Harvard University Press, 1994.

[huang1993] Huang Q. and Beni G. *Stationary waves in 2-dimensional cyclic swarms.* **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems**, Yokohama, Japan, July 26-30, 1993.

[husbands1997] Husbands P., Harvey I., Cliff D., and Miller G. *Artificial evolution: a new path for artificial intelligence?* **Brain and Cognition**, 34: 130-159, 1997.

[ishida1996] Ishida, H. et al. *Odour Source Localization System Mimicking Behaviour of Silkworm Moth.* **Sensors and Actuators**, 99: 225-230, 1996.

[ishida1994] Ishida H., Suetsugu K., Nakamoto T., and Moriizumi T. *Study of autonomous mobile sensing system for localization of odor source using gas sensors and anemometric sensors.* **Sensors and Actuators A**, 45:153-157, 1994.

[johnson1997] Johnson S., Sutter J., Engelhardt H., Jurs P., White J., Kauer J., Dickinson T., and Walt D. *Identification of multiple analytes using an optical sensor array and pattern recognition neural networks.* **Analytical Chemistry**, 69:4641-4648, 1997.

[karsai1993] Karsai I. and Penzes Z. *Comb building in social wasps: self-organization and stigmergic script.* **Journal of Theoretical Biology**, 161, 505-525, 1993.

[kazadi2000] Kazadi S., Goodman R., Tsikata D., Green D., and Lin H. *An autonomous water vapor plume tracking robot using passive resistive polymer sensors.* **Autonomous Robots**, to appear 2000.

[kazadi1998] Kazadi S. Unpublished design of plume tracking robot.

[kazadi1997] Kazadi S. Unpublished design of puck collecting robot.

[kennedy] Kennedy J. and Eberhart R. *Particle swarm optimization.* **Proceedings IEEE International Conference on Neural Networks.**

[kellly1996] Kelly I. and Keating D. 1996. *Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots.* **Proceedings of Mechatronics 1996.**

[kelly1998] Kelly I. and Keating D. 1998. *Faster learning of control parameters through sharing experiences of autonomous mobile robots.* **International Journal of Systems Science.** 29(7), 783-793.

[kelly1998a] Kelly I. and Keating D. *Increased learning rates through the sharing of experiences of multiple autonomous mobile robot agents.* **Proceedings of WCCI**, 1998.

[kotay1998] Kotay K., Rus D., Marsette V., and McGray C. *The self-reconfiguring robotic molecule: design and control algorithms.* **Algorithmic Foundations of Robotics**, Agrawal P., Kavraki L., Mason M., Peters A.K. (eds), 1998

[kube1993] C. Kube and H. Zhang. *Collective robotcs: from social insects to robots.* **Adaptive Behavior**, 2(2): 189-219, 1993.

[kuwana1995] Kuwana Y., Shimoyama I., and Miura H. *Steering Control of a Mobile Robot Using Insect Antennae.* **IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 2.** 530-535, 1995.

[kuwana1996] Kuwana, Y., Shimoyama, I., Sayama, Y., and Miura, H. *Synthesis of Pheromone-Oriented Emergent Behavior of a Silkworm Moth.* ***Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems**, 1722-1729, 1996.*

[lee2000] Lee D., Kazadi S., and Goodman R. *Swarm engineering for the traveling salesman problem.* To appear in **Ants'2000 workshop.**

[lewis1996] Lewis, N. et al. *Array-Based Vapor Sensing Using Chemically Sensitive, Carbon Black-Polymer Resistors.* **Chem. Mater.** 8:(9) 2298-2312, 1996.

[lewis1996a] ewis N. *The Caltech electronic nose project.* **Engineering and Science**, 60(3): 3-13, 1996.

[lonergan1996] Lonergan M., Severin E., Doleman B., Beaber S., Grubbs R., and Lewis N. *Array-based vapor sensing using chemically sensitive carbon black polymer resistors.* **Chem. Mater.**, 8: 2298-2312, 1996.

[maes1994] Maes P. *Modeling Adaptive Autonomous Agents.* **Artificial Life**, 1(1-2): 135-164, 1994.

[mafra-neto1994] Mafra-Neto A. and Carde R. *Fine-scale structure of pheromone plumes modulates upwind orientation of flying moths.* **Nature**, 369, 142-144, 1994.

[mankin1995] Mankin R. and Hagstrum D. *Three dimensional orientation of male* cadra cautella *(lepidoptera: pyralidae) flying to calling females in a windless environment.* **Environ. Entomol**, 24(6): 1616-1626, 1995.

[maris1996] Maris M. and Boekhorst R. *Exploiting physical contraints: heap formation through behavioral error in a group of robots.* **IROS '96 IEEE/RSJ International Conference on Intelligent Robots and Systems**, 1996.

