ABSTRACT

Title of Thesis: SWARM INTELLIGENCE AND STIGMERGY: ROBOTIC IMPLEMENTATION OF FORAGING BEHAVIOR

Mark Russell Edelen, Master of Science, 2003

Thesis directed by: Professor Jaime F. Cárdenas-García
Professor Amr M. Baz
Department of Mechanical Engineering

Swarm intelligence in multi-robot systems has become an important area of research within collective robotics. Researchers have gained inspiration from biological systems and proposed a variety of industrial, commercial, and military robotics applications. In order to bridge the gap between theory and application, a strong focus is required on robotic implementation of swarm intelligence. To date, theoretical research and computer simulations in the field have dominated, with few successful demonstrations of swarm-intelligent robotic systems.

In this thesis, a study of intelligent foraging behavior via indirect communication between simple individual agents is presented. Models of foraging are reviewed and analyzed with respect to the system dynamics and dependence on important parameters. Computer simulations are also conducted to gain an understanding of foraging behavior in systems with large populations. Finally, a novel robotic implementation is presented. The experiment successfully demonstrates cooperative group foraging behavior without direct communication. Trail-laying and trail-following are employed to produce the required stigmergic cooperation. Real robots are shown to achieve increased task efficiency, as a group, resulting from indirect interactions. Experimental results also confirm that trail-based group foraging systems can adapt to dynamic environments.

SWARM INTELLIGENCE AND STIGMERGY:

ROBOTIC IMPLEMENTATION OF FORAGING BEHAVIOR

by

Mark Russell Edelen

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2003

Acknowledgements

First and foremost, I would like to thank my heavenly Father and the Lord Jesus Christ, for making this possible. Without God’s supernatural intervention, I would never have persevered. All credit and praise belong to my God.

I would like to thank Professors Amr Baz and Balakumar Balachandran for their assistance and support. A special thanks goes to Professor Jaime Cárdenas-García for all of his valuable advice and patience. I appreciate the freedom that he allowed me in pursuing a topic of my interest. That is a rare luxury, but he made it a reality.

Lastly, thank you to everyone who played a role in proofreading this text, donating supplies, or simply supporting me while I labored. I want to thank you all for your understanding and selfless desire to help.

Dedication

You are my God, and I will give you thanks;
You are my God, and I will exalt you.
- Psalms 118:28

Whatever you do, work at it with all your heart, as working for the Lord, not for men, since you know that you will receive an inheritance from the Lord as a reward. It is the Lord Christ you are serving.
- Colossians 3:23-24

Go to the ant, you sluggard!
Consider her ways and be wise.
- Proverbs 6:6

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

Children spend hours closely studying the wanderings of ants in their backyards. How do the ants know how to follow the path to a source of food? How do they know how to return to the nest? From inquisitive children to the researchers of academia, insect swarms tend to generate a unique sense of wonder. There is a certain mysterious synergism present in the swarms of ants, bees, and other social insects. An external human observer, simultaneously viewing individual simplicity and group complexity, fails to comprehend the nature of the cooperation. This is the essence and the appeal of swarm intelligence. It is also the difficulty in studying such systems.

This research is motivated by the segment of swarm intelligence research that joins two otherwise unrelated fields: entomology and robotics. Within the past fifteen years, a movement has begun to move beyond the theoretical understanding of biological swarms. Instead, engineering and artificial intelligence researchers are seeking to replicate the efficacy of biological swarms to solve real-world problems. While the applications of swarm intelligence have propagated beyond the fields of engineering and computer science into business, telecommunications, finance, social psychology, etc., robotics-based applications are the focus of this research. A number of researchers have undertaken the task of demonstrating complex, intelligent system behavior in groups of cooperative, mobile robots. This has been accomplished with some success, albeit limited to highly constrained experimental situations. The tendency of most researchers is to discuss the potential benefits of swarm-based robotics without developing an operational robotic swarm.

The primary goal of this thesis is to employ current theoretical knowledge to achieve a physical swarm-intelligent system composed of real robots. Implicit in this statement of purpose is the necessity of establishing metrics for swarm intelligence in a multi-agent system. In order to do so, the benchmark example of foraging will be examined in detail. The task of collective foraging is highly appropriate for such an investigation, due to direct biological inspirations and generally accepted metrics for task completion efficiency. In addition, since group foraging is often used to demonstrate swarm intelligence in multi-robot systems, this task provides a basis for comparison to past research efforts.

The primary goal of implementing a functional robotic swarm requires a detailed working knowledge of swarm-intelligent systems. First of all, a theoretical understanding of group foraging is required as a foundation of system design. Therefore, mathematical models of foraging are reviewed and analyzed in order to thoroughly understand foraging system dynamics. Computer simulations of swarms then function to bridge the gap between theory and experimentation. The simulations foster deeper understanding of system behavior dependence on several important variables. These simulations are particularly valuable for investigating system behavior as a function of swarm size.

Insights gained via theoretical modeling and computer simulations direct the experimental design of a robotic swarm. In an idealized situation, the robotic system should precisely duplicate the system behavior predicted theoretically. However, the constraints and uncertainties of real-world implementation make such an expectation unreasonable. Therefore, it is necessary to limit the scope of the experimental system and focus on several key elements of foraging behavior. In this case, the experimental multi-robot system was designed to demonstrate swarm-intelligent foraging behavior under specific design constraints in order to investigate a limited subset of system characteristics. The design constraints are as follows:

- no direct communication
- decentralized control
- behavior-based control

The key system attributes under investigation are the following:

- task efficiency
- system dynamics
- food source exploitation modes

These attributes depend on a number of system parameters; the following three are investigated here:

    - number of robots
    - pheromone decay rate
    - item distribution

Having provided the motivations and goals of the research, an overview is presented to provide the reader with a broad perspective of each chapter in relation to the complete work. In the following two chapters (2 and 3), the literature relevant to the field of swarm intelligence is reviewed. Chapter 2 provides a thorough review of collective robotics research, from the historical foundations of the field to the current state of the art. Special attention is given to successful demonstrations of swarm intelligence through physical robotic implementation. Chapter 3 covers, in a similar manner, the biological inspirations of this research. The foraging behavior of insect societies is discussed. In Chapter 4, mathematical models of foraging are reviewed and employed to predict and explain specific characteristics of swarm-based systems. Computer simulations in Chapter 5 demonstrate many of the theoretical propositions presented in Chapter 4. The software simulations also allow for the study of large-scale swarms that involve as many as 100 individual agents. The chief aims of this research are addressed in Chapters 6 and 7, which focus on foraging experiments with physical robotic agents and a synthesis of the results from simulation and experimentation. In Chapter 6, the details of robot design, experimental setup, and results are provided. In Chapter 7, the results of simulations and experimentation are analyzed and the results from these two types of investigation are compared and related to the predictions of theoretical models. Chapter 9 contains general conclusions, lessons, and directions for future research.

Chapter 2: Background – Swarm Robotics

Swarm intelligence claims historical underpinnings from both biology and engineering. Reviewing the genesis and subsequent development of swarm intelligence research therefore requires approaching the subject from both entomology and robotics. First, the subject will be discussed from the perspective of historical robotics research.

Section 2.1: Cooperative Mobile Robotics

Swarm-based robotics falls within the broader classification of cooperative, autonomous, mobile robotics. This popular area of robotics research focuses on the study of multi-robot systems designed to cooperatively achieve a task. Multi-robot cooperation has become increasingly important for a number of reasons. Among these are the following [1]:

- Tasks are often inherently too complex for a single robot to accomplish.
- Several simple robots can be a cheaper and easier solution than one powerful robot.
- Multi-robot systems are generally more flexible and fault-tolerant than single robots acting alone.

The broad field of cooperative mobile robotics has flourished for decades, while swarm-based robotic systems have emerged fairly recently. The historical origins of swarm robotics are presented in the next section. The remainder of the chapter examines key aspects of a swarm-based robotic system--control architecture, communication, physical morphology—and concludes with an explanation of important robotic tasks related to swarm intelligence.

Section 2.2: Origins of Swarm-Based Robotics

In order to fully trace the intellectual heritage of swarm-based robotics, one must begin in the early 1970s. At that time, coordination and interaction of multiple agents were being studied in the field of distributed artificial intelligence (DAI) [1], but investigations were limited to problems involving software agents. This tendency persisted until the late 1980s, when the robotics research community began to explore cooperative robotic systems [2]. The earliest forays into cooperative robotics related to cellular (or reconfigurable) robotic systems, cyclic swarms, multi-robot motion planning, and primitive architectures for multi-robot cooperation. The first two of these topics eventually merged into the current field of swarm intelligence. Cellular robotic systems, such as CEBOT (CEllular roBOTs), were initially explored by Fukuda et al. [3]. CEBOT drew direct inspiration from biological organisms to generate an architecture based on decentralized hierarchies of robotic cells [1]. Gerardo Beni started working on the same topic from a more theoretical perspective in 1988 [4]. Within a year, Beni had coined and popularized the term “swarm intelligence” in relation to robotics [5]. He later identified a vague definition of the term swarm intelligence: “a property of systems of non-intelligent robots exhibiting collectively intelligent behavior” [6]. The most precise definition provided by Beni was not a written verbal definition, but a mathematical construct is presented graphically in Figure 1. Note that the generalized definition (on the right) is less strict than the original, as it allows for systems to be termed swarm-intelligent by completing effective work, W, more efficiently than a non-cooperative group. Beni’s definition requires thatthe useful work in a given task can *only* be performed by N interacting agents, and only for N greater than a critical number, N_c.

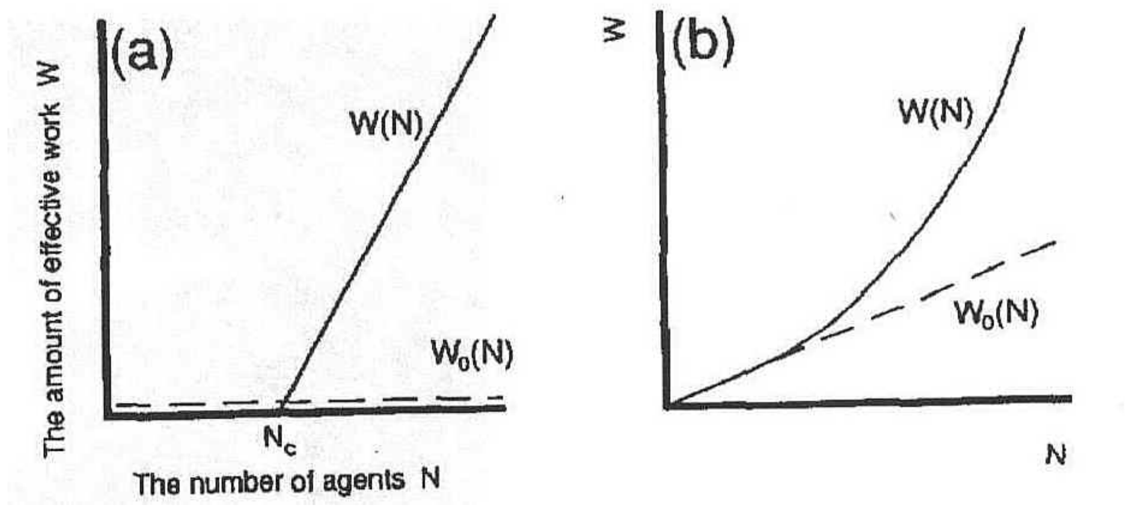


Figure 1. (a) Beni's definition of swarm intelligence (b) the generalized definition. W(N) denotes the amount of work achieved by N interacting agents, and $W_0(N)$ denotes the work achieved by N independent agents (taken from [7], p. 344)

Deneubourg, Theraulaz, and Beckers, who study swarm intelligence from an ethological perspective, define a swarm as "…a set of (mobile) agents which are liable to communicate directly or indirectly (by acting on their local environment) with each other, and which collectively carry out a distributed problem solving" [8], p. 123. These definitions conveniently avoid the problem of defining intelligence, although Deneubourg et al. imply that intelligence is somehow related to problem solving. The reader is referred to [9] for a thorough discussion of robotic intelligence, the definition of which is debated philosophically as much as technically.

Around the same time that cellular robotics research was taking shape, Rodney Brooks, from MIT's Artificial Intelligence Laboratory, published a groundbreaking paper in the still undeveloped field of cooperative robotics [10]. In what has clearly become the most widely cited source in the swarm intelligence community, Brooks outlines a “robust layered control system for a mobile robot” known as *subsumption architecture.* Brooks’ paper initiated the rapid progression of the behavior-based control movement [11]. Combined with the inertia of cellular robotics and identified by Beni’s terminology, swarm intelligence emerged in the early 1990s as a major component of mobile robotics research. Brooks and Beni provided the impetus for the emergence and rapid expansion of swarm intelligence as a research field. This is evidenced by the nearly omnipresent references to behavior-based control in the robotics literature [11]. However, despite the rapid advance of swarm intelligence, a number of robot control architectures were developing simultaneously. These control strategies are described in the following section.

Section 2.3: Swarm Control Architectures

Regardless of the intended application, each robot in a multi-robot system requires some preprogrammed processes by which it can assimilate sensory input and respond with appropriate actuation. A small group of popular design architectures have emerged that control the “brains” of these robots. The techniques of each architecture are applicable to a range of programming and hardware domains, but the central design framework remains consistent. Currently, these architectures exist amidst the opposing viewpoints of two philosophical approaches to robotic intelligence. The classical AI approach, which dominated the mobile robotics community for years, gives rise to architectures focusing on intensive central processing and high-level reasoning. On the other hand, swarm intelligence adherents advocate a far simpler approach based on the nature of interactions between individual robots. This behavior-based approach is presented first, followed by an assortment of specific architectures spanning the spectrum from classical AI to swarm intelligence.

Subsection 2.3.1: Behavior-based Control

As mentioned previously in Section 2.2, behavior-based approaches to controlling agents in a multi-robot system have had a great impact on the course of cooperative robotics. Many of the current leaders in robotic swarm intelligence research cite Brooks' landmark subsumption architecture as an important influence (e.g., [1, 12-17]).

Brooks' 1985 paper presented a novel control strategy that is both flexible and robust. It differed fundamentally from any of the prevailing control schemes of the day by decomposing the control system based on task achieving behavior. This stood in stark contrast to the traditional functional decomposition of contemporary strategies. In developing this new decomposition, Brooks considered certain requirements for controlling intelligent autonomous mobile robots. He postulated that an effective control system must meet four basic requirements:

1. Multiple Goals
2. Multiple Sensors
3. Robustness
4. Additivity

In addition to these four requirements, Brooks made certain largely philosophical assumptions. Three of these assumptions have become central tenets in swarm-based robotics, and are reproduced below (quoted from [10], pp. 3-4) :

- *"Complex (and useful) behavior need not necessarily be a product of an extremely complex control system. Rather, complex behavior may simply be the reflection of a complex environment. It may be an observer who ascribes complexity to an organism—not necessarily its designer."*
- *"Things should be simple….When building a system of many parts one must pay attention to the interfaces. If you notice that a particular interface is starting to rival in complexity the components it connects, then either the interface needs to be rethought or the decomposition of the system needs redoing."*
- *"The worlds where mobile robots will do useful work are not constructed of exact simple polyhedra. While polyhedra may be useful models of a realistic world, it is a mistake to build a special world such that the models can be exact. For this reason we will build no artificial environment for our robot."*

Brooks proceeds from these points to explain a system of layered *levels of competence* for a robot aiming to explore the MIT robotics laboratory. Each level above the default is constructed in sequence to form a cascade of competence levels. Each higher level subsumes, or includes, the roles of lower-level layers when they request control of actuators. Conversely, the lower levels are in subsumption to the higher behavioral levels; thus, this arrangement is termed subsumption architecture. Partitioning at any level in the structure leaves behind a complete operational control system.

In order to implement such a control system, Brooks suggests that one should first build and fully debug the simplest level of competence. When this is achieved, the next layer is built on top of the "zeroth" layer, and the process is repeated (see Figure 2). The method of architectural development has come be known as the “constructivist approach,” which is an important accompaniment to behavior-based control. Higher levels are permitted to sample the input of and suppress the output of lower levels.

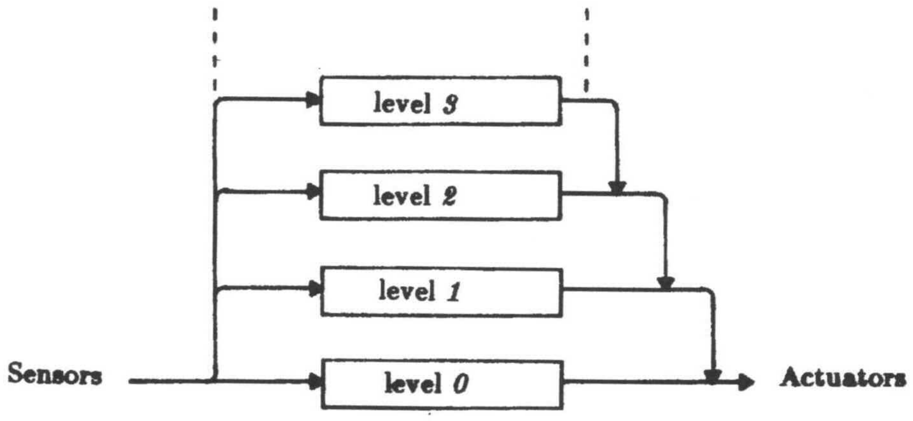


Figure 2. Layered competence levels (taken from [10], p. 7)

In 1991, Brooks published two companion papers to further develop his novel conception of artificial intelligence in distributed robotics. “Intelligence without Representation” [18] and “Intelligence without Reason” [9] combine to argue a strong case against traditional AI models. Brooks’ essential argument is that tradition artificial intelligence has an unrealistic goal of replicating “human level intelligence in a machine” [9], p. 5. The complexity of such an undertaking, he posits, is beyond the grasp of research efforts in the foreseeable future. Rather than focusing on a fantastic and distant dream of Turing equivalence, Brooks proposes that researchers first focus on highly simplified intelligent systems. Once simple and complete intelligent systems are created, incremental improvements automatically ensure the validity of subsystems and interfaces. Traditional AI robots generate abstract representations of the world around them in order to plan future actions. In the opinion of Brooks and other advocates of behavior-based robotics, representations of the world are an unnecessary abstraction in many cases. Rather, they contend that it is better to "use the world as its own model."

Finally, in [9], Brooks cements his case with an extremely thorough defense of behavior-based robotic control. He argues quite convincingly that traditional AI ignores the intelligence evident in the natural world, while behavior-based approaches more closely resemble efficient biological systems. Four canonical statements of behavior-based control are provided from [9] to conclude and summarize his important work:

1. *The world is its own best model.*
2. *The world grounds regress.*
3. *Intelligence is determined by the dynamics of interaction with the world.*
4. *Intelligence is in the eye of the observer.*

Subsection 2.3.2: Other Control Architectures

The behavior-based, or subsumption, architecture described above is more general than some of the well-defined architectures employed in multi-robot systems. The work of Brooks and Mataric has developed into an overriding control philosophy more than a specific architecture. Many of these other architectures were influenced by both traditional AI and behavior-based approaches to robotic problem solving. As a rule, the following architectures are quite explicit and precise, making them less important in the overarching philosophical control debate. More than anything, the following architectures are reflections of broader philosophical approaches to multi-robot systems research.

- CEBOT (Cellular Robotics System) is a decentralized, hierarchical control system inspired by cellular organization in biology. A dynamically configurable system is created using autonomous “cells” that are capable of reconfiguration in response to changing environments [2].
- SWARM, as the name suggests, is the closest architecture to a strictly behavior-based approach. It is a distributed system of many (usually) homogeneous robots (refer to [1] for a more detailed description).
- ALLIANCE/L-ALLIANCE were developed to study cooperation in a heterogeneous, small group of loosely coupled robots [1]. Each robot senses the effects of their own actions and those of other agents through perception and explicit communication. L-ALLIANCE incorporates reinforcement-learning into the architecture.
- ACTRESS consists of a group of “robotors,” including 3 robots and 3 workstations working together to perform specified tasks [2]. Issues such as communication efficiency between robots and environment managers are studied.
- GOFER has been used to study distributed problem solving by a group of mobile robots using traditional AI techniques. A central task planning and scheduling system communicates with all robots based on a global view of the environment [1].
- Neural networks are similar to behavior-based architectures, with some important distinctions. Both strategies employ mainly reactive control schemes, but in neural networks the response is based on a weighted

combination of all input signals [19]. Neural networks are also always state-free.

Section 2.4: Communication

Cao, Fukunaga, and Kahng [1], Arai, Pagello, and Parker et al [20], and Dudek et al [21], in their seminal reviews of advances in cooperative mobile robotics, all focus on the importance of communication in swarm-based robotic systems. The very essence of cooperative robotics requires some type of communication, however indirect, in order to produce meaningful, beneficial interactions. Of the five taxonomic axes for swarm robots proposed in [21], three are based on inter-robot communication: communication range, topology, and bandwidth. Clearly, the nature and extent of communication within a robotic swarm is vital to successful demonstrations of swarm intelligence. Arai, Pagello, and Parker identify the important distinction between implicit and explicit communication, stating that “implicit communication occurs as a side effect of other actions, or ‘through the world’, whereas explicit communication is a specific act designed solely to convey information to other robots on the team” [2], p. 656. Further discussion of communication divides the continuum into these same two halves: explicit and implicit communication.

Subsection 2.4.1: Explicit Communication

Cao, Fukunaga, and Kahng provide a through review of explicit communication techniques used in cooperative robotics [1]. Explicit communication consists of sending and receiving of messages that are intended to convey information. In some cases, each message is intended for an individual robot. Each robot may have some kind of identification signal, for example, allowing other robots to receive its position, heading, velocity, etc. Messages can be sent from one individual to many recipients, as in "sign-board" type communication. In [22], explicit communication takes the form of a "hello-call" protocol. This signaling protocol allows the robots to form communication chains, effectively extending the communication range of a single robot. Explicit communications become increasingly complex as the number of robots increases. For very large groups of robots, these schemes may become impractical.

Subsection 2.4.2: Implicit Communication

Implicit communication, also known as stigmergy, occurs through the environment. This type of communication scales well for potential future applications requiring hundreds or thousands of robots [23]. It is most often implemented in cooperative foraging, clustering, or sorting tasks [12, 16]. Messages are "sent" only by altering some aspect of the environment which is then sensed by another individual robot. The environmental alteration can be intentional, as in the case of trail-laying [24], or unintentional, as in clustering or sorting tasks [25, 26]. Information gained through implicit communication tends to be extremely limited. Messages cannot be designated for particular individuals, as the environment is available to all agents. Information acquisition and processing is often much faster than the case of explicit communication.

Section 2.5: Learning

Learning tends to be much more difficult to implement in cooperative mobile robotics as compared to traditional robots acting alone. Cooperative tasks, due to the nonlinear interaction between individuals, make it necessary to use specialized learning algorithms. Despite the complexity of such algorithms, it is highly desirable to develop multi-robot systems capable of learning to cooperate. This could augment or replace the difficult task of "finding the correct values for control parameters that lead to a desired cooperative behavior" [1], p.10. Current efforts to develop learning in multi-robot systems are motivated by this reasoning.

Most of the learning algorithms used in cooperative robotics are based on reinforcement [1, 27]. This traditional formulation is posed in terms of states and actions. The robot executes an action from within a particular state, at which time it may change states. The action may or may not be reinforced, depending on the desired behavior. In time, the robot learns to correlate states and actions to produce meaningful behavior. This type of paradigm is used in the L-ALLIANCE control architecture to help robots learn to estimate the performance of neighboring robots. It has also been used to teach a group of robots a simple, artificial robot language [1]. Some researchers have even designed tasks with the sole purpose of "teaching" a group of robots how to cooperatively perform a separate task [28]. One example of this is the work of Gaussier and Zrehen, who employ a neural network technique designed to allow each individual in a group of Khepera robots to learn its body geometry through repetitive obstacle avoidance [14].

Mataric proposes a reformulation of the reinforcement paradigm that uses basis behaviors, rather than actions, as the focus of reinforcement [27]. In this manner, the learning search space is reduced considerably. A space of a few basis behaviors replaces more common spaces involving dozens of actions. The result is a more rapid learning process. While work continues in the area of learning, it remains one of the least developed aspects of cooperative robotics research.

Section 2.6: Morphologies

The physical morphology of each individual in a swarm-based robotic system is an important consideration in any swarm design. Mechanical design is also highly interdependent with the control architecture, making it a critical element of robotic design. In most cases of swarm intelligence demonstrations with real robots, the swarm is at least physically homogeneous (if not behaviorally as well). There are rare cases where this is not the case, but this discussion focuses on morphologies common for individuals within a homogeneous swarm. Methods of locomotion and environmental manipulation will be discussed, followed by a review of commonly used commercial platforms of autonomous mobile robots.

Subsection 2.6.1: Locomotion

Two main classes of locomotion are present for autonomous mobile robots: wheeled or legged. While both approaches are valuable for specific purposes, swarm-based systems nearly always employ wheeled locomotion. Walking, legged robots involve a higher degree of mechanical complexity [19], especially in loosely constrained or dynamic environments common to swarm applications. Legged robots also tend to be more expensive and incapable of high maneuverability or rapid movement. For these reasons, legged robots are generally unsuitable for swarm investigations and will not be discussed further. Rather, the discussion will focus on the broad class of wheeled locomotion.

[29],[30], and [31] provide excellent summaries of the most common types of wheeled locomotion in autonomous robots, which are described briefly below:

1. Differential Drive – Composed of two parallel, independently-powered drive wheels on opposite sides of the robot and one or more casters for stability, this drive is very simple and quite common. Note that *differential* refers to the fact that the velocity vector of the center of mass is the resultant of two independent components. A differential gear system is not involved.
2. Dual Differential Drive – Designed to mechanically solve the problem of following a straight path, this drive includes two differentials in a more complex gear system than the simple differential drive. Two motors are used: one for driving both wheels the same direction to go straight, and one to turn by driving them in opposite directions. This drive configuration is simple enough to be built easily with LEGO components (see Figure 3).

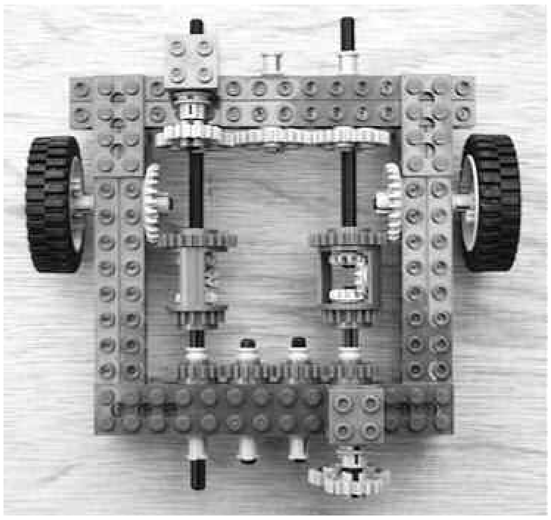


Figure 3. LEGO Dual Differential Drive [30]

3. Skid-Steer Drive – This variation of the differential drive is often used with tracked vehicles, but sometimes with four or six wheel platforms as well. As the name implies, this configuration relies on tire or track skidding in order to turn.
4. Steering Drive – This is the standard locomotion setup in modern cars and most other vehicles, but is somewhat rare in the robotics community. It features one or two front steering wheels and two fixed rear wheels. As in cars, the front, rear, or all four wheels can be powered. Steering drives suffer from large turning radii and the inability to rotate without translation.
5. Tricycle Drive – In what is actually a subset of the steering drives, the tricycle drive uses a single, powered front wheel and two passive rear wheels to achieve certain mobility advantages. The powered front wheel maintains driving force at any turning angle.
6. Synchro Drive – The synchro drive uses three or more wheels, all of which are driven and steered. All of the wheels turn and rotate in sync, remaining

parallel at all times. Through complex gearing, a synchro drive employs only two motors to produce changes in direction without altering robot orientation.

7. Articulated Drive – Commonly seen in wheeled excavators, articulated drives consist of front and rear chasses joined at a central pivot point. Steering is accomplished by rotating the front chassis with respect to the rear chassis about the pivot point. This is essentially a variation of the steering drive.
8. Pivot Drive – This drive system is composed of a four-wheeled chassis with non-pivoting wheels and a central rotating platform capable of vertical motion. While in motion, the pivot drive travels in a straight line. To turn, the robot stops, lowers the pivot platform until the wheels are suspended, rotates through the desired angle, and finally raises the platform.

These are the most fundamental types of wheeled locomotion for autonomous robots. The advantages and disadvantages of each are summarized below in Table 1.

Table 1. Robotic Locomotion Configurations

Drive Configuration	Advantage(s)	Disadvantage(s)	Example(s)
Differential	simplicity	control (straight line)	Pioneer3 [32]
Dual Differential	control (straight line)	efficiency (frictional losses)	LEGO robots [29]
Skid-Steer	simplicity	control odometry (skidding)	Pioneer3-AT [33]
Steering	simplicity	path planning	*various, see* [34]
Tricycle	maneuverability (compared to regular steering drive)	handling uneven terrain	Fiat [35]
Synchro	control	complexity	B21r[36]
Articulated	simplicity	path planning	M96 Rover [37]

Pivot	control	complexity	Ritorno [29]

A few other highly specialized types exist, including the “Killough Drive” or Omnidirectional Holonomic Platform (OHP) developed by Pin and Killough. This special case introduces the issue of holonomy. A non-mathematical definition of holonomy is given by Ferrari, Ferrari, and Hempel [29]: “the capability of a system to move toward any given direction while simultaneously rotating” p. 151. In practice, holonomic systems are more difficult to design in terms of mechanical complexity. However, they are desirable due to increased maneuverability [29] and simplified control theory. Yamaguchi, [38], discusses some of the theoretical difficulties in developing a distributed control law for multiple nonholonomic robots. In particular, he cites the lack of established methodologies for asymptotically/exponentially stabilizing nonholonomic systems. Despite the increased complexity in control theory, most researchers employ nonholonomic locomotion schemes (e.g., [23, 24, 26, 38-43]).

Subsection 2.6.2: Manipulation of the Environment

The requirements for environmental manipulation vary greatly between swarm-based applications. In some instances, the group of multiple robots has no need to alter the environment in any manner. However, in tasks such as cooperative foraging or sorting, it is of course necessary to manipulate objects in the environment. Indirect communication also takes place via changes to the environment, some of which are intentional communication markers. In most cases, physical manipulation of the environment is limited to simple end effectors such as grippers. These types of tools are easily augmented to a preexisting robot, which explains the popularity of building upon one of the available commercial platforms.

Subsection 2.6.3: Commercial Platforms for Multi-Robot Systems

One of advantages to behavior-based multi-robot systems is the inherent simplicity in individual behavior. In terms of sensing, processing, and manipulation abilities, individual robots can be extremely simple. As a result, the majority of swarm-based robotics experiments take place with “off-the-shelf” commercial robots. In many cases, very minor (if any) modifications are required before the robot(s) are properly equipped for swarm-compatible tasks. Communication protocols are sometimes added or modified, along with the addition of actuators for specific environmental manipulation. The onboard processing unit and locomotion design rarely require modification, except in highly unusual applications. A summary of the most widely-used commercially available robotics platforms is provided below in Table 2.

Table 2. Cmmercial Robot Platforms

	Processor	**RAM (K)**	**ROM (K)**	**Cost**	**Size**
Khepera II [35]	Motorola 6833	512 Kb	512 Kb	$2,300	70 mm D X 30 mm H
Moorebot [36]	Intel 386SX	4 Mb	80 Mb	?	25 cm D X 28 cm H
Pioneer3	Hitachi H8S	32 Kb	1 Mb	?	44 cm X 38 cm X 22 cm H
Handy Board (processor only)	Motorola 6811	32 Kb	NA	$300	varies
LEGO Mindstorms	Hitachi H8	32 Kb	16 Kb	$200	varies

Section 2.7: Cooperative Tasks in Mobile Robotics

Cooperative systems of autonomous, mobile robots exhibit an array of control schemes, communication modes, and physical morphologies. The choice of each of these elements is generally dictated by the desired task. Most multi-robot systems are designed with a global task or goal. Cooperating to achieve this goal is what drives the system architecture, morphology, etc. While a great many tasks have been proposed and studied, swarm-based multi-robot systems generally focus on a handful of major task classifications. These task categories are presented below.

Subsection 2.7.1: Formation Control

As is the case for each of the major tasks in swarm-based robotics, dynamic formation control relates closely to a biological parallel. Formation control, however, relates on the broadest scale to the natural realm. Many of the behaviors used to model cooperative robotic formations arise from social insects such as bees, ants, wasps, and termites. However, flocks of birds, schools of fish, and herds of land animals also exhibit elements of swarm intelligence as they establish and maintain complex formations while moving as a social unit. Such a richness of inspiration from ethology has resulted in formation generation and control being one of the earliest and most rapidly developed areas of swarm-based robotics research [2, 20]. While the early efforts at formation generation and maintenance focused on very simple behaviors, recent work has moved into the realm of sophisticated self-organized flocking [44], pursuit problems[38, 40] , dynamic formation shifting [45], etc. Hayes and Dormiani-Tabatabaei define flocking as "the formation and maintenance of coherent group movement" [44], p. 3900. From the perspective of swarm-based robotics, flocking tasks are accomplished using only local sensing. Individual agents rely on information such as the position and velocity of their nearest neighbors in order to plan movement [46]. Hayes developed robust flocking algorithms resistant to agent failure. These algorithms were then demonstrated successfully in simulation and with real robots. Their flocking experiments with real robots feature Moorebots, one of the popular choices for swarm-based robotics. A detailed description of Moorebots, which were developed and utilized extensively at CalTech, is found in [47]. Fredslund and Mataric [45] use the Pioneer2 DX from ActivMedia, Inc. as a robot platform to validate their simulation results predicting the ability of a robotic swarm to shift formations dynamically. Their physical experiments also highlight obstacle avoidance within the framework of a formation.

The above examples of flocking and formation control were developed independent of a specific application. However, much of the work done in robotic flocking relates to military applications [38, 40, 46]. Cooperative hunting has become of great importance to the military, due to a number of factors presented in [48]:

- Physical
    - Robots can access and maneuver in space too small for humans.
    - Robots survive in contaminated environments.
    - Robots can remain motionless indefinitely.
- Psychological
    - Robots are not influenced by fear in carrying out military operations.
    - Robots complete tedious tasks without suffering mental fatigue.

- Risk
    - Robots can perform kamikaze missions.
    - Inexpensive robots can be chosen for high-payoff/high-risk situations.

Balch and Arkin, [46], employ strictly behavior-based control to simulate complex formation control in a group of five virtual robots. Simple behavioral rules were used to generate formations such as *line*, *column*, *diamond*, and *wedge*. Formation location and centering was accomplished using one of three referencing methods: (1) unit-center-referencing, (2) leader-referencing, and (3) neighbor-referencing. Developed for the United States Army, this system could soon be found in real-world combat situations. Perhaps the most important researcher in the field of cooperative formation control for military use is Yamaguchi, from the University of Tokyo. Yamaguchi has studied combat-related behaviors such as surrounding a target [38, 40] and invasion prevention [49]. Target location and enclosure is one of the most impressive demonstrations to date of practical formation control in a group of mobile robots. Yamaguchi recently developed the control algorithms for implementing this behavior in a group of Hilare-type nonholonomic mobile robots [38], but experimental results with real robots have yet to appear. The near future promises to provide realistic physical demonstrations of flocking and cooperative hunting in the same vein as current simulation results.

Subsection 2.7.2: Object Manipulation

The second major class of tasks within swarm-based cooperative mobile robotics is object manipulation. This is a generic term that encompasses an array of subtasks. The subtasks can be organized into two main classes: payload transportation and strategic, competitive manipulation in a dynamic environment.

Payload transportation describes a subset of object manipulation tasks often used to demonstrate multi-robot cooperation. Cooperative transport by a group of robotic agents is often based on the behavior of ants and termites. Sudd's studies of cooperative transport in ants revealed that a group of ants will resort to cooperative lifting or dragging in the event that an item of food or prey is too heavy to be moved by an individual [50]. In physical robotics experiments, the problem is often formulated as box-pushing or cooperative lifting. Kube and Zhang [51] developed control algorithms for cooperative lifting, with special attention to stagnation recovery. They also simulated a group of robots in cooperative lifting situations. Bay, from Virginia Polytechnic Institute, describes the development of "Army-Ant" cooperative lifting robots for implementation as a flexible material handling system [52]. A slight variation from Bay's homogeneous group of robots is the leader/follow strategy employed in [53]. Both control schemes accomplish the basic task of cooperatively lifting a box and transporting the item to a specific location.

Competitive, strategic object manipulation within a dynamic environment is one of the most complex tasks in cooperative robotics. A wide variety of approaches exist for this class of tasks, ranging from artificial intelligence (AI) to swarm intelligence. AI approaches depend largely on high-level processing and reasoning abilities of each individual robot [11], while swarm intelligence techniques adhere to the tenets of Brooks' reactive, behavior-based control. The exemplification of this set of tasks is RoboCup, an international and educational research initiative in the form of an annual robotic soccer competition [54]. Each participant designs a group of mobile robots with the task of scoring as many goals as possible against a team of robot opponents. As in human soccer, a goal is scored by cooperatively manipulating a ball through a field and into a designated goal region. Papers abound describing various design and control schemes used in RoboCup competitions (e.g., [11, 39, 54-57]. Two examples of behavior-based approaches are found in [11] and [57]. In [11], Werger makes a convincing argument for the superior performance of behavior-based approaches to this task as compared to the classical AI methodology.

Subsection 2.7.3: Foraging

Cooperative foraging is one of the most important tasks in the study of multi-robot cooperation. Of the three major areas of application (formation control, object manipulation, and foraging), the latter has the closest ties to biological swarms. Therefore, foraging is also the obvious task undertaken to demonstrate swarm intelligence in a simulated or physical robotic system. Drogoul and Ferber [58] note that foraging is "widely accepted as the best illustration of 'swarm intelligence'…" (p. 1) and Cao, Fukunaga, and Kahng, [1] describe foraging as "one of the canonical testbeds for cooperative robotics" (pp. 3-4). As will be discussed in the following chapter, biological swarms are highly efficient examples of swarm intelligence. Foraging in ants and bees provides some of the clearest examples of swarm intelligence in nature. Further, these examples from nature often motivate the models upon which foraging robot systems are designed. This tight correlation between robotic and natural foraging provides the impetus for a multitude of efforts to demonstrate swarm-like robotic cooperation in foraging.

The fundamental definition of foraging is the location and collection of objects (or targets) [58]. In biological systems, the targets are generally considered to be food or prey. For robotic implementation, the objects to be collected are application dependent. In the research and development of foraging swarms, the objects often chosen are small "pucks" or discs capable of being grasped and released by a robotic end effector. Note that the generalized definition of foraging makes no constraint on the location of object aggregation. However, the most common type of foraging is central-place foraging. In fact, the terms have become synonymous in the swarm intelligence literature. Central-place foraging describes a system in which objects are returned to a "home," which is usually located at or near the center of the foraging environment [59]. The spatial and temporal distribution of objects within the foraging environment is unspecified by this definition. Researchers have studied a wide range of distribution schemes, including uniform, random, and clustered spatial distributions, as well as the introduction of objects at varied intervals in time. Object distribution and the number of foragers tend to be the most important parameters studied with respect to system behavior.

Before discussing the historical and recent attempts to demonstrate cooperative foraging, it is instructive to consider the methods of foraging used by traditional robotics. These methods are generally employed in the case of a single robot, but are also used in non-cooperative multi-robot systems. Werger and Mataric [41] identify the following single-agent foraging methods:

- The Omniscient Planner: The use of a planner that is aware of the entire foraging environment, including the position of the forager.
- Position/Orientation Sensing: The use of absolute global position information, including GPS, compass, radio-sonar positioning, and/or dead reckoning systems.
- Taxis: Following some sort of beacon.
- Unique Location Recognition: The use of local environmental features to identify landmarks to which the agent orients itself.

While some of these methods are also useful in cooperative multi-robot foraging systems, they tend to be utilized less than methods unique to swarm-based robotics.

The first main approach used in swarm-based foraging is an exception to the above generalization: beacons. Beacon-following methodology characterized the earliest efforts at cooperative robotic foraging (e.g., [16, 41, 60]), and persists in more recent research efforts [7, 43]. Beacon and beacon chains emerged as an initial solution method in the early 1990s. Goss and Deneubourg, [16], pioneered the field of cooperative foraging, or "harvesting," in 1992 with one of the first discussions of this topic. Drawing upon their experience in social insect behavior, Goss and Deneubourg simulated a group of mobile robots that form dynamic chains of beacons within the foraging field. Their simulation results predicted that future attempts at physical implementation would succeed in efficient cooperative foraging. The more common beacon-based approach is for each forager to broadcast a signal upon location of a group of targets. Other robotic agents are able to recognize the signal and engage in homing behavior to locate the "food" source. Altenburg [60], one of the earliest researchers to conduct foraging experiments with real robots, designed a group of robots utilizing beacons and homing to achieve central-place foraging. Sugawara, Sano, and Watanabe [7, 43] have conducted similar experiments with real robots to demonstrate swarm intelligence via successful foraging. Beacon signals range from visible light [7, 43] to modulated infra-red (IR) [60] to auditory communication [61]. Werger and Mataric [41] present a novel extension of beacon-based foraging by eliminating the need for a traditional signal. Their approach consists of constructing physical chains of robots to guide foragers to targets. Arranged and maintained only through touch sensing, the robotic chain functions as a hybrid of a beacon and a trail. This hybrid technique of creating chains of foragers, studied theoretically in [58], represents a conceptual segue to trail-based approaches.

The fundamental problem in cooperative foraging is defining the nature of the inter-robot cooperation. In general, some form of direct or indirect communication is employed to produce cooperation and improved task performance. The method of trail-based foraging moves further along the spectrum from direct to indirect communication. Scientists have long understood the effectiveness of trail laying and following in the foraging strategies of social insects, such as ants [62]. These biological swarms provide inspiration for recent work in robotic foraging. Despite the large amount of theoretical simulation of trail-based foraging in robotics (e.g., [39, 40, 43, 58, 63]), which is discussed in more detail in Chapter 4, physical experimentation is much less developed. In fact, robotics researchers have yet to successfully demonstrate trail-based foraging in a physical system of robots. One of the early attempts to perform robotic trail-laying and following actually used toilet paper as the trail [64]. The greatest progress toward trail-based foraging comes from the work of Russell et al., from Monash University in Australia, who studies trails with the general aim of improved mobile robot navigation. In a series of papers [24, 65, 66], Russell presents successful development of robots capable of laying and following two types of trails. In both cases, the trails are designed for use as short-lived navigational markers in a variety of applications. Russell, [24], describes the design and testing of robots capable of laying and following a heat trail. The heat trail is generated by an on-board quartz-halogen lamp directed toward the floor by a parabolic reflector. This trail is detected by a pyroelectric sensor sensitive to infra-red energy emitted by the heat trail. These robots were demonstrated to successfully follow the heat trail up to 10 minutes after application. Russell notes in conclusion that, despite a successful demonstration, the halogen heat source "is a considerable power drain for a mobile robot" [24], p. 431. Deveze et al., [65], and Russell, [66], suggest the use of chemical markers as "virtual umbilical trails indicating the route back to their starting position" [66], p. 5. The chemical marker or odorant, camphor, is chosen for non-toxicity, invisibility, safe use on floor materials, and slow sublimation rate. Each robot was also equipped with a piezoelectric quartz crystal gravimetric sensor, allowing the robot to track the odor trail with considerable accuracy. The main drawback of this approach, as acknowledged by the author(s), is the slow response time of the chemical sensor (several seconds). While neither of these demonstrations attempted to display group foraging using trail-laying and following, they represent the most advanced examples of robotic trail-based navigation.

Along the continuum from direct to indirect communication, the far extreme of indirect communication is represent by “stigmergy.” Stigmergy refers to indirect communication through the environment, which is also termed “implicit communication" [57] . A more precise definition, given in [12], identifies stigmergy as “the production of a certain behaviour in agents as a consequence of the effects produced in the local environment by previous behaviour” (p. 181). This term has been adopted from the biological swarm intelligence community, where the term originated in 1959. A French biologist, Grasse, gave this name to the emergent cooperation observed in nest building of termites [67]. The most conservative view does not consider trail laying and following as an example of stigmergy, due to the fact that the trails are deposited with the purpose of communication. However, the concept has developed in time to include trail-based foraging as well as traditional insect corpse-gathering and some forms of nest construction. Pure stigmergy--interaction through the environment without the intent of communication--is the inspiration for the third major method of swarm-based robotic foraging. This method of foraging is actually adopted from observations of corpse-gathering, not foraging, in insects. Beckers, Holland, and Deneubourg [12] designed a robotic foraging system using pure stigmergy. In order to collect a random distribution of pucks in an experimental foraging environment, the robots pick up or release each puck based solely on the local puck density. In areas of low puck density, the robot has a high probability of collecting one or more pucks. Where puck density is high, the robot tends to release all of its pucks. In this manner, the pucks are eventually arranged in a single cluster. Technically, this example does not fall under the heading of central-place foraging, due to the fact that the pucks are not clustered in a predetermined location.

Despite this caveat, the multi-robot system forages successfully and achieves cooperative behavior with completely indirect communication.

Subsection 2.7.4: Clustering and Sorting

The task of cooperative sorting in swarm-based robotics is closely related to foraging. In fact, many sorting experiments emerge from robotic demonstrations of foraging [12], or vice versa. Both tasks take direct inspiration from the behavior of social insects. While foraging corresponds directly to biological food searching, sorting reflects a variety of clustering behaviors. For example, the brood sorting or corpse gathering behaviors of ants and termites are well-documented [8, 62, 68-72]. This body of literature observes the efficiency of sorting with only stigmergic cooperation. Deneubourg et al. [25] laid the theoretical groundwork for a swarm-based robotic sorting system. Basing their analysis on biological data, these authors developed a simple probabilistic model of collective sorting and generated simulations of robotic swarms performing similar tasks. More recently, two groups of researchers have sought to implement sorting with real robots [26, 73]. Martinoli, Ijspeert, and Mondada [73] use the popular Khepera mobile robots to achieve cooperative sorting under the influence of probabilistic behavioral control. The sorting takes place according to a probabilistic model very similar to that of [25] and [12]. Clusters of items are incremented or decremented on the basis of local item density and cluster geometry. These simple rules are sufficient in [73] to generate a single cluster of 20 objects in approximately 3 hours using 10 Khepera robots. The work of Melhuish, et el., [26] represents the first successful demonstration to date of robotic sorting of two dissimilar types of targets.

Using red and yellow plastic discs as targets, Melhuish's group of robots achieved segregated clustering in the span of several hours. While the task efficiency (as measured by completion time) is fairly low in this experiment, it is an important first step toward more efficient swarm-based sorting. Cooperative sorting of dissimilar targets promises to soon be an area of greater research focus. In fact, Bonabeau, [68], issues challenges to the swarm intelligence community for future robotics research on the topic of swarm-based sorting.

Subsection 2.7.5: Miscellaneous Applications

Although the majority of swarm-based robotic tasks lie within the bounds of one of the above categories, a number of isolated special cases exist in the literature. These outliers compose a group of other miscellaneous tasks related to swarm intelligence. Some of the interesting tasks being studied are:

- Self-replication (started by [3]; see Ch. 6 of [68] and references therein)
- Reconfigurable robots ([74, 75])
- Odor localization ([76, 77])
- Geographical exploration and map-making ([1, 22])
- Military reconnaissance and surveillance ([48, 78])

Section 2.8: Conclusion

The conceptual and historical roots of swarm intelligence have been presented from the perspective of cooperative mobile robotics. A number of popular control architectures were described; the most significant of these is Brooks' behavior-based, subsumption, approach. Physical morphologies were described and compared, as well as a brief discussion of learning in multi-robot systems. Finally, the canonical tasks for robotic swarms were discussed, with particular focus on foraging.

Chapter 3: Biological Inspirations

At this point, the reader is reminded that two complementary disciplines converge at the point of swarm intelligence. The engineering background of the field has been discussed, leaving the important biological inspirations for swarm-based systems. The term *swarm* naturally evokes thoughts of living systems such an angry beehive. This biological association makes perfect sense when one remembers the influence of natural systems on pioneering robotics researchers such as Beni [5], Brooks [10], et al. Within the ethology community, swarm intelligence is part of a broader class of phenomena know as self-organization (SO). Self-organization and swarm intelligence are sometimes used synonymously, but the former term is less common outside of the biological community. Swarm intelligence can be considered to be the application or engineering branch of self-organization research. Due to the predominant usage of the term self-organization in the ethology literature, this term will be preferred in the remainder of this chapter.

Section 3.1: Self-Organization in Biological Systems

Subsection 3.1.1: What is Self-Organization?

In their authoritative book on self-organization, Camazine et al. present the following definition of SO:

> *Self-organization is a process in which patterns at the global level of a system emerge solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's*

components are executed using only local information, without reference to the global pattern. [70], p. 8

This definition attempts to be as precise as possible while maintaining generality to include the wide array of manifestations of self-organization. The multitude of examples of SO is attested to by the origins of the term itself. A few examples are sand grains forming rippled dunes, swirling spirals emerging from chemical reactants, cells generating highly structured tissues, geese flying in a V-shaped formation, fish traveling in schools, and, of course, a host of examples from bees, wasps, ants, and termites. Biological scientists actually adopted the term after its introduction in the context of physics and chemistry. It was introduced to describe the emergence of macroscopic patterns and structures out of processes and interactions defined at the microscopic level [68]. Ethologists later extended the term to apply to a wide range of collective phenomena in animals, especially social insects [69]. Yet another definition from Bonabeau and the Brussels group (Theraulaz, Deneubourg, and Camazine) focuses more directly on ethological SO:

> *[SO] does not rely on individual complexity to account for complex spatiotemporal features that emerge at the colony level, but rather assumes that interactions among simple individuals can produce highly structured collective behaviours*. [69], p. 188

Before moving on to discuss how self-organizations works, it is important to clarify the definitions provided for SO. The terms “complexity” and “patterns” have been supplied as descriptors of global system characteristics in a self-organized system. These terms can be rather nebulous, requiring further elaboration and specification. First of all, complexity is a relative term in these definitions [70]. Self-organization produces complex behavior or complex patterns *compared* to the complexity of the individual agents producing the global organization. In social insects, the complexity of an individual is simply insufficient to explain the organization, flexibility, and robustness exhibited in an insect colony. The concept of a pattern is another potentially ambiguous term found in definitions of SO. Whether a pattern is composed of living units or inanimate objects from the environment, it is always an organized arrangement of objects in space or time. In SO, these patterns emerge without any direction or orchestration from external influences. Examples of biological patterns include raiding columns of army ants, synchronous flashing of fireflies, termite mounds, pigmentation patterns on shells, etc. [70]. The next subsection moves from defining self-organization to understanding the mechanisms that produce system complexity from local simplicity.

Subsection 3.1.2: Mechanisms of Self-Organization

It has been established while defining self-organization that system complexity is produced through many interactions between individuals. Therefore, the first and most important mechanism of SO is interactions. The nature of these interactions is the key to investigating any self-organized system. Two general forces influence the frequency of these interactions: positive and negative feedback. Finally, self-organized systems often rely on the potential benefits of random fluctuations. Together, the following four elements are the major mechanisms of SO:

1. Interactions
2. Positive Feedback

3. Negative Feedback
4. Random Fluctuations

These four mechanisms are discussed below in further detail.

Interaction is the only essential mechanism required for self-organization or swarm intelligence. The nature and extent of interactions between individuals varies, but their existence is a clear prerequisite for the complex system behavior evident in self-organized systems. In biological systems, these interactions often allow an individual to obtain the information used to determine a response. Gathering information from an interaction is generally a result of some type of communication with nearest neighbors. In the case of flocking (and many other examples), however, the local information obtained in each interaction is simply the position (and possibly velocity) of a handful of nearest neighbors. This information is sensed directly; there is no need for each neighbor to communicate directly or indirectly. It is also unnecessary to leave some sort of environmental marker to communicate via the environment. In cases such as these, sensing information during an interaction is sufficient to produce complex system behavior. In most other cases, the information is gained through communication intended to convey information, or direct communication. A clear example of this type of interaction is the well-known dancing performed by some species of bees. When a foraging individual returns to the hive laden with nectar, it performs a “dance” that conveys the approximate location of the food source [69, 70, 79, 80] Often, however, the interaction and communication take place indirectly. This is the case in the trail-laying and following behavior of many species of ants [62]. Trails are laid with the intention of conveying directional information to other ants, but the communication is indirect in the sense that a trail is neither individual-specific nor time-specific. It can be followed at any time (until it evaporates) and by any individual forager. Another type of interaction produces information transfer through the environment. This has also been referred to as *implicit* communication, or, more commonly, stigmergy. Derived from the Greek *stigma* (sting) and *ergon* (work), Grasse introduced the term in his studies of task coordination and nest reconstruction of termites [67-69]. In termite-mound building, an individual termite is more likely to release a grain of building material on a preexisting mound. Each builder is also more likely to pick up a grain of dirt or sand in a flat area. In this way, each termite responds to the actions of other termites *through the environment*, and complex nest structures result [68].

Positive feedback, or amplification, is a common mechanism in self-organized systems. Positive feedback promotes radical changes in the system by taking an initial change and reinforcing it in the same direction. An important example of positive feedback, or autocatalysis, is found in the trail-based foraging of ants [62, 70, 81]. When a single forager discovers food, it leaves a pheromone trial while returning to the nest. Other ants follow this trail from the nest to the food source and reinforce the initial trail as they return home. In turn, more foragers leave the nest and so on. As a result of positive feedback, the pheromone concentration of the trail increases rapidly, as does the number of ants leaving the nest.