[martinoli1999] Martinoli A., Ijspeert A., and Mondata F. *Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots.* **Robotics and Autonomous Systems**, 29: 51-63, 1999.

[martinoli1999a] Martinoli A., Ijspeert A., and Gambardella L. *A probabilistic Model for understanding and comparing collective aggregation mechanisms.* Floreano D., Nicoud J., and Mondada F., eds. **Advances in Artificial Life. Fifth European Conference on Artificial Life**, Lausanne, Switzerland, September 13-17, 1999.

[martinoli1999a] Martinoli A. **Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Control Strategies**, PHd Thesis, Ecole Polytechnique Federale de Lausanne, 1999.

[mataric1997] Mataric M. *Reinforcement learning in the multi-robot domain.* **Autonomous Robots**, 4 (1): 73-83, 1997.

[mataric1997a] Mataric M. *Behavior-based control: examples from navigation, learning, and group behavior.* **Journal of Experimental and Theoretical Artificial Intelligence**, 9 (2-3): 323-336, 1997.

[mataric1996] Mataric M. and Cliff D. *Challenges in evolving controllers for physical robots.* **Robotics and Autonomous Systems**, 19 (1): 67-83, 1996.

[mataric1995] Mataric M. *Issues and approaches in the design of collective autonomous agents.* **Robotics and Autonomous Systems**, 16 (2-4): 321-331, 1995.

[mataric1995a] Mataric M. *Designing and understanding adaptive group behavior.* **Adaptive Behavior**, 4 (1): 51-80, 1995.

[mataric1994] Mataric M. *Learning to behave socially.* **SAB 94**, Cliff D., Husbands P., Meyer J-A., and Wilson S. (eds), Cambridge: MA, MIT Press, 453-462, 1994.

[mataric1993] Mataric M. *Kin recognition, similarity, and group behavior.* **Proceedings of the Fifteenth Annual Cognitive Science Society Conference**, Boulder Colorado, USA, pp. 705-710, 1993.

[mataric1992] Mataric M. *Integration of representation into goal-driven behavior-based robots.* **IEEE Transactions on Robotics and Automation**, 8(3): 304-312, 1992.

[mataric1992a] Mataric M. *Designing emergent behaviors: from local interactions to collective intelligence.* In **Proceedings, From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92)**. J-A. Meyer, H. Roitblat and S. Wilson, eds., MIT Press, 432-441, 1992.

[mataric1991] Mataric M. *Behavioral synergy without explicit integration.* **SIGART Bulletin**, 2(4): 130-133, 1991.

[melhuish1996] Melhuish, C. and Holland O. *Getting the most from the least: Lessons for the nanoscale from minimal mobile agents.* **Proceedings of Artificial Life V**. C. Langton, K. Shimorhara, eds. Mit Press: Cambridge, MA, 1996.

[melhuish1999] Melhuish C. **Strategies for Collective Minimalist Mobile Robots**, PHd Thesis, University of the West of England, 1999.

[murlis1992] Murlis J., Elkinton J., and Carde. *Odor plumes and how insects use them.* **Ann. Rev. Entomol.** 37:505-532, 1992.

[nolfi1994] olfi S., Floreano D., Miglino O., and Mondada F. *How to evolve autonomous robots: different approaches in evolutionary robotics.* Brooks R. and Maes P. (Eds.), **Proceedings of the IV International Workshop on Artificial Life**, Cambridge, MA: MIT Press, 1994.

[pei1993] Pei Q. and Inganas O. *Electroelastomers: conjugated poly(3-octylthiophene) gels with controlled crosslinking.* **Synthetic Metals**, 55-57: 3724-3729, 1993.

[pei1993a] Pei Q. and Inganas O. *Electrochemical muscles: bending strips built from conjugated polymers.* **Synthetic Metals**, 55-57: 3718-3723, 1993.

[pei1992] Pei Q. and Inganas O. *Conjugated polymers and the bending cantilever method: electrical muscles and smart devices.* **Advanced Materials**, 4(4): 277-278, 1992.

[prem1995] Prem E. *Grounding and the Entailment Structure in Robots and Artificial Life.* F. Moran et al. (eds.) **Advances in Artificial Life, Proc. of the Third European Conf. on Artificial Life**, Granada: Springer, 1995.

[radcliffe1994] Radcliffe N. and Surry P. *Formal Memetic Algorithms.* **Evolutionary Computing: AISB Workshop**, T. Fogarty (ed.), Menlo-Park, CA: Springer-Verlag, 1994.

[renou1994] Renou M. and Lucas P. *Sex pheromone reception in* Mamestra brassicae *L. (lepidoptera): responses of olfactory receptor neurons to minor components of the pheromone blend.* **J. Insect Physiology**, 40(1): 75-85, 1994.