Negative feedback balances the effects of positive feedback and stabilizes collective patterns. The catalytic nature of positive feedback requires an opposing force in most cases. Otherwise, systems could commit unreasonable amounts of resources to a particular activity. In fact, negative feedback often occurs due to depletion of limited individual or system resources. Negative feedback takes forms such as saturation, exhaustion, overcrowding and/or competition. Returning to the example of foraging ants, negative feedback stems from a limited number of available foragers, colony-level satiation, food source exhaustion, local crowding at the food source, competition between food sources, and/or pheromone evaporation [68].

Random fluctuations are surprisingly common in biological systems that exhibit self-organization. Many of these systems actually rely on certain stochastic elements for behavioral flexibility. The amplification of these random fluctuations (random walks, errors, etc.) allows for the discovery of new solutions as well as acting as seeds from which new structures can nucleate and grow. An excellent example of this is caused by stochastic trail-following in ants. It is well-known that ants follow trails imperfectly, especially trails with low pheromone concentrations [71, 82]. When an individual loses the trail and becomes "lost," it has the potential to "stumble across" an undiscovered source of food [83]. This could be a better (e.g., closer, richer, larger) food source than that currently being exploited by the colony. Clearly, random fluctuations are vital to efficient self-organization.

Subsection 3.1.3: Characteristics of Self-Organization

After discussing the internal mechanisms of self-organization, the perspective now shifts to an external, global viewpoint. The mechanisms of interaction presented in the previous section arise from the perspective of an observer focusing on individual behavior. Identifying global characteristics of these systems requires a much broader perspective—that of an external observer viewing the result of interactions, rather than the interactions themselves. Three primary characteristics of any self-organized biological system are presented below:

- Emergent Pattern Formation – As discussed above, the creation of complex spatiotemporal structures in initially unstructured media is omnipresent in self-organized systems. The term "emergent" refers to a property that arises "…unexpectedly from nonlinear interactions among a system's components" and that "…cannot be understood as the simple addition of their individual contributions" [70], p. 31.
- Multistability – The possible coexistence of multiple possible stable states, or attractors, is known as multistability. Depending on the initial conditions in a self-organized system, there can be a number of stable states to which the system evolves. A range of initial conditions corresponding to a particular stable state is a *basin of attraction* for that state, or attractor. For example, in ant colonies utilizing mass recruitment where two equal food sources are present, the colony tends to exploit one of the two sources. There are two stable states; the one that is actually chosen depends partly on randomness, but also on whichever source is discovered first [84].
- Bifurcations – Dramatic changes in system behavior, or bifurcations, occur in many self-organized systems. Camazine at al. define bifurcations as "the appearance of a qualitative change in behavior when a parameter-value changes quantitatively" [70], p. 33. In response to variations in certain system or environmental parameters, a system can switch from one

state to another through bifurcation phenomena [85]. Bonabeau and Cogne [86] contend that some biological systems ensure adaptability by maintaining themselves in the vicinity of a point of instability (a bifurcation point).

Section 3.2: Microscopic Organisms

Self-organization has been observed even on the simplest biological scale: microscopic organisms. Much of the research in this area relates to the human immune system. Since there are several main types of microorganisms within the immune system, it can be considered a heterogeneous system of self-organization. The key elements of the human immune system are lymphocytes (B-cells and T-cells), antibodies and antigens [87]. In a very simplified explanation, the lymphocytes produce antibodies and the antibodies recognize and eliminate antigens, which are unwanted infectious agents. The human immune system of an adult male contains over a trillion lymphocytes, which together utilize about 10^{20} antibody molecules [23]. Via countless interactions between these cells, the human immune system is able to self-organize in response to the dynamic environment that is the human body. Recognizing this efficiency, robotics researchers have developed swarm-based robotic systems using the human immune system as a model [23, 87-89].

Various species of unicellular organisms—bacteria, myxobacteria, myxomycetes, and cellular (amoebic) slime molds—exhibit remarkably complex system behavior despite the clear lack of individual intelligence or planning ability. Myxobacteria, for example, are predators that feed on other microorganisms. They travel in large clusters, or swarms, that demonstrate highly coordinated overall movement. In some species, periodic waves of movement, or ripples, have been observed [70]. Many types of bacteria create highly complex spatial patterns when grown on agar plates. *Eschera coli*, *Salmonella typhimurium, Bacillus subtilis*, and other species produce patterns such a spirals or concentric rings under specific environmental conditions [70]. The classic example of unicellular self-organization is that of the slime mold. These amoebas exist independently when food is plentiful, but during times of starvation, they aggregate into one cluster that begins to act as a multicellular creature [70, 84]. This creature, known as a "slug," composed of 10,000 to 100,000 cells, sometimes develops a well-defined body and even reproduces. Studies have shown that this process occurs in a completely homogeneous group; that is, there are no specialized "leader" slime molds [90]. The process of slime mold aggregation is one of the most complete examples of self-organization in any biological system [84].

Section 3.3: Social Insects

Subsection 3.3.1: Foraging

Ant colonies have come to be viewed widely as a prototypical example of how complex group behavior can arise from simple individuals [84]. For this reason, the behaviors of ant colonies have been studied with far more intensity than those of any other social insect. This trend is also due to the ease of observing and experimenting with ants. They are readily visible and respond well to testing in laboratory situations. Ants are robust in their behaviors, meaning that fundamental changes do not occur when their environment is altered slightly for observation or testing.

OVERVIEW

Foraging behavior in ants occurs in a wide variety of strategies, with several of these strategies exhibiting characteristics of self-organization. Some species utilize a combination of these strategies, depending on environmental conditions such as food availability and distribution [91-93]. This adaptability in ant foraging systems is discussed later in this section. Other species exclusively employ a single foraging method. A number of authors have attempted to classify the multitude of foraging methods used by ants, with two classic works containing the most important classifications [62, 94]. Oster and Wilson provide a fairly broad classification, identifying five basic foraging methods. These methods are presented below in Table 3.

Table 3. Foraging Methods Used by Ants (adapted from [94], pp. 248-251)

Foraging Method	Description
I	Workers leave the colony singly and retrieve prey and other food items as solitary individuals.
II	Solitary foragers signal the location of the food to nest mates by means of odor trials, ritualized “dances,” or some other mode of directional communication.
III	Foragers move away from the nest along trunk trails, departing at intervals to search unmarked terrain as individuals.
IV	Solitary foragers recruit fellow workers to a food source using odor trails; the group then assaults or collects the food as a group.
V	Workers proceed from the nest in bands or entire armies of individuals to engage in group hunting.

Holldobler’s The Ants [62], which has become the Bible of myrmecology (the study of ants), provides a more thorough system of classification. Holldobler’s classification is preferred in this work for its thorough treatment of all observed strategies. This system of organizing the foraging methods breaks all phenomena into three categories: hunting, retrieving, and defense. Three to four possibilities exist within each of these categories, presenting the possibility of 48 possible three-state foraging strategies.

Hunting

(1) by solitary workers

(2) by solitary workers directed to search sectors by trunk trails

(3) by groups of workers searching together

Retrieving

(1) by solitary workers

(2) by individuals who return home along trunk trails

(3) by individuals recruited to a food source by scouts

(4) by groups of workers who carry items as a group

Defense

(1) by guard workers during hunting

(2) the absence of such defense

(3) by guard workers during harvesting/retrieval

(4) the absence of such defense

While a full accounting of ant foraging strategies has not been completed, Holldobler suspects that virtually all of the 48 strategic combinations are employed in some species of ants [62]. In the remainder of the discussion, attention will be focused on the first two categories: hunting and retrieving. Combinations of techniques from these categories represent fairly complete systems of behavior, with defensive strategies coming into play only in the event of an attack of some sort to the foragers. Defensive behavior is also less related to self-organization, so it will be overlooked in this investigation. The following two subsections will look more closely at hunting and retrieving strategies, respectively.

HUNTING

Hunting techniques range from solitary exploration to group raids. Where each species falls on this continuum depends on factors such as the size of the colony, the climate, and the nature of the food. It is important to note that, while "hunting" connotes the active tracking and elimination of live prey, it is used in a more general sense in this context. Hunting also refers to the more common situation of searching for inanimate food items. Colonies most often employ solitary foragers in situations where the food distribution is characterized by a large number of widely distributed small food items [92]. This type of hunting is also referred to as *diffuse foraging*, due to the more or less random "diffusion" of workers from the nest [94]. In many species, such as *Cataglyphis bicolor*, solitary foragers set out from the nest in a generally straight line extending radially from the nest in a specific direction [62]. Initially, this direction is selected at random, but an individual worker tends to maintain a generally constant foraging direction on successive forays, especially if food was located on a prior trip. A particular direction will be abandoned if many unsuccessful trips fail to locate food, but an individual tends to persist in one or a very few directions throughout their lifetime [62]. Random fluctuations in foraging direction have been proven to improve deployment efficiency. Unpredictable deviations from the previous path "increase the chance that the workers will strike food items missed in earlier efforts" [62], p. 245. This type of hunting, combined with solitary retrieval, is thought to be the most primitive method in evolutionary terms [94].

The next type of hunting, which relies on trunk trails, has been studied extensively in the self-organization community [68, 70, 95-98]. Harvester ants are one species that utilize trunk trails to direct solitary foragers [62]. This species tends to exploit patchy food supplies with limited spatial distribution. Therefore, trunk trails leading to regions of locally high food density represent an effective biological adaptation of foraging behavior. Individual workers stray from the trunk trail to forage in the local vicinity for items of food [94]. Their chances of finding food are increased greatly by straying short distances from the trunk in short looping patterns.

African army ants are famous for their raids composed of millions of completely blind workers [70, 94]. The raid system is composed of a dense swarm front, a large delta-shaped region of intermediate trails, and finally one principal trail leading back to the nest. The swarm front is linked by a series of looping trails back to reinforcements arriving along the main trunk trail. Despite the rapid dynamic growth of such swarming raids, the trail is surprisingly stable. In a single day, an *E. burchelli* raid can travel over 100 meters through the jungle with very high trail fidelity [70]. While the details are not fully understood, scientists know that individuals periodically deposit pheromones along the trail, creating a stable trunk of high pheromone concentration through the mechanism of positive feedback. This is the clearest and most impressive example of ants hunting in groups.

RETRIEVAL

Prey (or food) retrieval is the second major category of foraging techniques. As presented above, there are four major retrieval techniques: (1) by solitary workers, (2) by individuals who return home along trunk trails, (3) by individuals recruited to a food source by scouts, and (4) by groups of workers who carry items as a group. The first of these four cases is the simplest technique, as it requires absolutely no cooperation between individual workers. A worker simply locates a food item and navigates toward the nest without the aid of pheromone trails or any other assistance from the colony. Navigation on the trip home is achieved in a number of ways [84]. Visual cues sometimes play a role, as ants can reference the sun, moon, physical landmarks, or an image of the forest canopy for orientation [62, 93]. In some other species, it is believed that memory alone allows for navigation, although this hypothesis has not been solidly established [62].

Trunk trails were described above in reference to hunting techniques. The existence of trunk trails makes prey retrieval a trivial task. These well-defined ant "highways" make the homing process very simple. Even in the case of the blind African army ant, homing is accomplished with high accuracy and speed by following the main trunk trail back to the nest.

The third retrieval category and, to some extent, the fourth, both assume that some type of recruitment has taken place. Group transport of a food item can occur without recruitment in the case of hunting in groups, as described above. This is the only exception to the use of recruitment, which is the predominant foraging method. Recruitment, in one form or another, is extremely common in ants and can be found in nearly every species. Recruitment is also the basis for the majority of investigation into self-organization in ant foraging. For this reason, recruitment is presented as a separate topic below. In that context, the two remaining retrieval strategies are covered.

RECRUITMENT

Recruitment strategies represent the most highly-evolved prey and food retrieval mechanisms in the insect world. Recruitment is defined as "…communication that brings nestmates to some point in space where work is required" [62], p. 265. This definition includes recruitment utilized for colony defense, nest construction and repair, emigration to new nest sites, and a variety of other colony activities. However, this discussion of recruitment will be limited to the context of foraging. Holldobler divides the various types of foraging recruitment into four groups [62], which are outlined below:

(1) Tandem Running – First, an individual worker returns to the nest after finding a food source. This worker employs tactile and/or visual stimulation to invite one worker on the journey back to the food source. On the return trip, the two run in tandem, maintaining close physical contact during the entire trip. As positive feedback takes effect, the number of foragers doubles on each successive trip (assuming that the food source does not become exhausted or overcrowded, which introduces negative feedback).

(2) Group Recruitment – Similar to tandem running, a laden worker returns to the nest and signals mainly via touch to invite a group of 2-10 nest mates to the source [93]. The recruiter then leads this group to the food source, often while depositing a pheromone trail. Other workers do not follow the trail without

tactile stimulation *and* an escort from a recruiter. This type of recruitment is common in species employing cooperative group transport.

(3) Trail-based Group Recruitment – This process is closely related to the previous technique, but with less dependence on leadership behavior. The recruiter conducts similar invitation behavior to a group of recruits at the nest *after* laying a pheromone trial from the source to the nest. In most cases, the leader returns to the food source, but this is not necessary for other recruits to follow the trail. Workers thatdid not witness the invitation behavior will rarely follow the trial.

(4) Mass Recruitment – In the early 1960s, entomologists first discovered mass chemical communication in ants, which is one of the most complex forms of social behavior in the social insects [62]. A worker lays a pheromone trail back to the nest and continues foraging (usually by following its own trail). There is no ritualized display, tactile, or acoustical communication with other workers at the nest. The pheromone trail alone provides sufficient impetus to draw foragers from the nest to the food source [86]. Other workers foraging in the field also follow the trail if they wander into its active space. Those workers who begin following the trail at some intermediate point need some way of knowing which way leads to the food source. This binary decision is made either through *a priori* knowledge of the direction of home (based on navigational cues, memory, etc.) or by encountering other workers along the trail. If a worker is met head-on laden with food, the trail-follower interprets this information to understand that it is headed in the correct direction. It has

been established that directional information is most likely not encoded in pheromone trials [62].

Recruitment is sometimes simply regarded as a method of assembling many workers in a specific location. That is hardly the case; rather, sophisticated foraging recruitment, such as mass recruitment, actually enables the colony to make decisions through the mechanisms of self-organization [70]. Ant colonies are able to collectively choose the most profitable food source from among a number of options, using only positive and negative feedback, and amplification of random errors. Consider the example simulated by Resnick in [84]:

> *So what happens when ants are released in an environment with three food sources? In some ways, it is helpful to think about the food sources as competitors, each trying to attract a stable trail of ants. In this competition the food source closest to the nest has two advantages: it is the one most likely to be discovered in a random walk from the nest, and it has the lowest critical density (that is, it needs the smallest number of ants to form a stable pheromone trail). So as the ants march out of the nest and explore the world, the…colony's first stable trail is likely to go to the closest food source. Once an ant joins a stable trail, it is unlikely to leave…so ants on the stable trail are taken out of circulation. Assuming a fixed supply of ants, there are fewer free ants remaining to explore and form trails to the other food sources. …But once the closest food source is fully depleted, the situation changes. The pheromone trail to that source dissipates, freeing the ants that had been gathering food along that trail. After that, there are again enough ants to form a new trail.… In this way the colony*

exploits the food sources one by one, in a seemingly planned fashion, moving outward from the closest food source to the most distant.[84], p. 67

In this example of three food sources at different distances from the nest, we see all of the major mechanisms of self-organization (positive feedback, negative feedback, and amplification of random errors) producing the characteristics of emergent pattern formation, multistability, and bifurcation phenomena. Positive feedback initially catalyzes the random discovery of a food source [81], creating the initial pheromone trail. This trail is reinforced by positive feedback, generating the emergence of a pattern in the form of the stable trail. This pattern formation could occur at any one of the three food sources, with unequal probabilities. Each stable trail is possible, representing three possible stable states—multistability. Negative reinforcement caused by pheromone evaporation and food source exhaustion allows the system to pass through a bifurcation point, observed as a rapid shift in behavior from focusing on one source to again searching for a source to exploit [85]. Experimental and theoretical studies have reproduced this type of collective decision-making, proving the self-organization allows a colony to choose between sources on the bases of distance, quality, and/or inter-colonial competition [70, 82, 85, 86, 99]. It becomes clear, even in such a simple example, why mass recruitment is one of the purest examples of self-organization [70].

COMMUNICATION

All forms of recruitment inherently require some type of communication between individuals. Ants have the ability to communicate via a number of modes, most of which are employed in recruitment behavior. The above discussion of the types of recruitment intentionally simplified the details of communication. The aim of this section is to present the various ways in which ants perform the communication necessary for recruitment. The major types of communication used by ants are chemical, tactile, visual, and acoustical.

Chemical communication is by far the most common form of communication used by ants for any purpose. Holldobler notes that an average ant colony possesses between 10 and 20 kinds of signals, with most being chemical. He contends that "…pheromones play the central role in the organization of ant societies" [62], p. 227. It is extremely important for a number of colony functions, including worker recruitment in foraging. Of course, in mass recruitment, workers rely solely on chemoreception to obtain information regarding the location of a food source. The primary means of chemical communication is pheromonal deposition and sensing. Mandibular regurgitation and fecal odor sometimes convey information, but pheromones are far more important. Pheromones are a type of semiochemical. A semiochemical is defined as "…any substance used in communication, whether between species (as in symbioses) or between member of the same species" [62], p. 227. A pheromone is somewhat narrower in meaning: "A pheromone is a semiochemical, usually a glandular secretion, used within a species" [62], p. 227. These chemicals can be classified as either olfactory or oral, according to the site of their chemoreception. Individual workers follow the "vapor tunnel" created by evaporation and diffusion of the deposited pheromone. A semi-ellipsoidal-shaped active space exists surrounding the central liquid trail, within which ants are able to detect the presence of a pheromone. The size of this active space depends on trail evaporation, diffusion rate, and the threshold concentration at which an individual responds. Workers have sensitive chemoreception systems, allowing them to move up gradients of molecular concentration, a process of orientation known as osmotropotaxis [62]. Intensive study of ant physiology has also uncovered the anatomical origins of many of the chemical systems. Without going into unnecessary detail, it is sufficient to identify the hindgut and midgut as the production sites for most recruitment-related pheromones [100, 101]. The Dufour’s gland is particularly important in pheromone production [93]. Trail-laying behavior in ants often consists of pheromone production in the Dufour’s gland, followed by deposition via the anus. Myrmecologists suspect that trail-laying behavior may have evolved as an adaptation from defecation [62]. Early ants may have responded to the odor of fecal matter deposited by workers returning from a source of food. This unintentional communication may have developed into the production of highly-specialized pheromones as a recruitment strategy.

Tactile communication, while far less important than chemical, is commonly used in recruitment strategies. In all recruitment types other than mass recruitment, tactile communication plays a potential role. The invitation behavior commonly seen in tandem running or group recruitment is commonly characterized by tactile communication [93]. When a worker ant returns to the nest laden with food, it signals to recruits by tapping “the nestmate’s body very lightly and rapidly with her antennae, often raising one or both forelegs to touch the nestmate with these appendages as well” [62], p. 258. The recruiter then commences in leaving a trail and/or physically leading one or more followers to the food source. Experimentation has shown that workers will not follow the recruiter in the absence of this tactile communication (see [62, 101], and references therein). In the case of tandem running, the leader and follower communicate continuously along their journey back to the food source. If contact is broken between the two, the leader will stop and wait for contact to be reestablished before continuing.

Visual communication at any range beyond one or two body lengths has not been documented in any species of ants, despite the excellent vision and motion recognition in some species [62]. In recruitment behavior, extremely short range visual signaling sometimes accompanies the tactile signals described above. Invitation behaviors sometimes include ritualized dances characterized by jerking movements intended to resemble turning toward the food source [94, 101]. These behaviors are always accompanied by tactile stimulation, however, making visual communication of only minor importance.

Acoustical communication is much less developed that chemical or tactile communication in ants, but some species do generate sound to accompany other forms of recruitment communication. Acoustical communication takes two forms: drumming and stridulation. Drumming describes the rhythmic beating of an ant's body on the substrate (soil, clay, tree branch, etc), while stridulation results from the rubbing together of specialized body parts to produce a "chirp" like a cricket [93]. In both drumming and stridulation, communication is far more effective on firm ground. This is because of the heightened ability of ants to detect vibration via a solid substratum compared to their near deafness to airborne vibration. In species such as *Aphaenogaster*, *Leptogenys*, and *Messor*, acoustical signals enhance the effectiveness of pheromones during recruitment [62].

ADAPTABILITY IN FORAGING

One of the defining characteristics of foraging behaviors in ants is adaptability. Ant colonies are well-known for their ability to adapt to a variety of changes in the environment. While some species of ants utilize only one method of foraging, most species employ a number of variations, depending on a number of external factors [62]. Some types of ants, such as the African weaver ant, utilize more than five distinct foraging strategies [101]. A number of researchers have explored the capability of ant colonies to respond collectively to environmental conditions. Studies of the European harvester ant, for example, show that solitary foraging is employed for initial discovery of food sources and for collection of widely distributed seeds. However, upon discovery of a large food source, or in the case of sparse seed distribution, a recruitment system is used. This type of adaptation ensures that resources are not wasted in conditions of meager food availability. Three species of desert ants demonstrate similar behavior in response to variable food density. They also modulate the number of active foragers according to altitude and season of the year [92]. Beckers, Deneubourg, and Goss, [99], and Holldobler, [62], discuss modulation of pheromone trail intensity based on food source quality. When returning from a rich food source, ants were shown to deposit more pheromone than on a return trip from an average food source. Trail networks have even been shown to transition from well-defined trunk trails to more traditional recruitment networks [81, 97]. Bonabeau and Cogne suggest a possible mechanism behind such adaptations, showing that ant colonies may maintain responsiveness to the environment by allowing certain colony-level parameters (e.g., the number of active workers) to oscillate in time [86]. Foraging adaptability is very common in ant species, as it provides for increased foraging efficiency.

Subsection 3.3.2: Sorting

The efficiency with which insects perform the related tasks of clustering and sorting has stimulated researchers to design new swarm-based algorithms for data analysis and graph partitioning [68]. Both clustering and sorting are found predominantly (and studied most thoroughly) in ants. Several species are known to cluster corpses within the nest to form a “cemetery.” Others sort larvae into several piles based on the age and size of each larva. While neither of these behaviors is fully understood, simple models in the vein of self-organization have generated similar patterns to those found in ant colonies [68]. These models hypothesize that clustering and sorting occur when agents move randomly in space and pick up or deposit items on the basis of local information. Observations of corpse clustering in *Lasius niger*, *Messor sancta*, et al., suggest that positive feedback is the dominant mechanism behind clustering [68]. Workers tend to deposit each corpse near another corpse or pile of corpses. In another example of stigmergic coordination, hundreds or thousands of ants interact through the environment in this manner to produce larger and larger piles or corpses until one or a very small number of piles remains. A somewhat more complicated phenomenon is brood sorting. Like cemetery formation, this behavior is widespread in ants, but it has only been studied intensively in *Leptothorax unifasciatus*. Workers of this species sort the larvae generally according to size and developmental progress. Eggs and prelarvae are clustered in the center of the area, medium larvae in the middle, and large larvae and prepupae around the perimeter. This clustering occurs by allocating different amounts of space to each type of larva. Eggs, for instance, are deposited with the minimal amount of individual space separating it from neighboring eggs, resulting in a central, tightly-knit cluster. A possible advantage to this type of spatial sorting is that items with similar needs are located together, facilitating efficient care of the developing ants.

Subsection 3.3.3: Nest Building and Assembly

Bonabeau asserts that, among all of the activities of social insects, nest building is the most "spectacular" [68]. He makes such a claim based on the great difference between individual and collective ability. Clearly, the ability of a single termite is dwarfed by the relative enormity and structural complexity of a termite mound. The impressive ability of social insects to perform assembly is most clearly demonstrated in termites, wasps, and bees. Each case is detailed further in the following paragraphs.

Mound-building behavior has long been studied, due in part to the awe-inspiring architecture of a large termite mound. After all, it was the study of termite behavior that inspired Grasse's creation of the term stigmergy. Understandably, nest building in termites and other insects is the clearest example of stigmergic behavior in all self-organized systems [68]. Camazine et al. [70] describe the mound of African termites as "air-conditioned skyscrapers immensely larger and arguably more sophisticated than the vast majority of human buildings" (p. 377). Refer to [70] for a thorough description of the intricate internal structure of a termite mound. If these termites and their mounds were scaled up to human size, the biggest would be "about a mile high…and five miles in diameter" [102]. While the complete group of processes by which termites build these mounds is beyond the scope of this work, the building can be reduced to two fundamental activities: digging and building. Combinations and variations in these activities result in mounds as high as six meters. The basic script that enables coordinated digging and building in termites is stigmergic. A simplified model of termite digging is described in [8]. The model suggests that digging progresses through positive feedback and stigmergy in the following manner. A termite moves randomly about the nest, with some probability at each time step of extracting a soil particle. When the particle is extracted, a pheromone is deposited that increases the probability of digging for future builders (Figure 4).

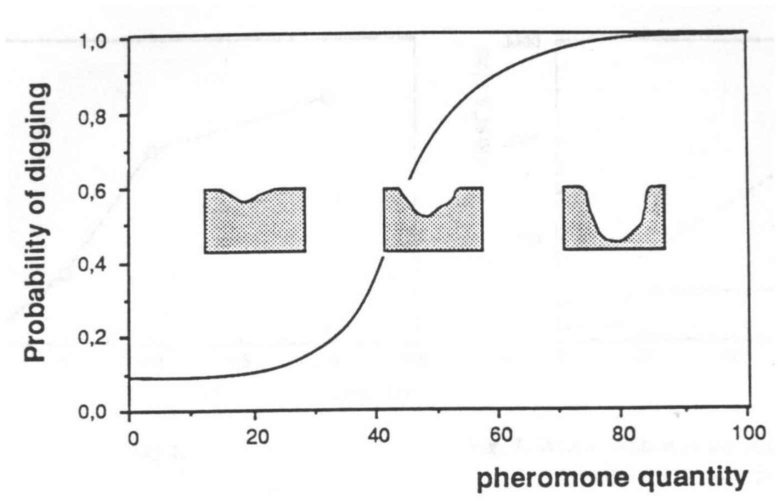


Figure 4. Stigmergy in termite digging (taken from [8], p. 125)

Pillar construction follows a very similar pattern, as described in [69]. The main difference in pillar construction is the existence of two behavioral phases. The first phase, the non-coordinated phase, is characterized by a random deposition of soil pellets. At some point, one of the deposits (piles) reaches a critical size, initiating the coordinated phase, during which this particular pile grows rapidly in size. Again, one observes the mechanisms of autocatalysis and amplification of random fluctuations. Michael Resnick programmed a simple computer simulation of this building behavior that reproduced some of the complexity in termite pillar construction. His observations of the non-coordinated phase produced an interesting musing on the building behavior of termites:

> *As long as a pile exists, its size is a two-way street: it can either grow or shrink. But the* existence *of a pile is a one-way street: once it is gone, it is gone forever. Thus a pile is somewhat analogous to a species of creatures in the real world. As long as the species exists, the number of individuals in the species can go up or down. But once all of the individuals are gone, the species is extinct, gone forever. In these cases, zero is a "trapped state": once the number of creatures in a species (or the number of wood chips in a pile) goes to zero, in can never rebound.* [84], pp. 79-80

This reasoning led Resnick to understand the process by which the number of pillar nucleation sites steadily decreases in the non-coordinated phase.

Wasps, like termites, produce complex nests known as combs. Each nest is composed of a large number of cells arranged in parallel vertical tubes. Cells are added sequentially by individual wasps, who build a cell at one of a number of potential building sites [70]. Certain building locations are preferred due to geometric factors that tend to maintain symmetry and even weight distribution in the nest. Since the nests hang vertically from a single stalk, weight distribution becomes an important consideration. There are essentially three types of sites available for adding a cell, with each site distinguished by the number of adjacent walls available for attachment. The probability of building at a site with three adjacent open walls is far higher than the probabilities for sites with one or two open walls [68]. These probabilities generate stigmergic cooperation that maintains progress toward a stable nest.

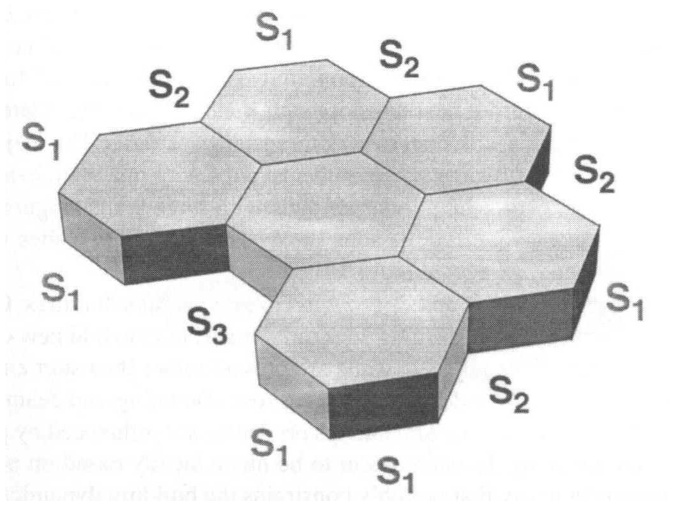


Figure 5. Potential building sites on a wasp nest (S_1 = one side, S_2 = two adjacent sides, S_3 = three adjacent sides, taken from [70], p.423)

Comb construction and organization in honey bees is another excellent example of self-organization in social insects. These spatial patterns are generated and maintained over the course of several individual life-spans, making it likely that individuals act based on local spatial and temporal information rather than memory [70]. A typical honey bee colony comprises approximately 25,000 adult workers, a single queen, as well as brood (eggs, larvae, and pupae) and accumulated food. The brood and food are stored in wax combs that are subdivided into about 100,000 cells [79]. These cells are organized according to a distinct pattern of three concentric rings—a central brood area, a surrounding rim of pollen, and a large peripheral region of honey. The biological significance of this structure in terms of its benefit to the colony arises from thermoregulation of the brood and efficient locations of pollen and honey for feeding. The conventional explanation of this global phenomenon is a blueprint hypothesis.

Proponents of this view suggest that individuals have an understanding that each cell or region of cells is meant for brood, pollen, or honey. The blueprint could be based on a temperature gradient from the center of the comb to the periphery, or it could be an innate function of instinct. This hypothesis is not supported by experimental observations, such as the frequent deposition of pollen and honey outside of their appropriate positions [79]. Camazine offers the main alternative view, which is based on self-organization theory. The self-organization hypothesis, which accurately reproduced the brood organization in simulation, describes the following process of pattern generation [79]:

(1) The central brood area develops from the queen's attempts to lay eggs near one another.

(2) The brood area is continually freed of honey and pollen to allow egg-laying, which enhances the compactness of the center.

(3) Honey and pollen are initially both present on the periphery, but emptied pollen cells near the outer edge are more likely to be filled with honey due to the much higher collection rate of honey.

(4) The only place remaining for new pollen to be deposited is near the outer fringe of the brood. In this narrow band, the removal of pollen and honey is very rapid due to the proximity to the brood.

Again, one observes that positive and negative feedback and stochastic elements generate complex spatial patterns. Also, note that this complexity occurs only through stigmergy, the most elegant mechanism of self-organized systems.

Subsection 3.3.4: Cooperative Transport

Ants are well-known for their extremely high weight-to-carrying capacity ratio. An adult ant of the species *Pheidologeton diversus*, for example, is able to carry prey as heavy as 10 times its own body weight. At approximately the same velocity, a group of 100 of these ants carried an earthworm weighing 5000 times the weight of a single worker [68]. Working together, each ant managed a burden 50 times its own weight, representing a five-fold improvement in task efficiency. A wide variety of ant species demonstrate similar cooperative transport behavior [62, 100, 103]. When an item of food is too large for a single forager to carry back to the nest, ant colonies have two options. In both cases, the solitary forager first returns to the nest to recruit nest mates. Depending on a number of factors, the recruits either cut the prey into smaller pieces or assist in cooperative transport. There are two interesting elements of this behavior. The first relates to the ability of an ant colony to automatically switch from solitary to group transport, which represents an example of a bifurcation in system behavior. The second concerns the actual mechanisms of coordination employed by the ants cooperating in transport. Studies in myrmecology have revealed that cooperative transport in ants is produced solely through stigmergy (see [68] and references therein). In other words, the ants "communicate" only through the object being carried. Each individual worker senses the forces applied to itself by the object that result from the summed forces of the other workers. This allows cooperative transport to occur without direct communication between individuals. Stigmergic cooperation in this task results in a prolonged period of organization early in the carrying process. Initially, each worker ant involved in the task repositions itself several times around the item. At some point, which is unpredictable in experimentation, cooperative behavior emerges and the ants are able to carry the object. Presumably, the applied forces of the ants become balanced during this initial alignment phase.

Section 3.4: Birds, Fish, and Mammals

Social insects are well known for their self-organized communities. However, the rest of the animal kingdom possesses demonstrations no less convincing than those of ants and bees. The primary difference in higher animals is cognitive ability; animals with some reasoning ability sometimes intentionally create complex system behavior. These systems can be mistaken for self-organized systems. Although this complication in identifying self-organized systems in high animals does exist, a multitude of examples remain clear. Camazine et al. [70] describe the array of examples outside of the insect world:

> *Noisy flocks of a thousand starlings burst into flight at the approach of a barking dog and maneuver gracefully around tall city buildings. At Carlsbad Cavern millions of Mexican free-tailed bats stream from the cave each evening to begin their nightly feeding foray. A herd of several hundred thousand wildebeest moves fluidly along the Serengeti plains of Africa. In the sea schools of millions of Atlantic cod swim together in tight formation as they migrate along the cold, deep waters off the Newfoundland coast.* (p. 167)

Mataric discusses the appearance of coordinated homing behavior in rats, pigeons, and salmon [61]. Cooperative hunting has also been studied in packs of wolves, tuna, and other predators [40, 45, 70, 104]. Trail-based foraging and recruitment behavior has been observed in such esoteric species as naked mole-rats [104]. A number of examples are provided below in Table 4.

Table 4. Examples of self-organization in nature (adapted from [70], pp. 170-171)

Organism	Scientific Name	Self-organized Process
Fish	many species	Coordinated movements in a school
Canada geese	*Branta Canadensis*	Chevron-shaped flight formation
Wildebeest	*Connochaetes taurinus*	Coordinated movements in a herd
Spiny lobster	*Panulirus argus*	Mass migration in single file
Locusts	*Schistocerca gregaria*	Coordinated movements in a swarm
Slime molds	*Dictyostelium discoideum*	Cohesive movements in aggregation

Easily the most popular example of SO in animals is flocking. Whether in birds or fish (schooling), flocking is a commonly studied behavior in biological systems [46, 61, 70]. Mataric identifies flocking as "…a ubiquitous form of structured group movement that minimizes interference, protects individuals, and enables efficient information exchange" [61], p. 19. Predator evasion is perhaps the most impressive property of flocks or schools. The two major methods of evasion are known as the Trafalgar Effect and flash expansion (see Figure 6). The Trafalgar Effect is "the rapid transfer of information throughout the school that enables the entire group to execute swift, evasive maneuvers at the approach of predators" [70], p. 172. Schools of fish, for example, are known for their incredible ability to spontaneously change direction despite an apparent lack of communication between individuals. Studies have shown that fish operate mainly based on the movements of their nearest neighbors. In this way, the reaction to a predator moves in a wave of propagation through the school at a rate much faster than the approach of the predator [70].

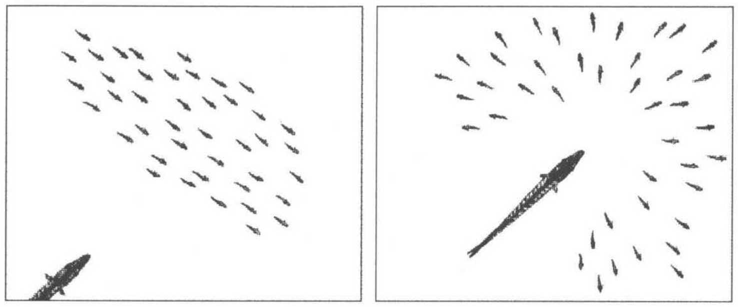


Figure 6. Flash expansion in a school of fish (taken from [70], p. 172)

This is just one example of a plethora of impressive examples of SO outside of the insect realm. Natural examples of swarm intelligence abound, and it remains to be seen what can be gained by deeper understanding of these systems.

Section 3.5: Misconceptions of Self-Organization

Due to the novelty of the field of self-organization and swarm intelligence, certain misconceptions are bound to arise. Fundamental misunderstandings generate these askew conceptions of self-organization and related ideas.

Misconception #1: *Self-organization fosters a falsely simplified view of nature.* The study of biological systems has proven the immense complexity of even the smallest-scale systems. In the case of self-organized systems, however, it is often possible to explain this manifested complexity in terms of a small set of surprisingly simple mechanisms. In these cases, the complexity may be generated by a simple response to a complex environment, rather than an inherently complex system. It is also important to note that models of self-organization only try to capture the essence of a system, not every minute detail of its operation.

Misconception #2: *Emergence is a mystical notion without scientific basis*. Due to the abrupt and seemingly spontaneous appearance of emergent patterns or behaviors, they are often considered beyond the scope of scientific inquiry. As discussed above, the mechanisms behind many emergent properties have been uncovered. In self-organization, as in many other fields of scientific study, observations simply defy *intuitive* understanding. This tendency does nothing to reduce their amenability to scientific investigation.

Misconception #3: *Individual agents intend to produce complex global behavior*. While this idea is often mentioned with respect to higher animals, such as herding mammals or flocking birds, it is often hypothesized about social insects. It is clearly unreasonable when applied to microscopic organisms. In the case of social insects, it is difficult to argue convincingly in favor of such a concept. Insects such as ants and bees have such limited cognitive abilities that even the simplest types of planning or processing are unlikely. Some higher animals with significant cognitive abilities could have some notion of the overall system behavior, but studies especially on flocking and schooling dispel such notions (see [70] and references therein). The fact that mathematical models and simulations closely reproduce self-organized biological behavior indicates that individuals need little to no understanding of group strategy.

Chapter 4: Foraging Theory and System Modeling

Section 4.1: Individual Agent Dynamics

Modeling the dynamics of the individual robots is not stressed in this work, due to the behavior-based control architecture. Such a control scheme places no emphasis on the implementation of traditional control theory to achieve optimal path planning. As in the world of social insects, the motion of an individual includes a significant stochastic component, and path trajectories in time and space are rarely smooth. Therefore, for the sake of completeness, only a brief development is provided of individual robot dynamics.

As described in Section 2.6, the choice of the drive system for a mobile robot determines the overall kinematics of locomotion. From a modeling perspective, the ideal case is a holonomic mobile robot, with rotation and translation uncoupled [38]. Unfortunately, such drive systems are highly complex and rarely implemented in robotics research applications. Yamaguchi presents pioneering work in his development of "…a distributed smooth time-varying feedback control law for coordinating motions of multiple nonholonomic mobile robots…" [38], p. 2984. This type of advancement in control theory reflects the infrequent use of holonomic mobile robots. The most common type of robot used in foraging research applications is the differential drive [24, 25, 41, 57, 60, 65, 73, 105], for which kinematics are discussed.

A wheeled mobile robot with a differential drive is modeled as a planar rigid body moving in a horizontal plane. The rigid body is connected via axles to the wheels on which it rolls. Driven independently, the two drive wheels are capable of producing both rotation and translation. The position of a robot, r_i, is given by (x_i, y_i), and its orientation by θ_i, in the static Cartesian coordinate system, Σ_0. The rotation angles of the two drive wheels are denoted ϕ_{1i} and ϕ_{2i}. The radius of each wheel, R, and the width between them, W, complete the preliminary definitions. The velocity and angular velocity of orientation of r_i are given as:

$$\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{pmatrix} = \begin{pmatrix} \frac{R}{2}\cos\theta_i(\dot{\phi}_{1i} + \dot{\phi}_{2i}) \\ \frac{R}{2}\sin\theta_i(\dot{\phi}_{1i} + \dot{\phi}_{2i}) \\ \frac{R}{2W}(\dot{\phi}_{1i} + \dot{\phi}_{2i}) \end{pmatrix} \tag{4-1}$$

A local coordinate system, Σ_i, is located at the midpoint of the two drive wheels. Refer to Figure 7 for a diagram of the robot with dimensions, vectors, and coordinate systems.

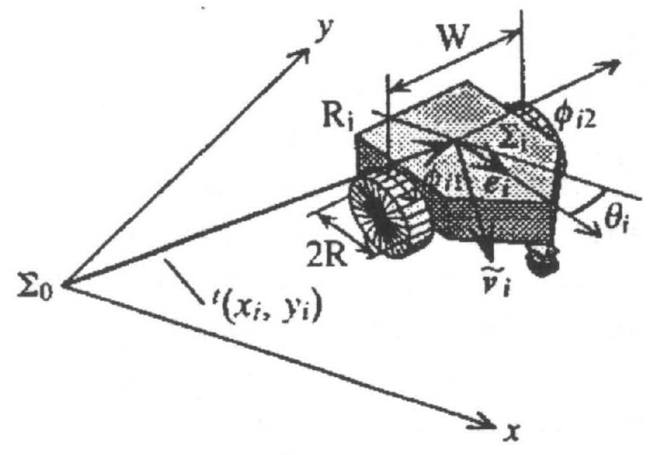


Figure 7. Vector and coordinate system definitions (taken from [38], p. 2987)

The unit vector, e_i, is defined on an axis of this local coordinate system. The axis specifies the orientation of r_i, i.e. θ_i, in the static coordinate system. Let v_i be the velocity vector of r_i. Expressed in the static coordinate system, Σ_0,

$$e_i = (\cos\theta_i, \sin\theta_i) \qquad \textit{and} \qquad v_i = (\dot{x}_i, \dot{y}_i) = \frac{R}{2}(\dot{\phi}_{1i} + \dot{\phi}_{2i})e_i \qquad \text{(4-2) and (4-3)}$$

It is useful to know the angular velocities of each wheel required to achieve a desired translational or angular velocity. In order to rotate at a general time-varying angular velocity, the difference in wheel velocities is given by:

$$\dot{\phi}_{1i} - \dot{\phi}_{2i} = \frac{2W}{R}\dot{\theta}_i(t) \tag{4-4}$$

For rotation about the midpoint of the rear axles (the origin of Σ_i), the required velocities are:

$$\dot{\phi}_{1i} = \dot{\phi}_{2i} = \frac{W}{R}\dot{\theta}_i(t) \tag{4-5}$$

Assuming translation in a straight path (either forward or reverse), the appropriate wheel velocities for translation are:

$$\dot{\phi}_{1i} = \dot{\phi}_{2i} = \frac{|v_i(t)|}{R} \tag{4-6}$$

These simple kinematic relationships determine the required actuation for controlling each individual robot. The issue of traveling in a curved path was not addressed, due to the fact that many robotic applications completely decouple rotation and translation. The experimentation found in Chapter 6 also does not require the ability to travel along a curved path. Instead, navigation is accomplished by alternating between straight forward or reverse translation and rotation.

Section 4.2: Foraging Theory and Task Efficiency

Before discussing the process of modeling self-organized systems, it is necessary to discuss the issue of task efficiency from the perspective of foraging theory. Task efficiency is an important ethological system performance metric with obvious importance to robotic systems as well. Maximization of task efficiency is, of course, desired in the design of a self-organized robotic foraging system. This requires metrics for measuring task efficiency. These metrics, if standardized, are also useful for comparing task efficiency to that found in biological systems, as well as between experiments conducted by different groups.

Foraging theorists identify two major methods of achieving efficient food source exploitation: time minimization and energy maximization [62, 94]. These two methods of optimizing foraging processes apply not only to social insects, but to the foraging practices of most animal life. The vast majority of foraging species are time minimizers, who tend to "…have a fixed quota of energy required for body maintenance and reproduction" [62], p. 388. Examples of time minimizers include animals such as birds, fish, and foraging mammals (e.g., deer). In these systems, the goal is to obtain the necessary energy quota in a minimum time (with minimal energy expenditure). In most microorganisms and insect societies, energy maximization is the goal. These communities are capable of utilizing a continuous influx of energy from the environment. In an ant colony, for example, excess energy corresponds to colony growth, which increases the overall survival probability of the colony. As a result, ant foraging seeks the maximum net yield of energy.

The most important methods of calculating and modeling theoretical task efficiency in ant foraging are concerned with energy maximization. Oster and Wilson, in their classic work on the ecology of social insects, describe several models of task efficiency for recruitment-based foraging [94]. These models either focus on the activities of individual workers or global parameters of the colony. In both cases, reward and cost functions are calculated based on the time and energy gains and expenditures associated with specific foraging activities. These functions are combined to calculate the net energy profit rate for an individual or group. Efficiency is increased to optimality by maximizing the net energy profit rate.

Robotic implementation of cooperative foraging also requires some consideration of task efficiency. In most experimental setups, some type of foraging task is defined. Completion of the task must be analyzed in some way with respect to efficiency. Unfortunately, the models of task efficiency developed by ethological foraging theory translate poorly into such scenarios. The main reason is that robotic foraging tasks are short-term processes, as opposed to the continuous nature of foraging in the societies of social insects. A colony of ants is never finished with the task of foraging. For this reason, robotic foraging tasks are generally viewed in terms of time minimization. Like most vertebrate animals, the foraging tasks of robots involve collection of a limited energy quota. Of course, in robotic experimentation, the "energy" takes the form of small targets collected in an enclosed foraging field. Robotics literature reflects this notion of time minimization as a measure of task efficiency. The most common metrics by which efficiency is calculated are the total time required for completion and the number of targets collected per unit time [7, 12, 58, 60]. Currently, it is difficult to use these measures of efficiency to compare the performance demonstrated in separate experiments. The use of time as an efficiency measure also makes it difficult to generalize results or compare to the efficiency of social insects. This disparity in measures of task efficiency between ethological foraging theory and robotic implementation highlights the need for standardized performance metrics in the swarm intelligence community.

Section 4.3: Modeling Self-Organization

The development of mathematical modeling in the field of self-organization and swarm intelligence has occurred almost exclusively from an ethological perspective. Each of the major classes of self-organization models arose from attempts to model the behavior of biological systems. Only rarely have researchers from the robotics community sought to develop independent models. Most often, robotics research borrows or adapts models with biological origins. This pattern seems appropriate--and perhaps the development of independent models is misguided--given the efficiency of biological systems such as groups of social insects. In any case, rigorous models of self-organization are essential for continued progress toward real-world robotic applications of swarm intelligence. Bonabeau, Dorigo, and Theraulaz convincingly argue in favor of using biological models of SO:

> *…very few applications of swarm intelligence have been developed. One of the main reasons for this relative lack of success resides in the fact that swarm-intelligent systems are hard to "program," because the paths to problem solving are not predefined but emergent in these systems and result from interactions among individuals…. Therefore, using a swarm-intelligent system to solve a problem requires a thorough knowledge not only of what individual behaviors must be implemented but also of what interactions are needed to produce such or such global behavior. …[A] reasonable path consists of studying how social insects collectively perform some specific tasks, modeling their behavior, and using the model as a basis upon which artificial variations can be developed,*

either by tuning the model parameters beyond the biologically relevant range or by adding nonbiological features to the model. [68], p. 7

This is the type of reasoning that has led most robotic researchers to design swarm-intelligent systems based on adapted biological models of self-organization.

The development of fairly simple mathematical models based on SO requires certain philosophical presuppositions. First of all, one must accept that it is possible, at some level of description, to explain complex collective behavior in terms of simple interacting entities. From a certain perspective, a social insect is a complex creature capable of processing many sensory inputs and making decisions on the basis of a large amount of information. From a modeling perspective, however, one assumes that this complexity can be reduced significantly while maintaining complex system behavior. Behavior-based control architectures in mobile robotics make the same fundamental assumption. In the development of SO models, researchers have therefore taken the approach of using very simple models unless further complexity is found necessary to reproduce the observed biological behavior on which the model was based [59]. This process is illustrated in Figure 8.

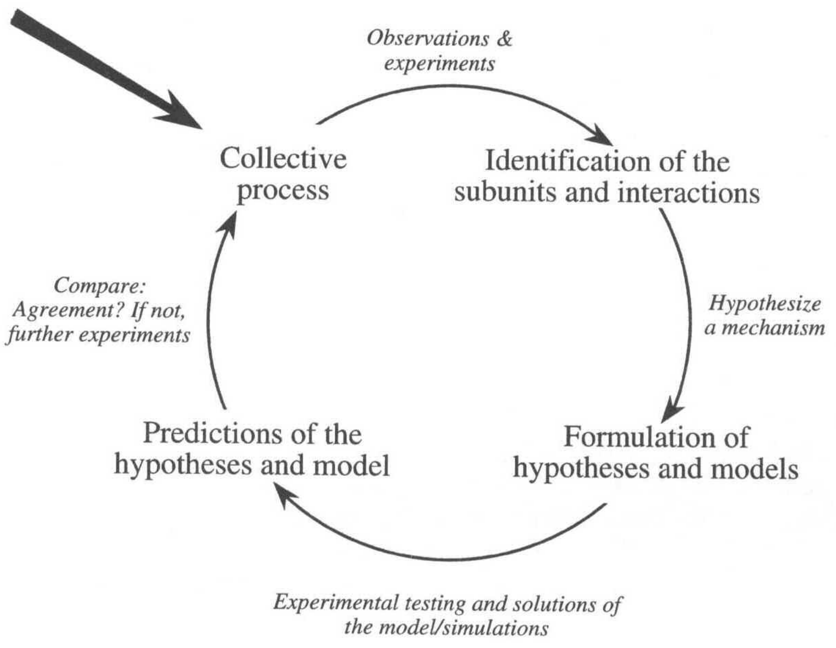

Observations & experiments
Collective process
Identification of the subunits and interactions
Hypothesize a mechanism
Formulation of hypotheses and models
Experimental testing and solutions of the model/simulations
Predictions of the hypotheses and model
Compare: Agreement? If not, further experiments

Figure 8. Research cycle for self-organized systems (taken from [70], p. 71)

Simplifying assumptions are only removed from the model if necessary. This process has proven successful in generating a number of surprisingly simple models of self-organized systems. Some of these models are presented below in Section 4.4.

Models of SO can be divided into three main classes: differential equation, probabilistic (or stochastic), and cellular automaton models. While not every model fits neatly into one of these categories, the classification accounts for the vast majority. Each class of models is described below.

The most common type of model for self-organized systems is differential equation models. This approach is most amenable to thorough analysis, as a set of differential equations allows one to investigate equilibrium conditions, bifurcation phenomena, etc. It is desirable to obtain a precise quantitative description of the behavior of a system over time by solving the system of equations. Although many such systems do not have closed-form solutions, numerical techniques are used to produce the system trajectories. This type of model requires that certain assumptions be made in models of discrete processes. Differential equation models assume that individual agents are infinitely divisible individuals behaving deterministically, which is valid for systems with large populations. However, it can be very difficult--due to continuity assumptions and other factors--to formulate a set of differential equations describing the behavior of certain biological systems.

Probabilistic models are commonly employed when differential equations are inappropriate. As opposed to the previous class, these models tend to use a discrete approach that models the behavior of each individual in the group. Simple behavioral rules based on probability are used to generate a few governing equations. Monte Carlo simulations are often conducted to produce similar results as those obtained from an actual experiment. The stochastic nature of these models yields qualitatively different results than those obtained from differential equations. Rigorous analysis is more difficult, but the model development process is generally simpler. Probabilistic modeling also tends to be very computationally intensive, sometimes requiring considerable computer processing resources.

The final class of models, cellular automaton (CA) models, was probably the first modeling technique used to investigate self-organization [4]. This type of modeling "…simulates groups of biological components by arranging components as points in a lattice or grid and allowing the components to interact according to simple rules" [70], p. 76. Usually performed within a two-dimensional lattice, CA simulations involve many cells characterized by a location and state. At discrete time steps, each cell updates its state based on its current state and the state of neighboring cells. While CA models exhibit highly complex behavior, the cell interactions are governed by only a few simple rules. For this reason, CA models are excellent for modeling and simulating SO. Conway's Game of Life is one famous example of such a CA model. Cellular automaton models have the advantage of providing impressive visual results, but, like Monte Carlo simulation, can be computationally intensive. Depending on the software implementation, CA can also require considerable programming expertise.

Section 4.4: Foraging Models

In the following three subsections, models are reviewed from the literature for foraging systems. The models are divided into the three main classes mentioned above: differential equations, probabilistic, and CA models. Each section presents the models most relevant to the computer simulation and experimental work in Chapters 5 and 6. Although several may be discussed, one primary model is featured and developed in each section. It is important to note that each of the models presented differs significantly from the simulation and experimental work described in later chapters. Unfortunately, current foraging models make significant simplifying assumptions, making it inappropriate to apply them directly to complete foraging systems based on mas recruitment. Therefore, the models are used to generate only qualitative predictions regarding system performance. The predictions are presented in Section 4.5.