[ressler1994] Ressler K., Sullivan S., and Buck L. *A molecular dissection of spatial patterning in the olfactory system.* **Current Opinion in Neurobiology**, 4:588-596, 1994.

[russell1995] Russell R. A. *Laying and sensing odor markings as a strategy for assisting mobile robot navigation tasks.* **IEEE Robotics and Automation Magazine**, 3-9, September 1995.

[russell1994] Russell R. A., Thiel D., Deveza R., and Mackay-Sim A. *A Robotic System to Locate Hazardous Chemical Leaks.* **IEEE International Conference on Robotics and Automation**. 13 (3): 232-239, 1994.

[schier1998] Scheier C. and Pfeifer R. (1998). *Exploiting embodiment for category learning.* R. Pfeifer et al., (eds.) **From animals to animats. Proc. of the 5th International Joint Conference on Simulation of Behavior, SAB 98**. Cambridge, Massachusetts: MIT Press, 32-37, 1998.

[schneider-fontan1998] Schneider-Fontan M. and Mataric M. *The role of critical mass in multi-robot adaptive task division.* **IEEE Transactions on Robotics and Automation**, 14(5), 1998.

[schoonderwoerd1997] Schoonderwoerd R. and Holland E. *Ant-based load balancing in telecommunications networks.* **Adaptive Behavior**, 5(2): 169-207, 1997.

[shannon1953] Shannon A. *Communication theory exposition of fundamentals.* **Institute of Radio Engineers, Transactions on Information Theory**, 1: 44-47, 1953.

[smela1993] Smela E., Inganas O., Pei Q., and Lundstrom I. *Electrochemical muscles: micromachining fingers and corkscrews.* **Advanced Materials**, 5(9), 630-632, 1993.

[smela1995] Smela E., Inganas O., and Lundstrom I. *Controlled folding of micrometer-size structures.* **Science**, 268: 1735-1738, 1995.

[smithers1994] Smithers T. *On why better robots make it harder.* **From animals to animats: Proceedings of the Third International Conference on Simulation of Adaptive Behavior**, Cambridge, MA: MIT Press, 1994.

[smither1997] Smithers T. *Autonomy in Robots and Other Agents.* **Brain and Cognition**, 34: 88-106, 1997.

[spears99] Spears W., and Gordon D. (1999). *Using Artificial Physics to Control Agents.* **Proceedings of IEEE International Conference on Information, Intelligence**, and Systems. November, 1999.

[sutton1990] Sutton, R. *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming.* **Proceedings of the Seventh International Conference on Machine Learning**, 1990.

[theraulaz1998] Theraulaz G., Bonabeau E., and Deneuborg J. *Response threshold reinforcement and division of labour in insect societies.* **Proceedings of the Royal Society of London, Series B**. 265: 327-332, 1998.

[thompson1996] Thompson A. *Evolutionary Techniques for Fault Tolerance.* **Proceedings of UKACC Internation Conference on Control**, 693-698, 1998.

[treiber1999] Treiber M. and Helbing D. *Explanation of observed features of self-organization in traffic flow.* **Physics Review Letters**. submitted, 1999.

[treiber1999a] Treiber M. and Helbing D. *Macroscopic simulation of widely scattered synchronized traffic states.* **J. Phys. A: Math. Gen.**, 32: L17-L23, 1999.

[ulrich1997] Ulrich I., Mondada F., and Nicoud, J. *Autonomous vacuum cleaner.* **Robotics and Autonomous Systems**, 9:233-245, 1997.

[unemi1996] Unemi T. and Nagayoshi M. *Evolution of Reinforcement Learning Agents - toward a feasible design of evolvable robot team.* **Workshop Notes of ICMAS'96 Workshop 1: Learning, Interactions and Organizations in Multiagent Environment**, 1996.

[wang1996] Wang Z., Nakano E., Matsukawa T., and Hanada K. *Cooperative object manipulation by multiple mobile robots: strategy and experimental implementation.* **Fourth International Conference on Control, Automation, Robotics, and Vision**, Singapore, 3-6 December 1996.

[werger1999] Werger B. and Mataric M. *Exploiting Embodiment in Multi-Robot Teams.* Submitted to AMAI special issue on mathematical aspects of ant-robotics, 1999.

[white1997] White, J. and Kauer J. *Generating Sensor Diversity through Combinatorial Polymer Synthesis.* **Analytical Chemistry**, 69 (17): 3413-3418, 1997.

[white1996] White J., Kauer J., Dickinson T., Walt D. *Rapid analyte recognition in a device based on optical sensors and the olfactory system.* **Analyical Chemistry**, 68(13): 2191-2202, 1996.

[zanen1996] Zanen P. and Carde R. *Effects of host-odour plume altitude and changing wind velocity on upwind flight manoevres of a specialist braconid parasitoid.* **Physiological Entomology**, 21: 329-338, 1996.