Subsection 4.4.1: Differential Equations

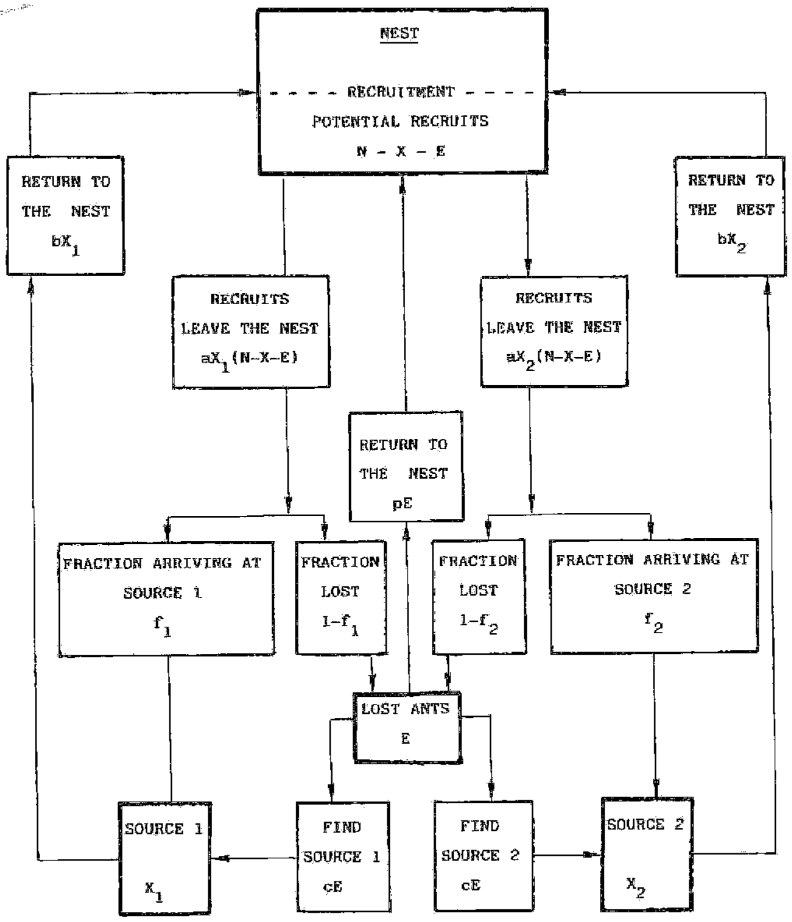
SYSTEM MODEL

Foraging models consisting of one or more differential equations provide the most complete mathematical descriptions currently available for recruitment-based foraging. The systems of equations used in these models are sometimes linear and often simple enough to investigate analytically. Stability analysis, bifurcation phenomena, and analysis of steady-state solutions are common procedures performed while studying these models.

One particular subset of differential equation models merits special mention before focusing on the mainstream techniques. A handful of researchers have incorporated pseudo-stochastic elements into their differential equations models (e.g., [43, 83]). These models can be termed *state transition* models, owing to the formulation of the set of differential equations. A number of system states are defined, along with relationships governing transitions between states. The stochastic state transitions, if any, are cleverly included into the model by creating customized probability parameters. Pasteels, Deneubourg, and Goss present the clearest example of a state transition model [83]. Their model describes trail recruitment by N ants to two identical sources. The model includes three system states:

- Number of ants at food source 1, X_1
- Number of ants at food source 2, X_2
- Number of lost ants, E

The lost ants are those that strayed from a pheromone trail in the following process. All remaining ants, numbering $N - (X_1 + X_2 + E)$, exist at the nest as potential recruits. A simple block diagram (see Figure 9) representing the system dynamics leads to a system of three differential equations.



$$
\begin{aligned}
\dot{X}_1 &= aX_1f_1(N-X-E) - bX_1 + cE\\
\dot{X}_2 &= aX_2f_2(N-X-E) - bX_2 + cE\\
\dot{E} &= a(X_1(1-f_1) + X_2(1-f_2))(N-X-E) - pE - 2cE\\
f_i &= X_i/(g + X_i)
\end{aligned}
$$

Figure 9. Simplified block diagram of foraging system and resulting system of differential equations (taken from [83], p. 166)

The authors present both steady-state analysis and solution trajectories. Results demonstrate bifurcation phenomena and multistability between two food sources. As the number of total foragers, N, is increased, the system passes through a bifurcation point and switches from symmetrical to asymmetrical exploitation of the two sources.

Sugaware and Watanabe develop a similar model for a single-source robotic foraging system [43]. Notably, the robots employ beacon-based communication rather than laying trials. The model contains a system of five differential equations, each corresponding to the number of robots in each of five possible behavioral states. As a result, this model meshes well with behavior-based control. The model is used to validate simulation results and show the dependence of task efficiency on group size and interaction duration.

A number of more typical models appear in the literature. These other models focus on state variables such as trail length [81, 97], number of feeding workers [71], or outgoing ant flux from the nest [70, 85]. Edelstein-Keshet et al. present two models based on trail length, but both models feature continuous trail-laying rather than mass recruitment [81, 97]. The authors analyze the models with respect to global adaptability in response to external conditions. Deneubourg, Pasteels, and Verhaeghe, develop another model of similar structure and purpose to illustrate "…that the degree of randomness [can] be optimally 'tuned' to particular ecological conditions, such as food quantity and distribution" [71], pp. 259.

A fairly recent (1999) foraging model based on differential equations is, by far, the most rigorous theoretical representation of a recruitment system. Nicolis and Deneubourg present "a model of food recruitment by social insects accounting for the competition between trails in the presence of an arbitrary number of sources…" [85], p. 575. This publication represents a significant advancement in foraging modeling, as it tackles the problem of inter-source competition between more than three sources. Prior to this model, such problems remained unsolved. The model is rigorous in an analytical sense, yielding the full bifurcation diagram of steady-state solutions. It is also relatively simple in formulation, which is not surprising in a self-organized system.

The model takes a macroscopic approach to the mass recruitment system, using pheromone concentration as the principal variable. It is mainly devoted to the choice of orientation among more than one trail upon leaving the nest. Nicolis and Deneubourg make important initial assumptions regarding the recruitment system, as listed below:

- Pheromone trails are always followed error-free from nest to food source.
- Ants are either within the nest or following a trail; wandering does not occur.
- Direct interactions between individuals are ignored; only stigmergy is possible.
- The outgoing flux of ants from the nest is a constant.

These simplifying assumptions render the model unable to manifest certain characteristics of self-organization. For example, error-free line following makes the accidental discovery of a new food source impossible. In fact, the assumptions listed above constrain the scenario to one in which all sources have already been discovered. Another limitation of the model is the inability to investigate behavioral dependence on the number of foragers. Despite these limitations, as will become apparent, the model illuminates a number of significant system characteristics.

The development that follows closely resembles that presented originally in [85], with only minor variations and/or simplifications. Refer to [85] for the full model derivation. Let c_i be the pheromone concentration on trail $i = 1,2,\ldots,s$. The time derivative of pheromone concentration consists of two parts: (1) a positive term from trail deposition and (2) a negative term from trail evaporation. The resulting equation for the time rate of change of pheromone concentration on trail i is:

$$\frac{dc_i}{dt} = \phi \sigma_i F_i\left(\left\{c_j\right\}\right) - v_i c_i , \qquad i = 1, \ldots, s \tag{4-7}$$

where σ_i is the quantity of pheromone on trail *i*, v_i is the corresponding evaporation rate, and $F_i(\{c_i\})$ is a function describing the relative attractiveness of trail *i* compared to all other trails. The form of F_i is a choice function commonly used in probabilistic models (e.g., [96]):

$$F_i = \frac{(k + c_i)^n}{\sum_{j=1}^{s} (k + c_j)^n} , \tag{4-8}$$

where *s* is the total number of sources; *k*, a concentration threshold beyond which pheromone begins to be effective; and *l*, the sensitivity of the process to c_i. Equation 4-8 accounts for three types of dynamic feedback:

1. Positive, nonlinear feedback of trail i onto itself through F_i.
2. Negative, linear feedback of trail *i* onto itself through pheromone evaporation.
3. Negative, nonlinear feedback of trail $j \neq i$ onto trail *i* from competition.

It is convenient to introduce the following non-dimensionalization of the key parameters:

$$C_i = \frac{c_i}{k}, \quad q_i = \frac{\sigma_i}{k}, \quad \Phi = \frac{\phi}{v} \tag{4-9}$$

Assuming that evaporation is constant on all trails, allv_i can be set to a common *v*. Then, scaling time allows *v* to be set to unity. Introducing the non-dimensionalization results in the reduced system:

$$\frac{dC_i}{dt} = \Phi q_i \frac{(1 + C_i)^l}{\sum_{j=1}^{s} (1 + C_j)^l} - C_i \tag{4-10}$$

The parameter *l* is set at$l = 2$ to reflect experimental data with several ant species, including *Lasius niger* [99]. With these simplifications, the solutions of Equation 4-10 depend entirely on Φq and *s*. For the trivial case of one food source, j = 1, there exists one, stable steady-state solution given by

$$C = \Phi q = \frac{\phi\sigma}{vk} \tag{4-11}$$

The trail concentration reaches a steady value proportional to the output flux and deposition rate, and inversely proportional to the evaporation rate and concentration threshold. These relationships, for the trivial case, make intuitive sense. The steady-state solutions of Equation 4-10 for multiples sources are of particular interest, obtained by setting the time derivative equal to zero. Dividing the equations applied to trails *i* and *j*, one obtains:

$$\frac{q_i(1+C_i)^2}{q_j(1+C_j)^2} = \frac{C_i}{C_j} \qquad \text{and} \tag{4-12}$$

$$\sum_{j=1}^{s}(1+C_j)^2 = \Phi\, q_i\, \frac{(1+C_i)^2}{C_i} \tag{4-13}$$

The set of steady-state solutions to Equations 4-12 and 4-13 are denoted $C_{i,\,st}$. Stability analysis of the different solutions determines the state actually chosen by the system. Equation 4-10 is linearized with respect to perturbations by setting

$$C_i = C_{i,st} + \delta C_i \tag{4-14}$$

and, using Equations 4-12 and 4-13, rewriting Equation 4-10 as

$$\frac{d\delta C_i}{dt} = \sum_k A_{ik}\,\delta C_k \tag{4-15}$$

$$A_{ik} = -\frac{2C_i^2(1+C_k)}{\Phi q_i(1+C_i)^2} + \frac{C_i - 1}{1+C_i}\,\delta_{ik}^{kr} \tag{4-16}$$

The "*st*" subscript has been dropped, as it is understood that C_i are steady-state. Stability is determined by examining the roots of the characteristic equation,

$$\det\left|A_{ij} - \omega\delta_{ij}^{kr}\right| = 0 \tag{4-17}$$

with the stability condition Re $\omega_i < 0$ for all *i*. While this model can be extended to examine the case of non-identical sources (see [70, 85]), the assumption is made for this analysis that each source is identical, or $q_i = q$ for all *i*. This assumption is valid especially for application in robotic experimentation, which generally predetermines source similarity. The homogeneous solution of equation (4-15) is given by

$$C_i = \frac{\Phi q}{s} = C\,, \tag{4-18}$$

representing equal exploitation of all sources. For example, in the case of equal exploitation of two identical sources, the trail concentrations are

$$C_1 = C_2 = \frac{\Phi q}{2} \tag{4-19}$$

In order to check the stability of this solution, we examine the elements of the Jacobian matrix, which can only take two values:

$$\begin{aligned} A_{ii} &= a = \frac{-2\Phi q}{s(\Phi q + s)} + \frac{\Phi q - s}{\Phi q + s}, \\ A_{ij} &= b = \frac{-2\Phi q}{s(\Phi q + s)} \end{aligned} \tag{4-19}$$

The characteristic equation takes the form,

$$(\omega - (a + (s-1)b))(\omega - (a-b))^{s-1} = 0\,, \tag{4-20}$$

yielding the following solutions:

$$\omega_1 = a + (s-1)b = -1 < 0 \quad \text{and} \tag{4-21}$$

$$\omega_{2\ldots s} = a - b = \frac{\Phi q - s}{s + \Phi q} \qquad (4\text{-}22)$$

The first solution, ω_1, is always stable, but the group of s-1 solutions loses stability for $\Phi q > s$. This result shows that the homogeneous solution may lose stability, causing the system to assume another stationary state. The only such solutions are semi-inhomogeneous, in which j trails having concentration C_1 are exploited in a different manner than the other s-j trail with concentration C_2. These concentrations are given by

$$C_1^{\pm} = \frac{\Phi q}{2j} \pm \frac{1}{2}\sqrt{\left(\frac{\Phi q}{j}\right)^2 - 4\left(\frac{s-j}{j}\right)} \qquad (4\text{-}23)$$

$$C_2^{\mp} = \frac{1}{C_1^{\pm}} = \frac{\Phi q - jC_1^{\pm}}{(s-j)}, \qquad j = 1, \ldots, s/2 \ \textit{for s even or} \ (s+1)/2 \ \textit{for s odd} \qquad (4\text{-}24)$$

where the + and – superscripts correspond to a trail more or less heavily marked, respectively. These solutions exist as long as

$$\Phi q \geq 2\sqrt{(s-j)j}\,. \qquad (4\text{-}25)$$

The stability is determined separately for the cases j = 1 and j > 1. The former case corresponds to differential exploitation of one trail as compared to all others; the latter to preferential exploitation of a group of trails compared to another group. For j = 1, the characteristic equation takes the form

$$(\omega - a_2 + c)^{s-2}(\omega^2 - (a_1 + a_2 + (s-2)c)\omega + a_1a_2 + a_1c(s-2) - (s-1)b_1b_2) = 0\,. \qquad (4\text{-}26)$$

Numerical evaluation of the roots reveals that the high concentration, C_1^+ (and the corresponding C_2^-), is always stable while the low concentration, C_1^- (and C_2^+), is always unstable. This is expected, based on experimental observations of collective decision-making in ants between trails [70, 71, 82, 99]. For j = 1 and s > 2, these solutions correspond to a limit point bifurcation. For s even and $j = s/2$, there is a pitchfork bifurcation of unstable branches emerging from $\Phi q = s$. Refer to the bifurcation diagram below for $s = 4$ (Figure 10). The homogeneous state loses stability at $\Phi q = 4$, the locus of a pitchfork bifurcation of two unstable branches (corresponding to $j = 2$). The limit points are located at $\Phi q = 3.5$, generating four semi-inhomogeneous solutions, two of which are stable. In the domain $3.5 < \Phi q < 4$, three simultaneous stable solutions exist, while two exist for $\Phi q > 4$. Note the multistability present in this system, one of the characteristic properties of self-organization.

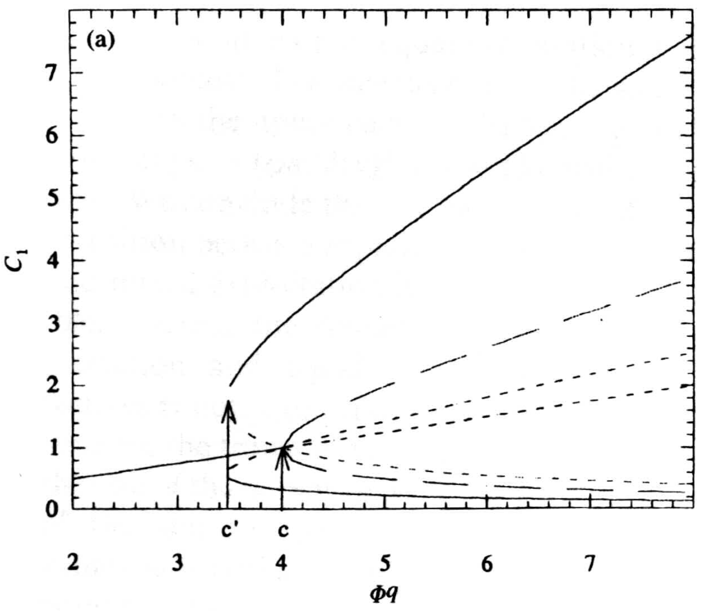


Figure 10. Bifurcation diagram for s = 4 (taken from [85], p. 582)

For the case of $j > 1$, manipulation of the characteristic determinant reveals the following two j and $s - j$ roots:

$$\begin{aligned} \omega_1 &= a_1 - c_1 = \frac{C_1 - 1}{1 + C_1} \\ \omega_2 &= a_2 - c_2 = \frac{1 - C_1}{1 + C_1} \end{aligned} \tag{4-27}$$

Given that the stability conditions for these roots are incompatible, one concludes that at least one root is positive. Hence, the semi-inhomogeneous solutions with $j > 1$ are always unstable. The state diagram of the system is shown in Figure 11, with $\Phi = 10$. Note the three main regions of the graph, each corresponding to different stability conditions of the homogeneous and semi-inhomogeneous solutions. Multistability is represented in the middle region, in which a number of stable states are possible. Note that bifurcation phenomena are possible as environment factors cause the system to move from one region to another.

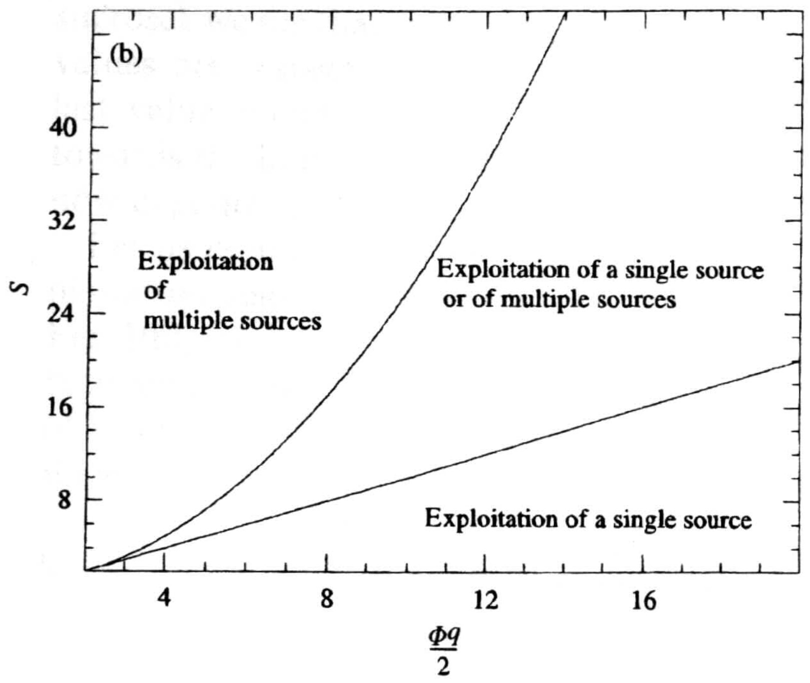


Figure 11. State diagram with exploitation modes (taken from [85], p. 582)

The critical values of s, denoted s_{c1} and s_{c2}, correspond to the number of food sources required to switch from one exploitation mode to another (for a given set of system parameters).

$$s_{c1} = \frac{\phi\sigma}{vk}, \qquad s_{c2} = \left(\frac{\phi\sigma}{2vk}\right)^2 + 1 \qquad \text{(4-28) and (4-29)}$$

The first critical number represents the transition point from exploitation of a single source to mixed exploitation. The second number, Equation 4-29, corresponds to transition from mixed to homogeneous exploitation of multiple sources. Note that these values depend on the colony size (related to ϕ), quality of sources (σ), and evaporation rate (v). These critical values explain the ability of an ant colony to adapt foraging behavior in response to global environment changes, such as addition of food sources. Due to the system dynamics presented above, colony-level behavioral plasticity is achieved without the need for individual awareness of the changes.

PURSUIT MODEL

While the other models in this section describe complete systems of central-place foraging, this particular model focuses on one specific element. As described in Subsection 3.3.1, some species of ants employ *group recruitment.* In this process, the recruiter leaves a trail to the food source while being followed by a number of recruited workers. The followers sometimes deposit trails of their own, reinforcing that of the leader. Ethologists have observed that ant trails become more direct as more followers reinforce the trail, eventually resulting in a straight line connecting the nest and food source. Bruckstein presents a mathematical analysis of this pursuit problem in [106], aiming to show "…that globally optimal solutions for navigation problems can be obtained as a result of myopic cooperation between simple agents or processors" (p. 62).

Bruckstein's analysis of the ant-inspired pursuit problem proceeds as follows. The "pioneer" ant, denoted A_0, follows an arbitrary path, P_0, at unit speed from home located at (0,0) to the food source, located at (L,0). The initial path is parametrized as:

$$P_0(t) = [x_0(t), y_0(t)], \tag{4-30}$$

where the parameter, time (equivalent to arc length), runs from 0 to $T_0 = L_0$, the length of P_0. At time $\tau_1 > 0$, the first follower, A_1, starts walking with its velocity vector always pointing toward A_0 (see Figure 12). The path of A_1 is given by:

$$P_1(t) = [x_1(t), y_1(t)], \tag{4-31}$$

where t runs from τ_1 to $T_1 = \tau_1 + L_1$, with L_1 being the length of P_1. At time $\tau_2 > \tau_1 + \delta$, A_2 begins following A_1, and so on. If ant A_{n+1} catches A_n, the two are joined on the path of A_n.

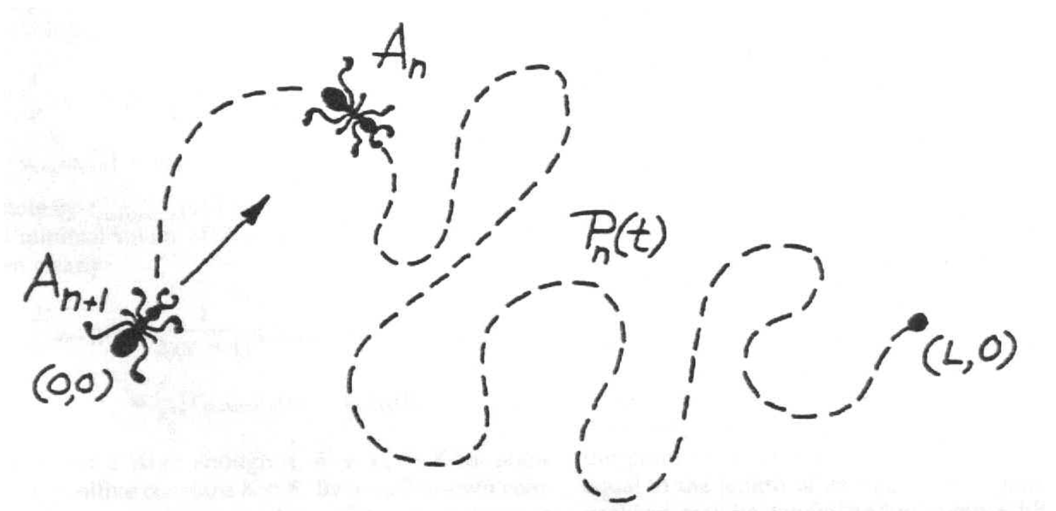


Figure 12. Diagram of theoretical pursuit problem (taken from [106], p. 60)

The rule of pursuit says that the velocity vector of A_{n+1} is always directed toward A_n, with unity magnitude. Therefore, the path of ant A_{n+1} is the solution of the differential equation:

$$\frac{d}{dt}P_{n+1}(t) = \frac{P_n(t) - P_{n+1}(t)}{|P_{n+1}(t) - P_n(t)|} \quad \text{with initial condition } P_{n+1}(\tau_{n+1}) = (0,0) \tag{4-32}$$

Bruckstein proceeds to provide a rigorous proof for the following theorem:

$\boxed{\textit{THEOREM: The sequence of pursuit paths, } P_n\textit{, converges to the straight line segment}}$

$\boxed{\textit{connecting (0,0) to (L,0).}}$

This proof of the limiting behavior of solutions to the governing differential equation, Equation 4-32, is an important result in self-organization. While reproduction of the proof is beyond the scope of this work, the result shows that a very simple rule of pursuit on the individual level generates self-organization on the group level.

Subsection 4.4.2: Probabilistic Models

Probabilistic modeling of biological self-organization is quite common, although it is less frequently employed for studies of complete foraging systems. The most prevalent usage of probabilistic modeling and Monte Carlo simulation relates to trail-following behavior. In particular, probabilistic models are traditionally used to examine the choice of a worker ant at a trail bifurcation point. These models are incapable of capturing the full behavioral complexity of a mass recruitment system, but they provide valuable insight into the mechanisms of self-organization in such systems. One popular example prevails in the swarm intelligence literature: the "binary bridge" or "binary choice" model [68-70, 96, 99]. The model was initially developed by Deneubourg et al. in 1990 in relation to a binary bridge experiment with the Argentine ant [96]. Subsequent investigations have borrowed the same model for foraging analysis of different ant species (e.g., [99]).

The binary bridge experiment was designed to study the transition in the Argentine ant "…from the uncoordinated state (the diffuse front) to the coordinated state (the exploratory trail)" [96], p. 159. The setup consists of a diamond-shaped bridge placed between the nest and the food source. In this way, the researchers were able to analyze the exploration traffic on two alternative routes. In [69], additional experimentation is presented with unequal bridge length. These experiments demonstrate collective decision-making in a swarm of foraging ants, as the ants regularly choose the shorter path.

The model formulation is quite simple, but it accomplishes the goal of reproducing the experimental results using a few simple probabilistic relationships. The model makes the following initial assumptions:

- The amount of pheromone on each branch is proportional to the number of ants that used the branch to cross.
- Pheromone evaporation does not occur.
- The probability of choosing either branch can be calculated using a simple general choice function.

The assumption regarding pheromone evaporation applied to the Argentine ant, but the assumption breaks down at large time scales. Therefore, the model is not valid steady-state analysis of long-term trajectories. According to these assumptions, let A_i and B_i be the numbers of ants that have used branches A and B, respectively, after *i* total ants have used the bridge. The probability that the (*i* + 1)th ant chooses branch A is:

$$P_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} \tag{4-33}$$

The complementary probability is given by $P_B = 1 – P_A$, due to the fact that one of the two bridges must be chosen. This equation quantifies the way in which a higher concentration on branch A corresponds to a greater probability of an ant choosing trail A.

The form of the equation was obtained from trail-following experiments in [83]. The parameter *n* determines the degree of nonlinearity of the choice function. Large *n* corresponds to sensitive dependence on differences in A_i and B_i. The parameter k quantifies the degree of attraction to an unmarked branch. A high value of *k* necessitates a greater amount of pheromone to make the choice nonrandom.

In addition to the choice function, Equation 4-33, it is necessary to define the dynamics of trail reinforcement. The dynamics follow directly from the first assumption of the model (see previous page):

$$A_{i+1} = \begin{cases} A_i + 1 & if \quad \delta \le P_A \\ A_i & if \quad \delta > P_A \end{cases} \tag{4-34}$$

$$B_{i+1} = \begin{cases} B_i + 1 & if \quad \delta > P_A \\ B_i & if \quad \delta \le P_A \end{cases} \tag{4-35}$$

where δ is a random variable uniformly distributed over [0,1]. These dynamics ensure that $A_i + B_i = i$, the number of discretized time steps (and ant passages). A simple Monte Carlo scheme can be used to analyze the model. Deneubourg et al. used a Monte Carlo simulation to compare the model's predictions to experimental results [96]. By tuning the parameters *n* and *k,* the model agrees closely with experimental results (Figure 13). The graph below shows the percentage of ants that followed the dominant trail as a function of the total number of ant passages on either trail. For low values of total ant passages, the percentage of preference is only slightly above 50%. But, as the total number of passages increases, over 90% of the ants have chosen the dominant branch during a given trial.

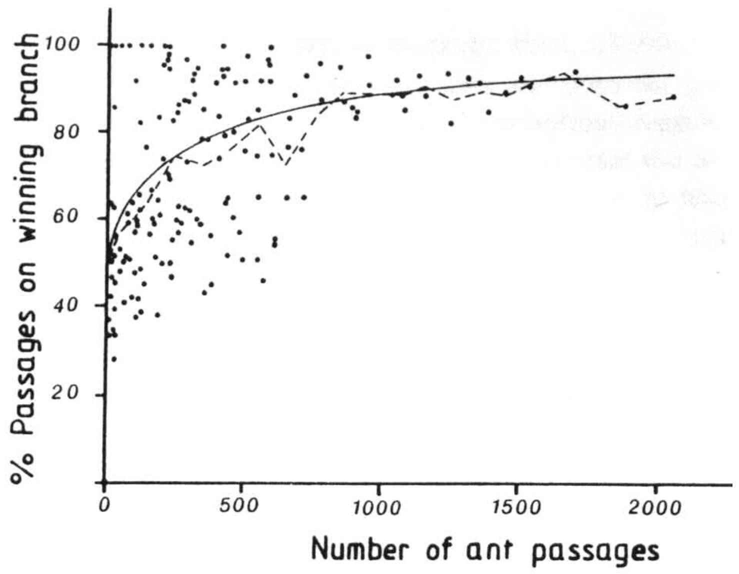


Figure 13. Theoretical model (solid curve) compared to experimental data points (taken from [96], p. 163)

This model clearly demonstrates the mechanism of positive feedback. Positive feedback amplifies initial fluctuations in the system, causing the colony to preferentially exploit one of the two equal alternatives. Multistability is clearly present, as exclusive exploitation of either source is a potential stable state. While [68] mentions the extension of this model to include branches of unequal length, only experimental results are reported.

Subsection 4.4.3: Cellular Automaton Models

Of the three main classes of foraging models, cellular automaton models are least applicable to central-place foraging via mass recruitment. Due to the limitations of using very simple rules to determine the state of each cell, it is difficult, if not impossible, to use CA to model a complete foraging system. Instead, CA is often employed to simulate trail networks (e.g., Figure 14). By far the most common application of CA to self-organization in ant foraging is modeling of the exploratory swarm raids of army ants [68, 70, 81, 95, 107]. Figure 14 shows the results of three CA simulations of swarm raid patterns of army ants. The trail patterns illustrate a central trunk trail trail leading away from the nest, with a delta-shaped front at the leading edge of the swarm. The three simulations, A, B, and C, correspond to various distributions of food within the environment. In scenario A, no food was present in the virtual environment; in B, food was distributed evenly in many clusters; in C, food was located in only a few large clusters.

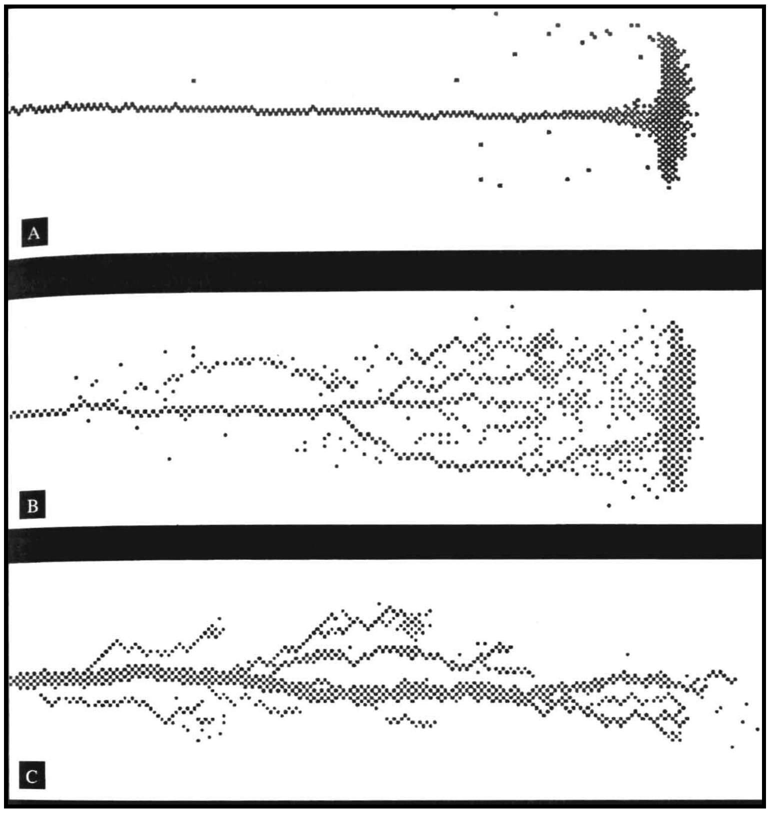


Figure 14. Results of CA simulation of swarm raids (taken from [70], p. 275)

This type of behavior is more amenable to CA modeling than mass recruitment-based foraging. The effort that comes closest to modeling mass recruitment is found in two related papers by Ermentrout, Watmough, and Edelstein-Keshet [108, 109]. The first of these papers, published in 1993, represents the definitive publication on using CA to model biological systems [108]. The paper presents a CA model of the formation of persistent foraging trails, showing "…that the formation of a dominant trail depends on the total ant density" [109], p. 357. The subsequent work uses the same model to investigate the initial formation of a trail by ants emerging from a central nest location. It is important to note from the outset that, in both investigations, the simulated "ants" continually deposit pheromones. This represents a significant departure from mass recruitment behavior, in which pheromones are only deposited on the trip home from a food source. However, these two papers represent the best example of CA modeling for a foraging system.

Ermentrout and Edelstein-Keshet identify two major classes of CA models—deterministic and "lattice gas"—that are employed simultaneously for their simulation [108]. The formation and decay of trails obey a deterministic model, while motion of the ants follows the "lattice gas" approach. As the name implies, the deterministic model changes the state of each cell based on deterministic rules incorporating the states of neighboring cells. The "lattice gas" method allows for partially random motion of *particles* within a spatial grid.

The basic CA model includes an algorithm for "ant" behavior within the two-dimensional grid and simple deterministic rules for the formation and decay of trails on the stationary cells. The "ant" behavior is given by the following rules:

1. The ants move at a fixed speed.
2. Each ant deposits pheromone at a constant rate. When following a trail, a larger constant rate ensues.

3. There is a constant probability (nonzero) that an ant following a trail will lose the trail.
4. There is a constant probability (nonzero) that an ant will cross a trail and not follow the trail.
5. When an ant comes to a fork, it has a higher probability of choosing the path with higher pheromone concentration.
6. While wandering, each ant has a probability per time step of turning a random angle.

The rules governing the trail formation and pheromone decay are:

1. The state variable for pheromone concentration increases when an ant passes.
2. The pheromone decays at a steady rate.

Given these simple rules, the automaton requires the following parameters for operation:

- grid size, m
- number of ants, N
- maximum and minimum pheromone concentrations, ϕ_{max} and ϕ_{min}
- amount of pheromone deposited while off trail, τ_{off}
- amount of pheromone deposited while on trail, τ_{on}
- probability of a wandering ant turning an angle of 45n°, B_n
- probability of losing a trail, P (fidelity α P^{-1})

All of the probabilities are per ant per time step. In [108], the grid has periodic (toroidal) boundary conditions, with m = 50. The continuous boundaries allow uninterrupted simulation for analysis of steady-states. In [109], a grid with "absorbing" boundary conditions was used to simulate an infinite domain, and m = 256.

The results of these complementary CA models are three-fold. First, [108] establishes that a critical ant density is required to establish a dominant trail. Second, the same investigation also identified an initial disordered phase prior to trail establishment. Figure 15 illustrates this early phase of the simulation, with each ant leaving a continuous trail but failing as a colony to establish any stable trails. Figure 16 shows the eventual formation of a dominant trail. Note that only two virtual ants are wandering and not following the one stable trail. Third, results from [109] indicate that the ability of a group to form strong trails is inversely related with individual fidelity to the trails.

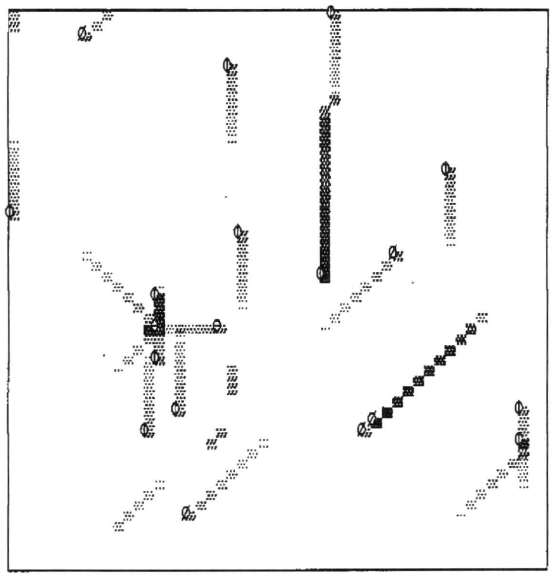


Figure 15. Early, disordered phase of CA simulation (taken from [108], p. 122)

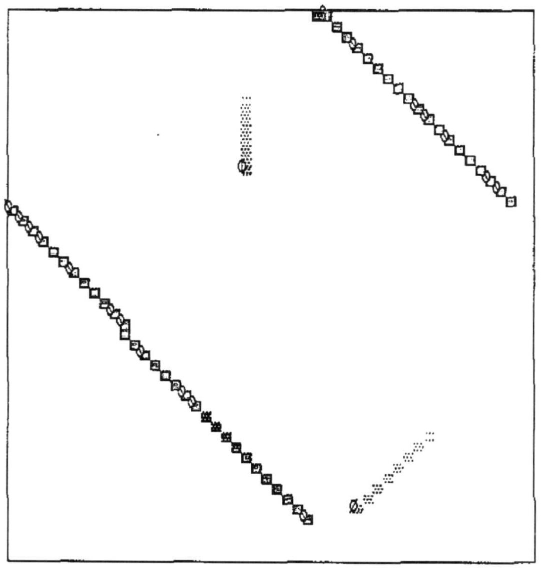


Figure 16. Formation of a dominant trail (taken from [108], p. 123)

Section 4.5: Model-Based Predictions

The models presented above effectively reproduce much of the complex system behavior seen in the recruitment-based foraging of social insects. Each model has a different perspective and level of mathematical rigor, making their contributions complementary to a full theoretical understanding of mass recruitment and self-organization. Taken as a whole, the results of these models provide several key qualitative predictions pertaining to the results of simulation and experimentation with real robots. The predictions are listed below and arranged according to their correspondence to one of the three characteristic properties of self-organization (see Subsection 3.1.3).

- Emergent Pattern Formation
    - A critical density of foragers is required to generate some trail patterns.
    - Formation of an established trail (or trails) may follow an initial "disordered phase" of unpredictable duration.
    - The nature of the trail pattern depends on parameters such as the number of foragers, food distribution, pheromone decay, etc.
    - Trail formation relies on autocatalysis of random fluctuations in the paths of searching foragers.
- Multistability
    - When multiple food sources are present, a number of stable states may exist, ranging from exclusive exploitation of one source to uniform exploitation of all available sources.

    - When multiple food sources are present, under certain conditions, only one stable state exists.
    - The foraging group may collectively choose a suboptimal stable state.
- Bifurcation Phenomena
    - Variations in key system parameters may correspond to dramatically different food source exploitation schemes.
    - Oscillations in certain parameters (especially in the number of active foragers) may allow a system near a bifurcation point to switch from a suboptimal to an optimal solution.

Chapter 5: Simulation

Section 5.1: Why Simulate?

Computer simulation is extremely common in the research of swarm intelligence. In fact, simulations are the most common method of researching such systems. They tend to be far simpler to implement than experimentation with real robots, especially for a large number of agents. Simulations are popular due to the ability to test the behavior of hundreds or thousands of individual agents, which is still unprecedented in physical experiments. Simulated swarms often generate results to confirm the predictions of theoretical models. They are also commonly used to predict the behavior of real robots. Due to these different aims, there are two major classes of computer simulations: those attempting to closely simulate and/or forecast real-world operation of physical robots, and those seeking to verify or augment theoretical understanding of self-organization.

The computer simulations in this work fall into the second category. The aim is not to accurately represent physical robots in a simulated computer environment. For example, two or more individual agents in this simulation were permitted to momentarily occupy the same space. In a simulation used to directly predict robot performance, collisions avoidance behavior would be included in the simulation. The overall goal here is to investigate the predictions of theoretical modeling from the literature (see Chapter 4), visualize system behavior to enhance understanding, and extend theoretical knowledge of scenarios for which mathematical modeling is intractable. It is important to emphasize the last point regarding cases in which theoretical work is either incomplete or absent altogether. In these cases, simulations are extremely important, as they advance the understanding of swarm-intelligent systems where mathematical models are as yet undeveloped.

A number of simulation tools exist for the purpose of investigating swarm intelligence and self-organization. These tools range in complexity and platform compatibility, as well as availability to an independent researcher. For this work, StarLogo was chosen as the simulation utility. This software is widely available, free, PC-compatible, and user-friendly. An introduction is provided in the next section.

Section 5.2: Introduction to StarLogo

The StarLogo programming and simulation environment was born out of a collaborative research effort in the late 1980s between Mitchel Resnick and Seymour Papert of MIT's Media Laboratory. As an aside, Papert's book entitled Mindstorms inspired LEGO's Mindstorms brand name (see Section 6.2). They developed an educational programmable robotics system known as LEGO/Logo. The system employed the Logo programming language to operate robots built from LEGO components. The Logo programming language had previously (late 1960s) been developed as an educational tool for children. It allowed a programming novice to manipulate one or a few on-screen "turtles" using a series of commands. Inspired by decentralized systems, cellular automata, and Brooks' subsumption architecture, Resnick sought to create a versatile simulation tool with Logo as a foundation. The result is StarLogo, which Resnick describes as:

> *…a massively parallel programming language that lets people control the actions of (and interactions among) thousands of computational objects. Whereas*

traditional versions of Logo allows users to control a single graphic "turtle" (or maybe a few graphic turtles), StarLogo give users control over thousands of graphic turtles. With StarLogo, people can create and explore a wide variety of decentralized systems.[84], p. 33

Like the original Logo, StarLogo is an object-oriented programming language with two main types of objects: “turtles” and “patches.” Unlike in more advanced programs such as C or C++, the user cannot create new classes of objects. The user can create several “breeds” of turtles, however. In many situations, such as simulation of an ant colony, only one breed of turtle is required. Both turtles and patches possess a small number of state variables, and both are able to execute StarLogo commands. Instructions are provided to the turtles and patches by writing *procedures*. Several of these procedures constitute a StarLogo program. The turtles and patches are capable of parallel operation, facilitating study of massively-parallel decentralized systems. Refer to [84] or [110] for more information, or Appendix A for StarLogo documentation.

Section 5.3: Program Description

StarLogo programs contain two accompanying sections: turtle procedures and observer procedures. Turtle procedures define the set of instructions carried out by each turtle, in parallel. From within a turtle procedure, an individual turtle can also ask a patch to execute a patch command. Observer procedures serve a variety of functions, from program initialization to data collection. The *observer* can ask patches or turtles to execute commands, create or destroy turtles, and/or monitor the status of patches and turtles.

Subsection 5.3.1: Turtle Procedures

There are three turtle procedures, which can be cast in terms of behavior-based control. Each procedure represents a particular behavior necessary to complete the task of foraging. The behaviors are based closely on the mass recruitment behavior of an individual worker ant, as described in Subsection 3.3.1. Each ant operates completely in parallel, acting as an autonomous agent with a homogeneous group. Refer to Appendix B for the complete StarLogo code for each procedure.

The default behavior is *wander*, in which each virtual ant moves forward along a partly random path. Each forward step of one unit is followed by a random heading perturbation of at most 25°. The result is a meandering path with fairly smooth and wide turns, and devoid of abrupt changes in direction.

If, during *wander*, an ant lands on a patch with *pheromone* greater than zero, the *follow* procedure begins. The ant rotates in place while checking the status of the patch one unit ahead of its current location. If the patch ahead is part of the trail and farther from home than the current location, the turtle moves to that patch and begins the process again. If the patch ahead is a food patch, the turtle moves onto it and starts *go-home*. In this manner, the turtles follow the trail of virtual pheromone to a food source. A timeout setting causes a turtle to abandon following and resume *wander* when the trail disappears or dead-ends.

The third procedure, *go-home*, involves "picking up" the food item (coloring the patch black) and returning to home. The ant sets a heading for the Cartesian origin, the center of the foraging field, and moves toward home. Setting a heading toward home contains a slight random element, producing minor deviations from the straight-line path to home. As it moves toward home, the virtual ant increments the value of *pheromone* on each patch by five.

Subsection 5.3.2: Observer Procedures

There are four observer procedures in this program, but one of the procedures is essentially a sub-procedure called by another procedure (see Appendix B). This leaves three main procedures, listed below along with the function(s) of each.

Table 5. StarLogo Observer Procedures

Procedure	Function(s)
setup	* draws foraging field boundary and home * generates distribution of food patches * creates turtles (virtual ants)
decay	decreases pheromone concentration (periodically)
color-patches	updates color of patches based on pheromone concentration

Most of the functions of the observer procedures are trivial, but the pheromone decay merits elaboration. Each patch is assigned a state variable, called *pheromone*, which represents the virtual pheromone concentration. This variable has a maximum of 100 and a minimum of zero. The maximum corresponds to a light shade of green (almost white), zero to black, and five to a very dark shade of green. Each time a trail-laying ant passes over a patch, *pheromone* is increased by ten. At periodic time intervals, which vary in length depending on the selected decay rate, *pheromone* is decremented by five for all patches. The decay procedure also ensures that *pheromone* does not fall below zero or rise above 100 for any patch. While this simplistic decay mechanism does not attempt to mimic biological pheromone evaporation, it is sufficient to produce swarm-intelligent system behavior very similar to that found in real ant colonies. Through these computer simulations, it became evident that biologically precise decay trends were unnecessary. As a result, the decay mechanism required no further sophistication for this application.

Section 5.4: Simulated Scenarios

Computer simulations were conducted of six separate scenarios, five of which involve variations of group foraging via mass recruitment. The first scenario is the exception, as it investigates a related but separate facet of self-organization in foraging. The process of *group recruitment* in ants generates pheromone trails in straight lines extending from the nest to the food source. This tendency is explored theoretically in Section 4.4 and simulated using StarLogo as the first scenario.

The remaining scenarios simulate full-scale foraging tasks. The first, Case 1, is the only non-cooperative foraging situation. All others include trail deposition and following, as well as trail decay in time. The trails are represented by varying shades of the color green. Brighter shades correspond to higher pheromone concentrations. Data collection in each trial occurred internally, using StarLogo constructions designed for monitoring simulations.

Subsection 5.4.1: Straight Trail Simulation

Trail-straightening behavior has been discussed from a theoretical perspective in Subsection 4.4.1. A simple simulation provides confirmation of the theoretical result and allows one to easily visualize the self-organizing process. The simulation includes one virtual “leader” ant that starts at home and wanders along a random path while leaving a trail. After some time interval, the first follower ant begins walking, maintaining a heading directed toward the leader. The follower also leaves a trail, but of a different color. This process is repeated several times, with each follower treating its immediate predecessor as the leader. The time interval between each follower is varied to demonstrate the effect on convergence to a straight path.

Subsection 5.4.2: Case 1

The first case is an example of non-cooperative group foraging. A number of virtual ants depart from home and retrieve single items of virtual food without leaving trails. The food distribution is generated randomly before each trial within the StarLogo program. 100 food patches are distributed randomly within the foraging area to start the test. When only 25 remain, the simulation is stopped. The simulation was run using a range of 2-90 virtual ants.

Subsection 5.4.3: Case 2

The second case is the first to involve cooperative group foraging. Mass recruitment is the mechanism of cooperation that produces useful and complex group behavior. The food distribution is a single cluster of 60 food items, located near the edge of the foraging field. Positive reinforcement allows the virtual ants to generate persistent pheromone trails to the cluster of virtual food. When 12 food patches remain, the simulation is stopped. The simulation was run using a range of 5-90 virtual ants and the pheromone decay rate was varied as a parameter.

Subsection 5.4.4: Case 3

This case is similar to Case 2, but with two virtual food clusters. The ant behavior is identical, as are the mechanisms by which self-organization occurs. In order to study more complex group behavior, a number of variations were introduced in this case. The simplest case is two clusters of equal size (41), equidistant from home, and located on opposite sides of home. One simulated variation of this was making one of the two clusters closer to home (ratio of distances = 2/3), while maintaining equal size and angular position. A second variation was to introduce the closer food source at some time later than the farther source. The source close to home was introduced after 10 out of the 41 food items were retrieved from the original source. These variations allow observation of system behavior such as multistability, bifurcation phenomena, adaptability, and collective decision-making. The simulation was stopped after all food items were retrieved. The simulation was run using a range of 30-100 virtual ants. Trail decay rate was also varied as an experimental parameter.

Subsection 5.4.5: Cases 4 and 5

Cases 4 and 5 differ only in the number of food sources present. In Case 4, 3 food sources of equal size (22) were presented, equidistant from home. The same applies to

Case 5, except four sources were used (of size 25). These two cases allow observation of multistability and bifurcation phenomena, due to a number of potential exploitation patterns in each situation. Current theoretical modeling fails to explain all of the phenomena present in competition between multiple sources, especially more than three. The simulations were stopped after all clusters were fully depleted. The simulation was run using a range of 30-90 virtual ants; pheromone decay rate was also varied.

Section 5.5: Results

Subsection 5.5.1: Straight Trail Simulation

This simulation was performed to confirm and provide visualization of the theoretical result presented in Subsection 4.4.1. In the following figures, the path of the initial ant is shown in black. The first follower ant sets out from home and heads directly toward the leader ant. The second follower operates according to the same rules, but following the first follower ant, rather than the original leader. As expected, for any random initial path taken by the "leader" virtual ant, the paths of subsequent followers converged to a straight line. One example of this is shown in Figure 17, with the initial path shown in black. Note that the path of the final ant begins to approximate a straight line. Using higher numbers of followers results in a perfectly straight line. Figure 18 illustrates the progression from the initial to final path, using six total virtual ants. Six snapshots in time are provided (from top left to bottom right), with each figure showing the trail pattern shortly after each ant departs from home.

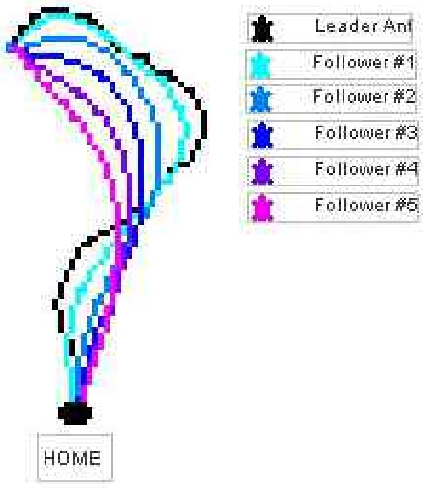


Figure 17. StarLogo simulation of trail straightening

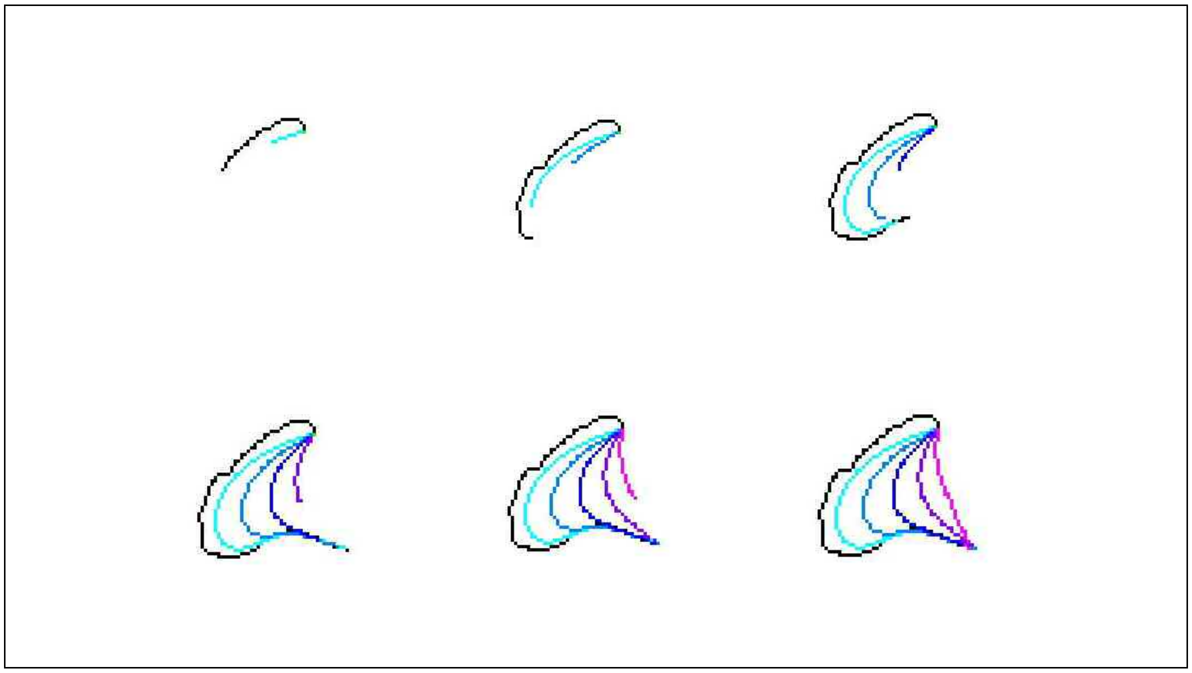


Figure 18. Six steps in the sequence of trail straightening (top left to bottom right)

Note that a similar process of convergence to a straight path occurred in multiple tests with random initial paths. The convergence was accelerated by using shorter intervals of time between departures from the nest.

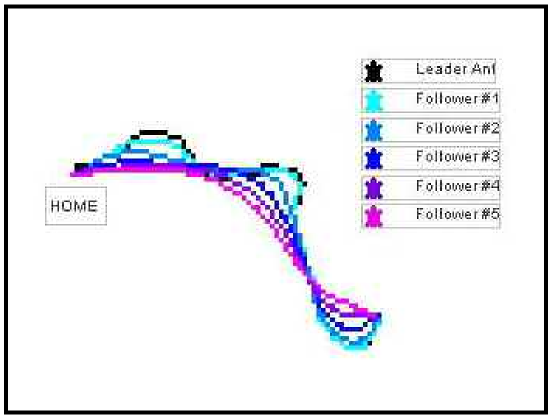


Figure 19. Trail straightening with short departure interval

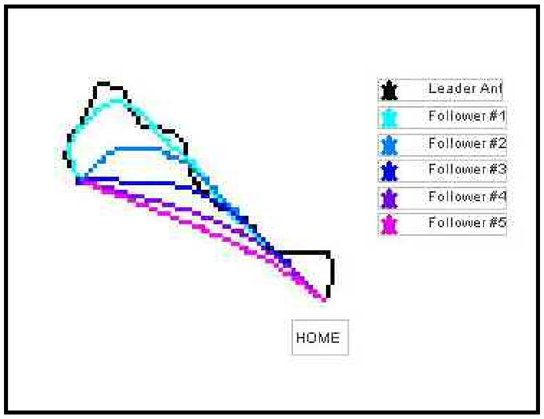


Figure 20. Trail straightening with long departure interval

The simulation shown in Figure 19 featured a short departure interval between ants. As a result, each ant followed the preceding path more closely, resulting in slower convergence to a straight line. By comparison, the longer interval illustrated in Figure 20 produced accelerated convergence to a straight line. The path of the fifth follower ant is nearly a direct path from home to the endpoint.

Subsection 5.5.2: Case 1

In the first simulated case of non-cooperative foraging, the initial food distribution is generated automatically through StarLogo. As shown in Figure 21, the initial test setup includes 100 randomly distributed food items represented on-screen by 100 yellow “patches,” or pixels. The outer boundary of the simulation is colored red, which cannot be crossed by a simulated ant. Home is shown at the center of the foraging field in brown. Initially, as in experimentation with real robots (see Chapter 6), the foragers are arranged facing directly away from home with uniform angular distribution. When the simulation is started, each forager begins to search, with a significant random element to the search path. The initial direction of each ant, therefore, does not determine which sector of the field it will visit first. Figure 22 shows the status of the simulation after 20 time steps. With such a large number of ants (N=70), most of the initial 100 food patches have been retrieved. Note the absence of pheromone trails, which are useless in such a food distribution.

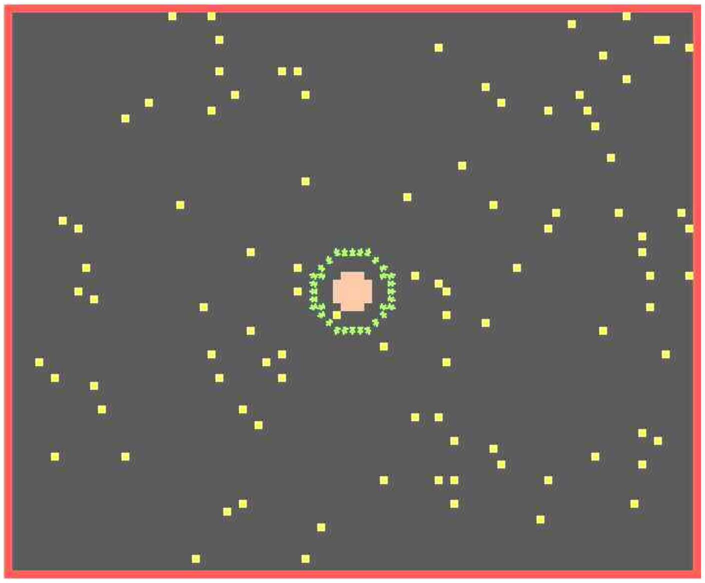


Figure 21. StarLogo simulation (100 distributed food patches, 70 ants, time = 0)

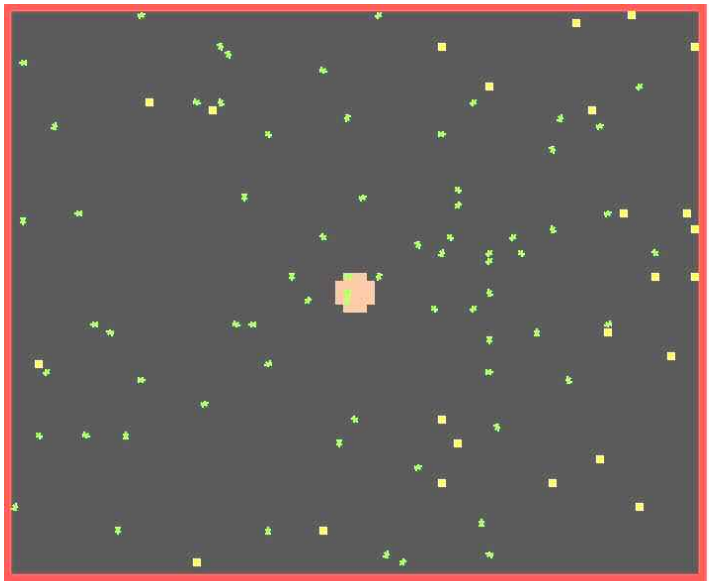


Figure 22. StarLogo simulation (70 ants, time = 20)

The rate at which food is retrieved by the colony provides a useful global description of the foraging dynamics. Results (both simulation and experimental) will be presented, in part, by showing the time evolution of food collection. Figure 23 is one example of such a graphical representation of the foraging system. Notice that the task was considered complete after 75 out of 100 food items were retrieved. An interesting observation from the case of random food distribution is that the general curve shape remains the same for different numbers of ants in a given trial. The example shown in Figure 23 is very similar to the progression seen for $5 < N < 90$. In each case, the collection rate diminishes gradually as the system approaches task completion. This reduction in retrieval rate reflects the decreasing food density in the foraging field. At the beginning of a trial, the probability of each individual ant finding a food patch is higher than near the end of the test. As this applies to each individual, the global result is a steadily decreasing collection rate. This trend is seen for the entire range of N, but in every case the trend is weak. From beginning to end, the collection rate varies only slightly, but according to a reliable and gradual decay.

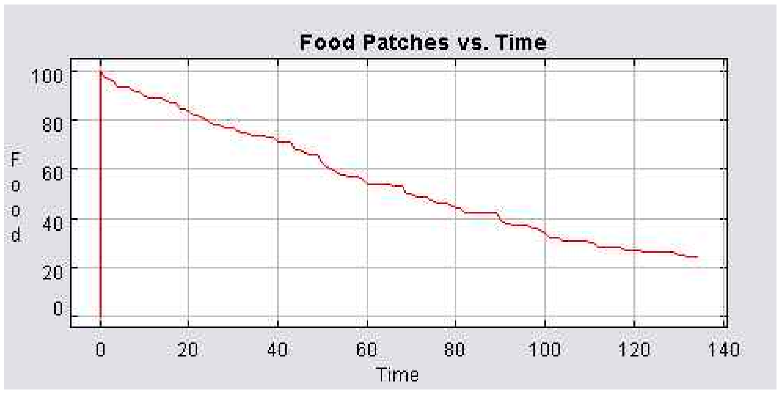


Figure 23. Time evolution of food remaining (N=10)

In order to investigate the dependence of performance on the number of foragers, N, the total time required to complete the task was chosen as a performance parameter. As discussed in Section 4.2, robotic (simulated and physical) foraging systems are often analyzed in this manner. Essentially, the system efficiency is defined according to time minimization. Figure 24 illustrates the relationship between the number of ants and the task completion time. As expected, the completion time rises asymptotically as N approaches zero. The trend closely resembles a power law relationship between the two variables. As N is increased, the performance improvement continues, but at a decreasing rate. In this simulation, the completion time begins to level off completely around N=100, at which point the task requires very few time steps. Saturating the foraging field with simulated ants does not correspond to a realistic biological system, as energy would be wasted by many of the foragers.

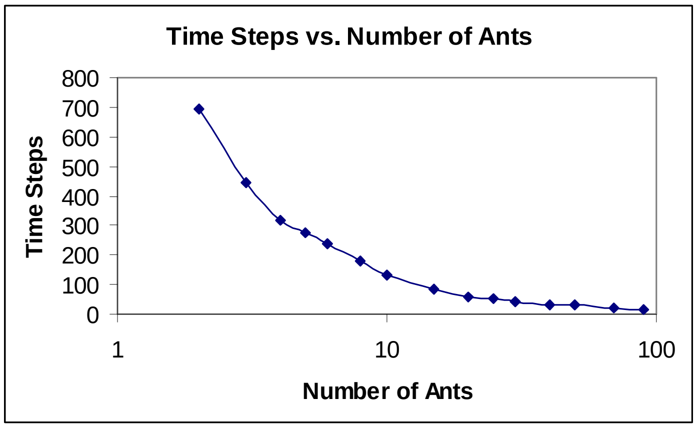


Figure 24. Completion time versus N (average of 3 trials[1])

Especially in the case of robotic foraging systems, unintended interaction between robots is an important issue. Many systems are not scalable to large numbers of robots, due to the detrimental effect of collisions between robots. In the simulation, two or more simulated ants were allowed to occupy the same space, thereby ignoring collisions. However, an accounting was kept of these "interactions." The data is shown below in Figure 25, in terms of collision frequency. The frequency of collisions allows comparison across trials of different duration. In what resembles another power law relationship, the collision frequency increases dramatically with the density of ants in the environment. Refer to Chapter 7 for a more detailed discussion of power laws and their appearance in foraging systems.

[1] All subsequent graphs of this type reflect averages over three trials, unless otherwise noted.

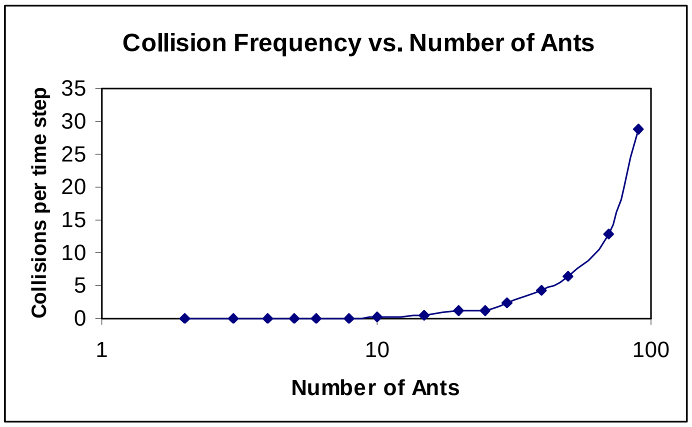


Figure 25. Collisions frequency versus N (average of 3 trials[2])

It is evident from this test that a trivial foraging task can be accomplished without cooperation, as one would expect. The group performance increases with group size, but solely due to increased spatial density, resulting in faster coverage of the finite area of the foraging field.

Subsection 5.5.3: Case 2

In the second scenario simulated using StarLogo, cooperation is introduced. The food is clustered together in one corner of the foraging field, a short distance from the red boundary. In this scenario and those to follow, the virtual ants are equipped with trail-laying and following behavior. Mass recruitment is used to exploit the single source. However, negative feedback is also present in the form of simulated pheromone evaporation. The rate of pheromone evaporation, or decay, is a parameter varied in these simulations. A typical foraging process is shown in Figure 26 and Figure 27. These figures show the initial formation and subsequent reinforcement, respectively, of a

[2] All subsequent graphs of this type reflect averages over three trials, unless otherwise noted.

pheromone trail to the single source in the top left corner of the field. Note the increased intensity (brighter shades of green) and width of the trail in the second figure. For sufficiently slow pheromone decay rates, d, the colony of virtual ants builds a stable trail to the food source. In Figure 26 and all subsequent StarLogo screens, ants following a trail are colored red.

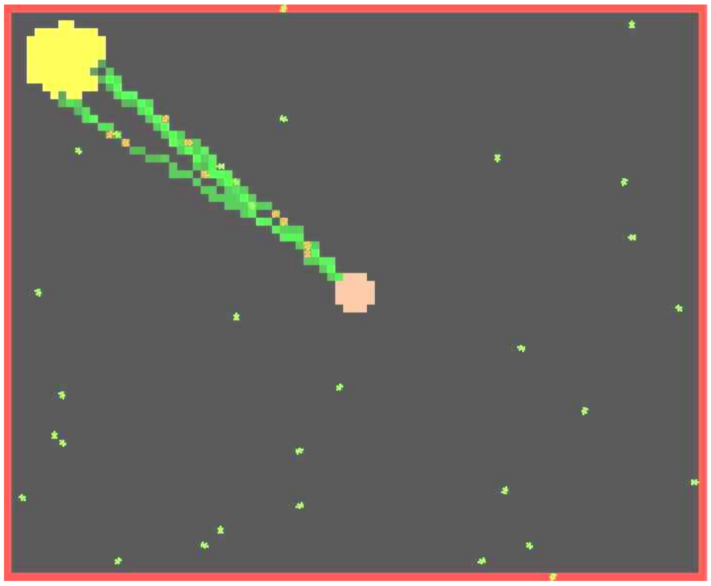


Figure 26. StarLogo simulation (N=40, time=30)

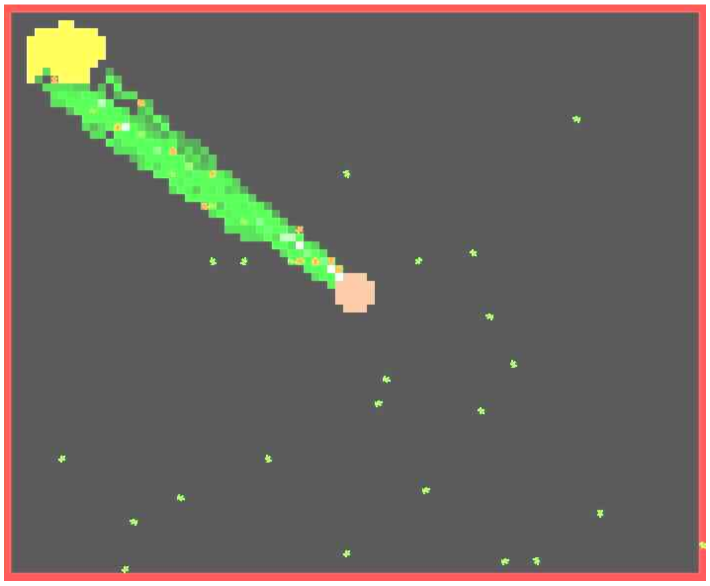


Figure 27. StarLogo simulation (N=40, time=50)

The rate of pheromone decay, d, is an important parameter in these simulations. Three values of d were used in the case (d = 10, 20, and 35). The first corresponds to a decay rate slow enough for a single ant to follow its own trail back to the food source. For d = 20 or d = 35, this is not possible. In the case of d = 35, the pheromone trail decays so rapidly that the end of the trail nearest to the food source (deposited first) disappears before the ant arrives at home. As the parameters d and N are varied, the system exhibits three main types of behaviors. From small d and/or large N, the colony reliably establishes a strong trail to the food source. The resulting time evolution of food collection is shown in Figure 28.

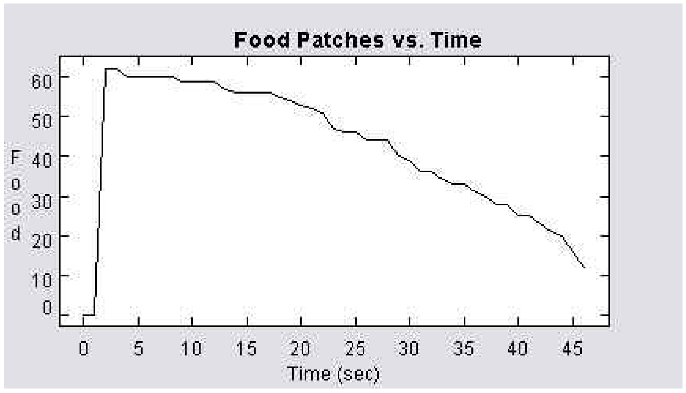


Figure 28. Time evolution of food remaining (N=70, d=10)

By comparing this curve to the one from the non-cooperative case (see Figure 23), a fundamental difference becomes evident between the two cases. While non-cooperative foraging produces a gradually decreasing collection rate, simulated mass recruitment generates a steadily increasing rate of collection. Due to positive feedback via the pheromone trail, the rate at which items of food are retrieved increases throughout the test. This represents a clear advantage of cooperation, as the performance of the group improves over time. The difference in concavity between the two types of curves shown in Figure 23 and Figure 28 illustrates the clear benefit of cooperation. The downward concavity of Figure 28 is a signature of cooperation in foraging.

The second type of system behavior (higher d and/or lower N, compared to behavior described in previous paragraph) represents a transition from effective cooperation to the failure of stigmergic cooperation caused by rapid trail decay. This transition behavior is characterized by short intervals of time in which the collection rate begins to increase. These periods occur when a trail is briefly established and utilized by a few followers. However, as shown in Figure 29, negative feedback due to rapid trail decay, combined with insufficient ant density, periodically interrupts the momentum of the system. In terms of the food vs. time curve, this type of behavior is characterized by intervals with the signature downward concavity interspersed with flat sections of the curve.

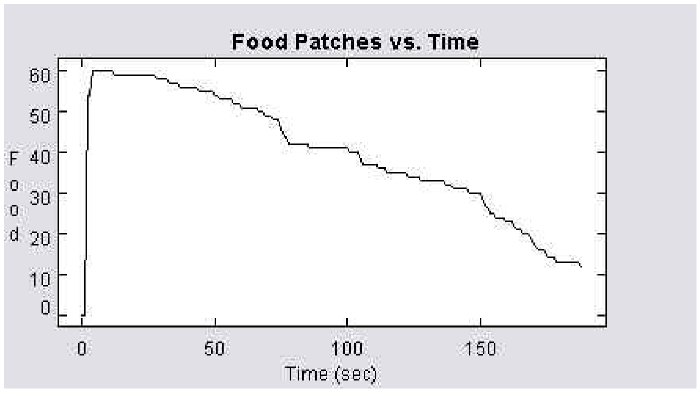


Figure 29. Time evolution of food remaining (N=25, d=35)

The third type of system behavior (high d and/or small N) resembles that seen in the non-cooperative case, Case 1. The number of food items remaining in the field tends to follow a nearly linear trend in time, with a slight leveling-off near the end of the trial. The concavity is slightly positive, as shown in Figure 30, which is quite similar to behavior of the system foraging without trails for randomly distributed food items. The main difference is that food discovery is much less probable, due to the spatial concentration of food items in one corner of the field. This low probability of discovery results in extended periods during which no items are collected. The curve in Figure 30 remains flat from t = 425 to t = 600, indicating a period of fruitless searching by the colony. For rapid trail decay (d = 35) and small group size (N < 30), the system exhibits this non-cooperative type of behavior, despite the ability to deposit and follow trails.

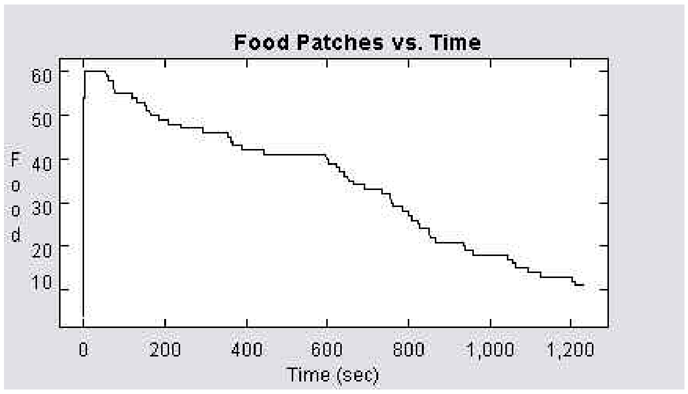


Figure 30. Time evolution of food remaining (N=10, d=35)

The task completion time for each test reflects the three modes of system behavior described above. Figure 31 shows the completion time versus group size with the decay rate, d, as a parameter. For N > 40, the decay rate has minimal effect on the system behavior, as ant density is sufficient to establish and maintain a stable trail. For N < 40, the curves begin to show a dependence on decay rate. As the number of ants decreases, the curve corresponding to d = 35 transitions first into non-cooperative behavior, characterized by much longer task completion time. For slower decay rates, this transition occurs at a smaller group size. The intermediate behavior, in which trails are formed for short periods of time before fading out, corresponds to completion times between approximately t = 150 and t = 325.

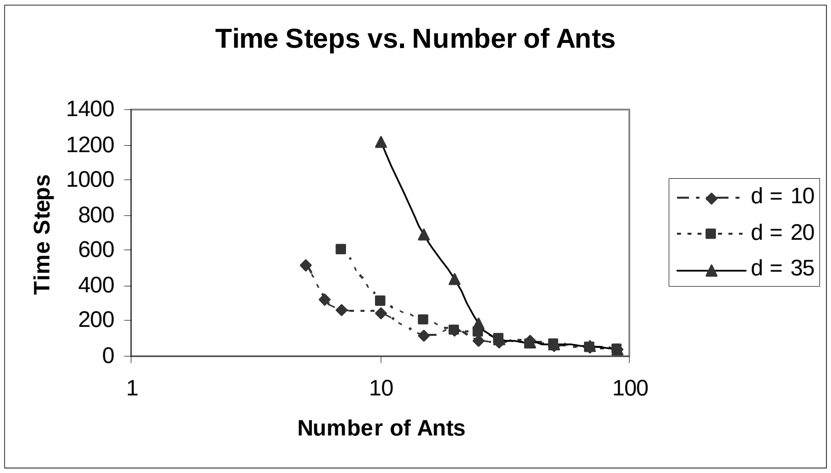


Figure 31. Completion time versus number of ants (d=decay rate)

The relationship between group size and collision frequency is shown in Figure 32. As expected, the collision frequency increases with group size for all decay rates. The curves lie in close proximity to one another, indicating that pheromone decay rate is not an important parameter with respect to collision frequency. The trends closely resemble that seen in Figure 25, resembling a power law relationship.

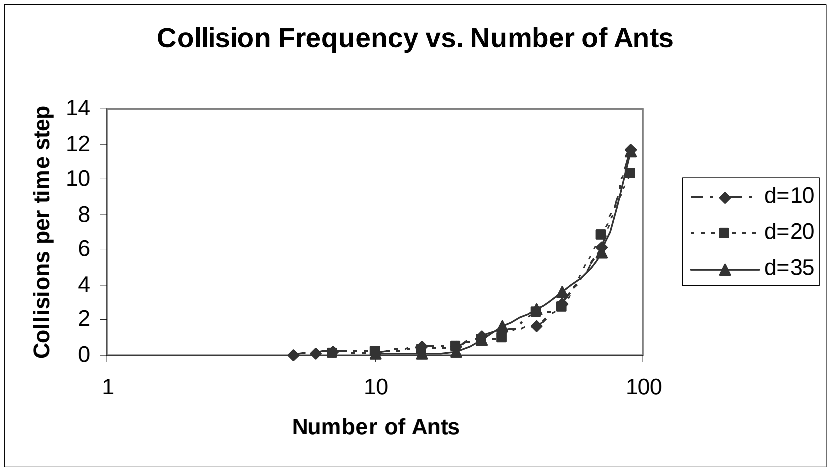


Figure 32. Collision frequency versus N

Subsection 5.5.4: Case 3

Case 3 features three variations of the two-cluster scenario. The first situation is nominal, with two equidistant and identical food sources. The second is a variation, with one food source moved closer to home. Lastly, the third scenario involves presentation of the closer food source after partial exploitation of the first source. Results for these three variations on the theme are presented in turn.

EQUIDISTANT FROM HOME, BOTH PRESENTED AT T = 0

In this scenario, the system generally operated in one of three states. For some combinations of decay rate and group size, the group failed to perform cooperatively, representing one state. The other two foraging modes represent two coexistent stable states: (1) preferential exploitation of one source and (2) simultaneous exploitation of both sources. Examples of trials that exhibited the two modes are shown in Figure 33 and Figure 34, respectively. Either of the two modes was possible for many combinations of parameter values. Furthermore, the system sometimes switched between the two modes during a single trial.

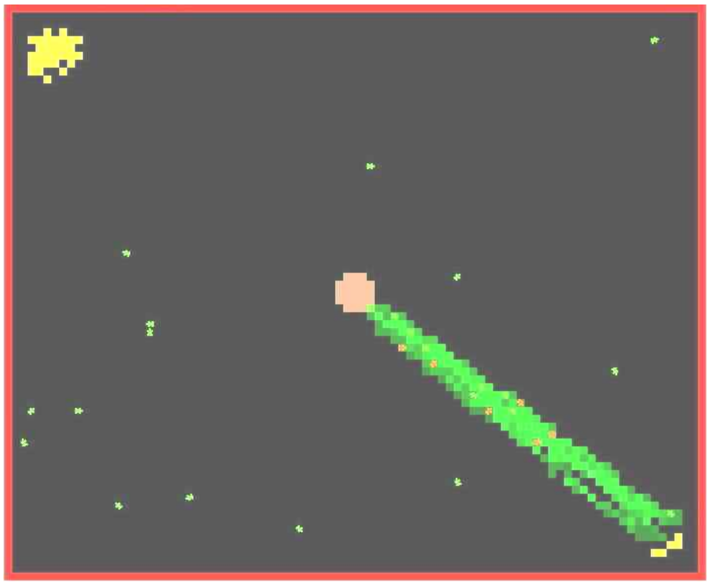


Figure 33. StarLogo simulation (N=30, d=25, t=407)

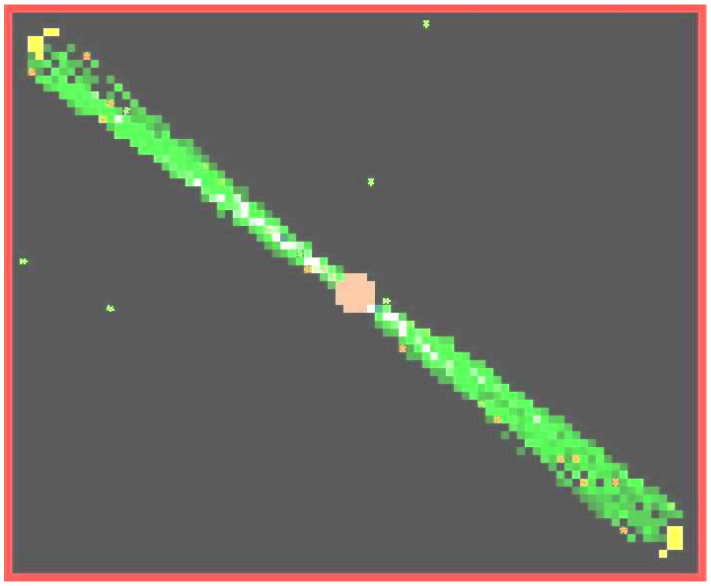


Figure 34. StarLogo simulation (N=30, d=15, t=206)

The two plots below clearly illustrate the two cooperative system states. In Figure 35, the colony initially discovers both sources and establishes weak trails to both. However, due to insufficient ant density and/or rapid trail decay, the colony is only able to maintain a stable trail to one source (Food Source B). The colony then focuses solely on this source until it is fully depleted. It then switches focus to the remaining source (Food Source A).

Note that the choice of Source B over Source A is entirely dependent on random initial fluctuations of the system. In cases where one source was collectively chosen by the colony, Source A and Source B were chosen with roughly equal probability.

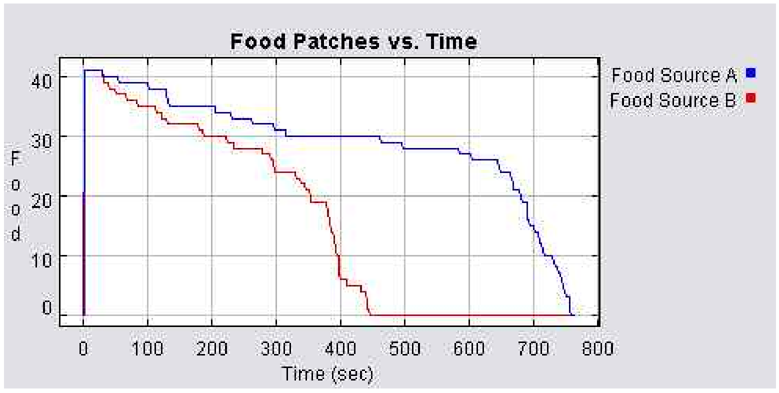


Figure 35. Time evolution of food remaining (N=30, d=25)

Figure 36 illustrates mutual utilization of the two food sources. The number of food items diminishes steadily at roughly the same rate for both Source A and B. This type of behavior occurred for large group size and/or slow trail decay. In these conditions, the colony is able to establish two stable pheromone trails.

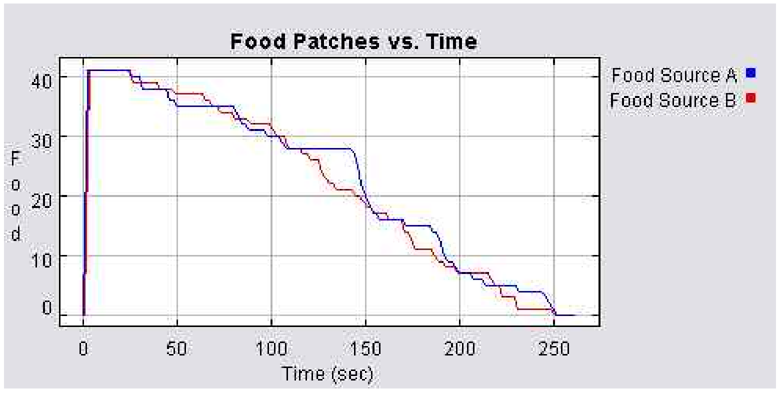


Figure 36. Time evolution of food remaining (N=30, d=15)

For each value of the parameter d, trials were run for N = 30, 50, and 70. Figure 37 illustrates the performance achieved in each test condition. For N = 70, ant density was sufficient for concurrent exploitation of both sources, irrespective of decay rate, d. With smaller group size, the system performance decreases, indicating a transition to exploitation of a single source. For N = 30 and d = 35, the colony was unable to achieve cooperative behavior of any kind, resulting in a very high completion time (t = 1346).

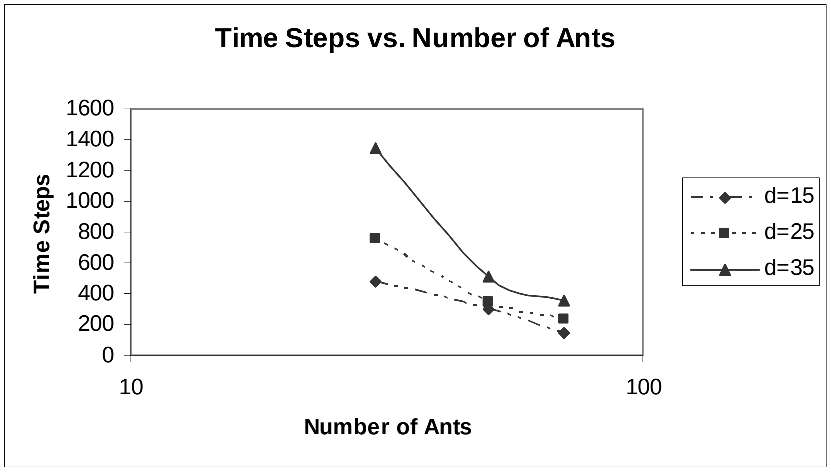


Figure 37. Completion time versus number of ants

UNEQUAL DISTANCE FROM HOME, BOTH PRESENTED AT T = 0

This scenario features two identical sources, but with the near one located at 2/3 of the distance to home of the farther source. The same three possible system behaviors as in the equidistant scenario were observed. The main difference in this case is that, for exploitation of a single source, the probability of collectively choosing the near source was far greater than 50%. Of the 27 total simulation trials, only once was the far source exploited preferentially over the near source. Typically, unless trails were established to both sources, the system evolved like the example of Figure 38, Figure 39, and Figure 40. These figures represent three moments in time for a single simulation. In Figure 38, the colony has established a strong trail to the source closer to home. A very weak trail exists to the farther source, to which only one ant is traveling via the trail.   Figure 39 shows the point in time when the closer source is fully depleted. Note that the trail is very well defined. In time, however, the strong trail evaporates without reinforcement.

This increases the number of ants available to establish a trail to the remaining food source. Figure 40 shows that a trail is eventually established to the remaining source. The last remnant of the old trail is still visible to the lower right of home.

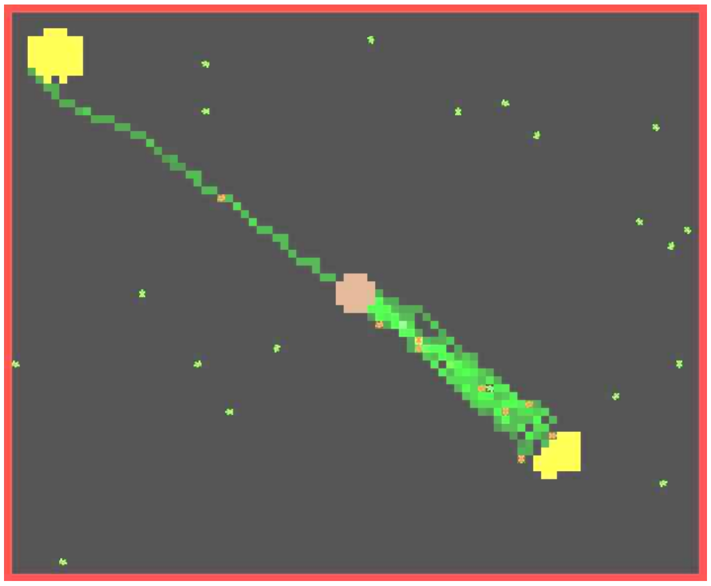


Figure 38. StarLogo simulation (N=30, d=15, t=106)

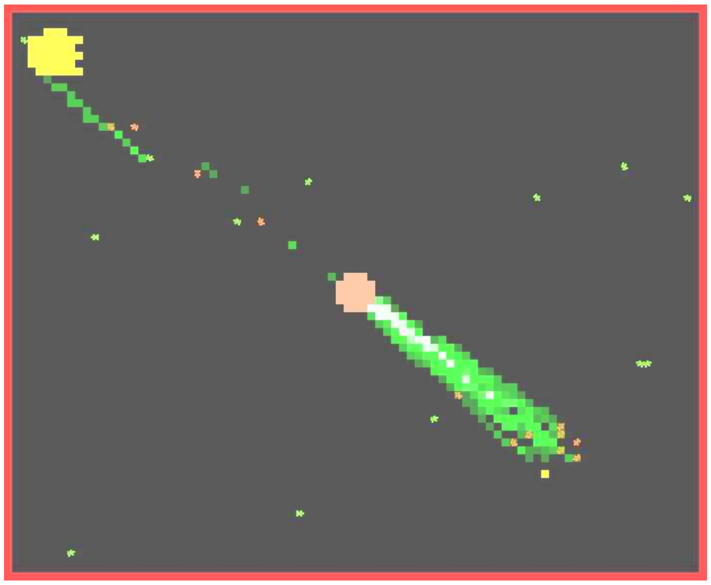


Figure 39. StarLogo simulation (N=30, d=15, t=203)

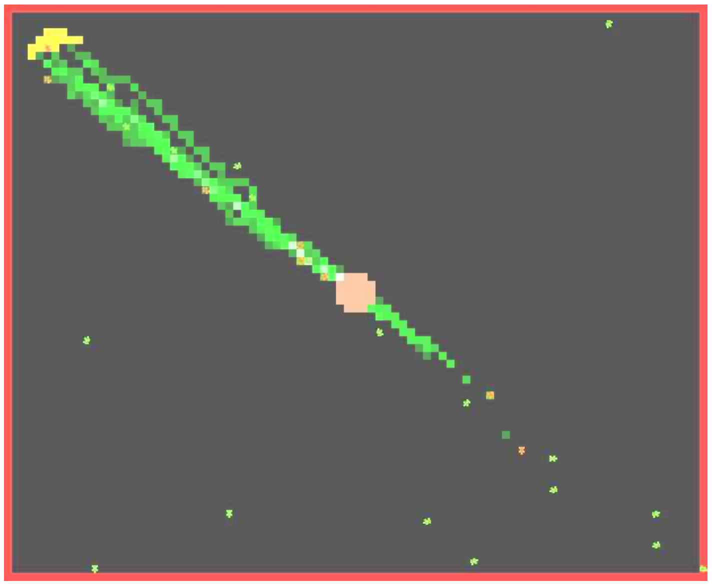


Figure 40. StarLogo simulation (N=30, d=15, t=302)

The two cooperative foraging modes are shown below by the time evolutions of food retrieval. By comparing Figure 41 and Figure 42, the difference is clear between the two situations. Figure 41 shows the behavior typical of a moderately-sized colony (N = 30) foraging under the influence of a rapid decay rate (d = 35). The colony is unable to establish two strong trails. In this case, the colony chooses Food Source B (closer to home), an efficient choice compared to Food Source A. Upon complete exploitation of Source B, the colony redirects its effort to Source A. This transition is clear from dramatic increase in collection rate from Source A (at approx. t = 475) soon after Source B is completed (at approx. t = 325).

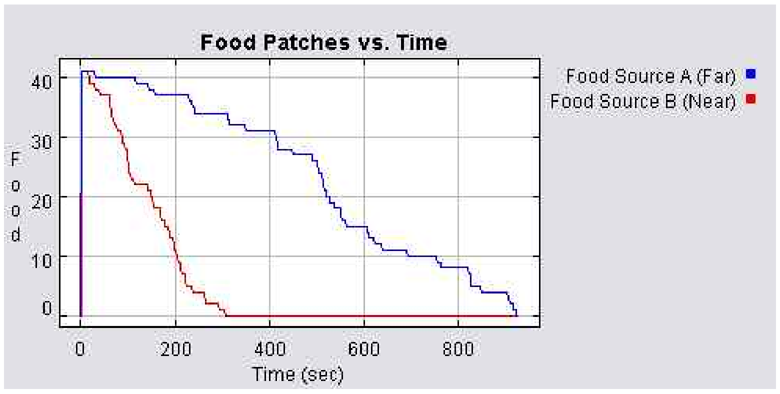


Figure 41. Time evolution of food remaining (N=30, d=35)

The collection trend in Figure 42 closely resembles that in Figure 36, when the sources were equidistant from home. In both trials, high ant density combines with slow decay rate to facilitate simultaneous exploitation of the two sources. Despite the unequal distance from home, both sources were visited at approximately the same rate throughout the test.

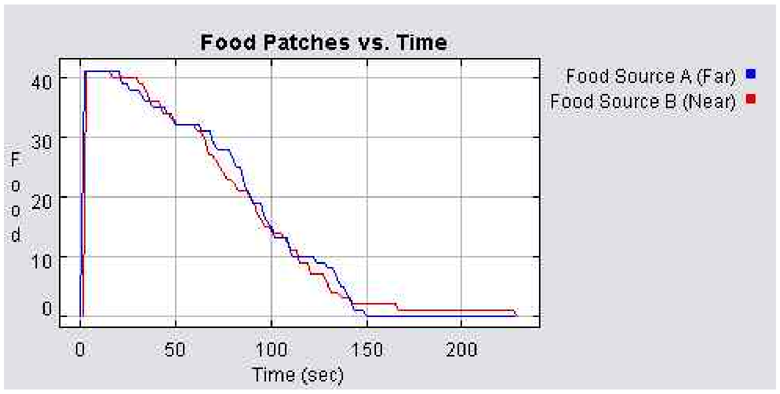


Figure 42. Time evolution of food remaining (N=50, d=15)

The task efficiency, measured by completion time, is shown as a function of group size and decay rate in Figure 43. As expected, the task efficiency increases with group size and the inverse of decay rate. Completion time less than approximately 300 time steps corresponds to a trial in which concurrent exploitation of the two sources occurred. For sequential exploitation of the two sources, more time steps were required.

At the upper extreme of completion time, for t > 800, cooperative behavior broke down due to rapid trail decay.

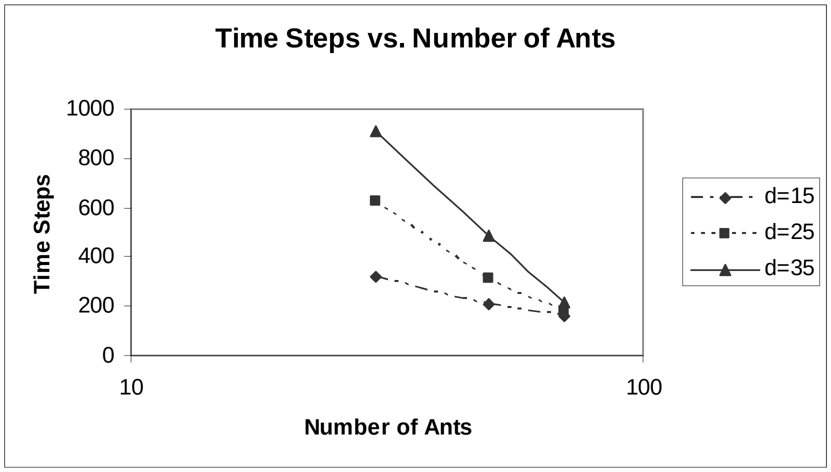


Figure 43. Completion time versus number of ants

UNEQUAL DISTANCE FROM HOME; ONE SOURCE PRESENT AT T = 0

In this variation of the two-cluster scenario, the source closer to home is presented at t > 0. It was introduced to the foraging field after retrieval of 10 food items from the source present at t = 0. This allows for initial discovery of the source and potential establishment of a trail prior to introduction of the new source. This situation was designed to test system adaptability in response to a dynamic environment. In approximately 90% of the trials, the new source was quickly discovered and utilized by the colony. This type of adaptation occurred even when a stable trail had been established to the source present at t = 0. The colony either abandoned the initial source or began simultaneous exploitation. The progression observed in one trial is shown in Figure 44 and Figure 45. The first shows the status of the colony immediately after introduction of the second food source (to the bottom right of home). Note that, while 10 food patches are missing from the initial food source, a stable trail has not been established. In addition, the new food source was discovered immediately after its placement into the field. The trails resulting from the initial discovery of this new source are quickly reinforced, and positive feedback eventually creates a strong trail to the new source (Figure 45). In this trial, the colony efficiently selected the new food source over the source initially present.

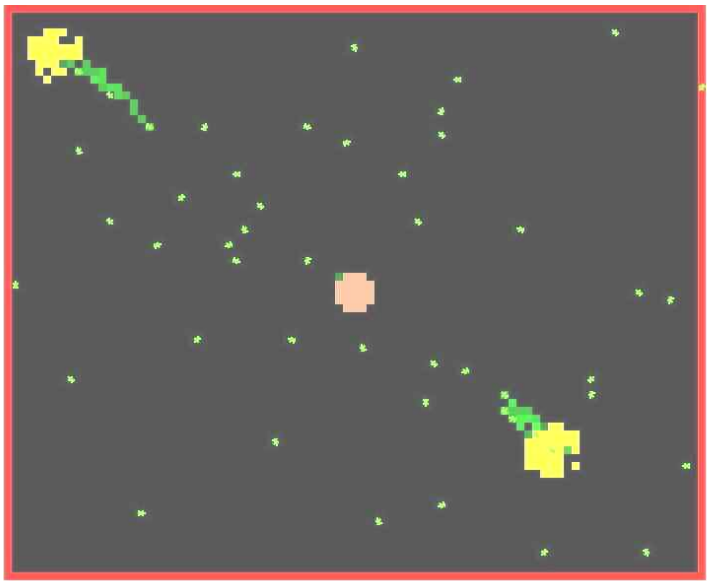


Figure 44. StarLogo simulation (N=50, d=35, t=157)

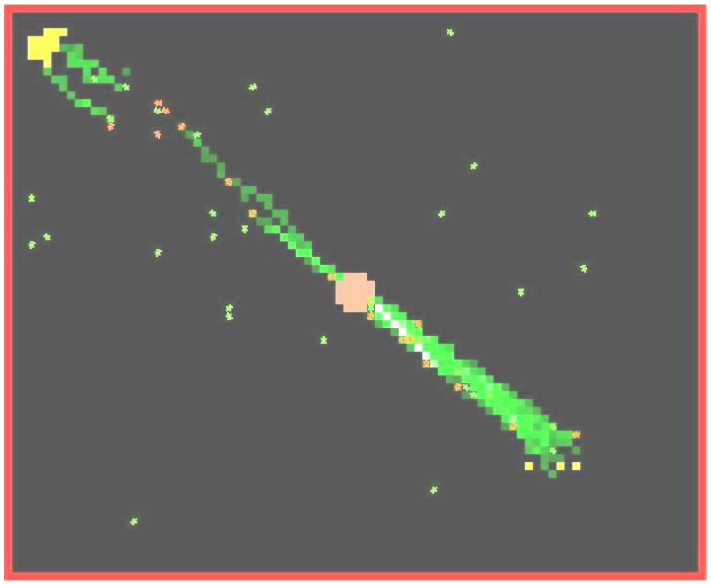


Figure 45. StarLogo simulation (N=50, d=35, t=250)

The following three plots represent three distinct responses to introduction of the new source. In the first plot, Figure 46, the collection rate from Food Source B accelerates quickly soon after its introduction. Interestingly, the collection rate of Source A is not noticeably affected by the new source. However, throughout the trial, the collection dynamics for Source A are characteristic of non-cooperation. In contrast, the collection curve for Source B exhibits the downward concavity produced by cooperative foraging. In Figure 47, the colony responds to the new food source by exploiting the two mutually. Note that this occurs with a large group of virtual ants and with a slow decay rate. These conditions allow the colony to discover and exploit the new source without abandoning the old source. A decision between the sources is not required, as system resources are able to utilize both concurrently. When the decay rate is increased to d = 35 (N remains constant at N = 70), the system is unable to establish a stable trail to the new source. It is discovered, but the colony eventually continues exploiting the original source before switching to new source (Figure 48). In this case, the momentum of positive feedback forced the system into a suboptimal foraging strategy.

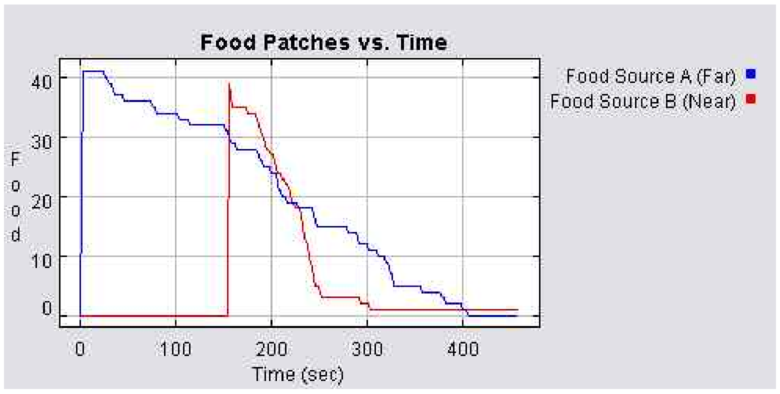


Figure 46. Time evolution of food remaining (N=50, d=35)

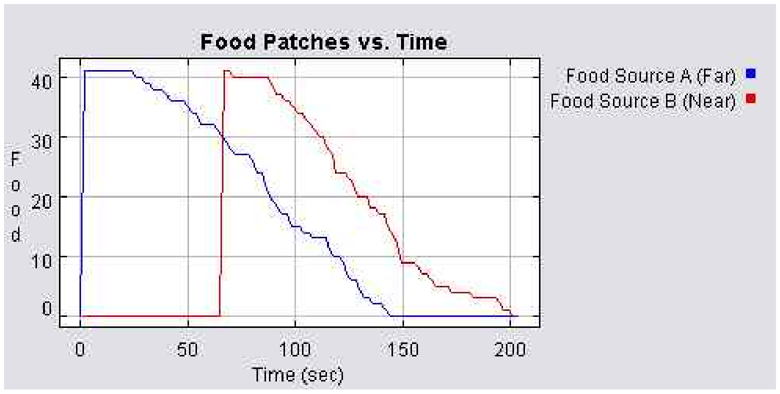


Figure 47. Time evolution of food remaining (N=70, d=15)

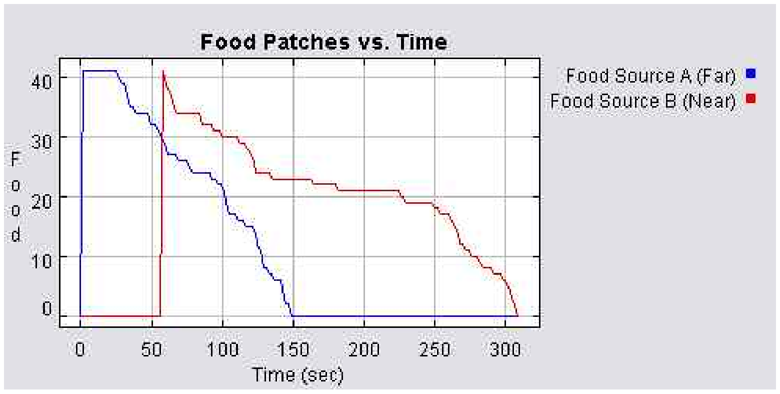


Figure 48. Time evolution of food remaining (N=70, d=35)

Figure 49 illustrates the overall task performance for all combinations of group size and pheromone decay rate. The trends are similar to those seen in Figure 43 and Figure 37. System performance is generally impaired by a rapid decay rate and small group size. The shortest completion times (t ≈ 200) occur when the system is able to exploit the new source without abandoning the original.

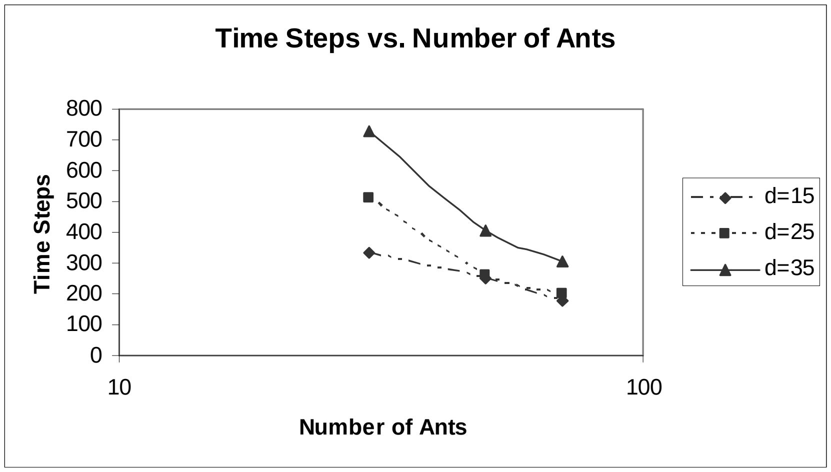


Figure 49. Completion time versus number of ants

Subsection 5.5.4: Case 4

Case 4 features three food sources, equidistant from home. All three sources were present at the beginning of each trial, with 120° between each source. All trials were run with N=100, with the decay rate varied from d = 20 to d = 55. This case was designed to investigate the existence of multiple exploitation schemes and the importance of decay rate as a system parameter. As discussed in the case of two food sources, more than one mode of foraging is possible for a given set of test conditions. With three sources present, three major modes of cooperative foraging were observed. The first mode consists of the establishment of one dominant trail, with minimal utilization of the other two sources (until the first is depleted). This type of exploitation is quite rare with N = 100, as the ant density is generally sufficient to support two or more stable trails. The second and third foraging states are simultaneous exploitation of two or three sources, respectively. These behaviors were much more common, again due to the large group size. The three images below from StarLogo display an example of the initial exploitation of two sources. Figure 50 shows the existence of two strong trials, with only a weak and incomplete trail leading to the food source above home. In Figure 51, both of the bottom food sources have been finished and only slight remnants remain of the trails. Notice that a numbers of ants are stranded on these weak trails. These “lost” ants eventually abandon following and resume the default wandering behavior. As a result, they become available for reinforcement of a trail to the one remaining food source. In Figure 52, the colony has rediscovered the final source and established a strong trail.

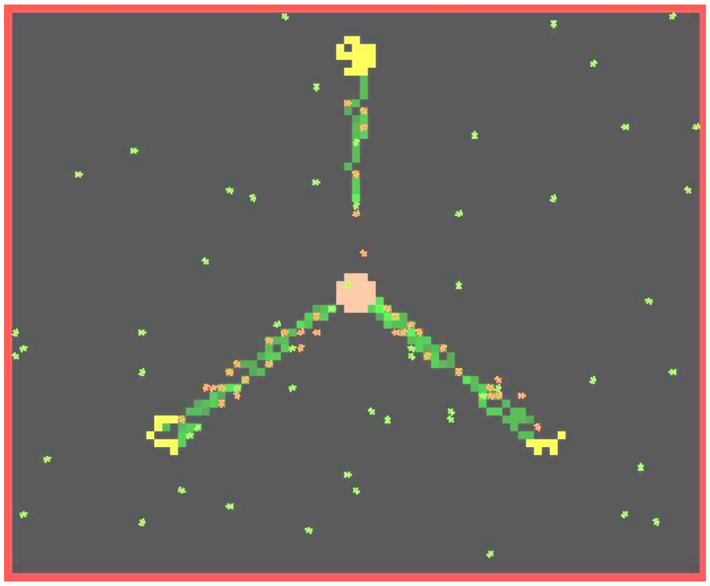


Figure 50. StarLogo simulation (N=100, d=20, t=132)

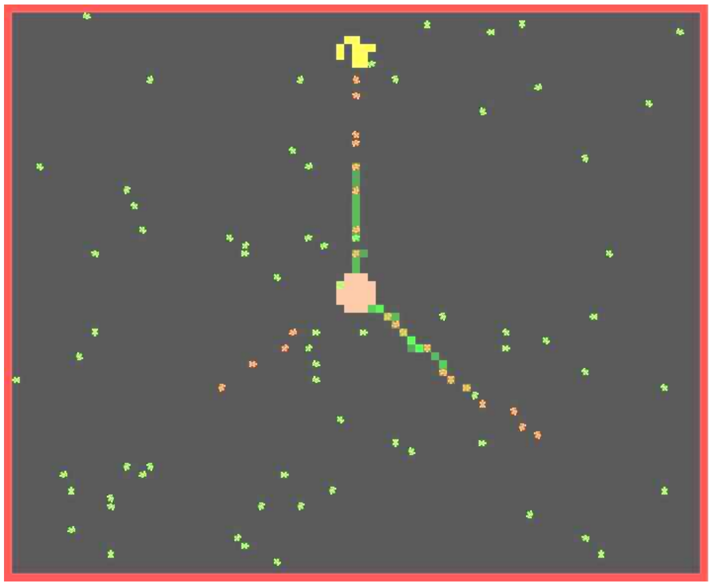


Figure 51. StarLogo simulation (N=100, d=20, t=185)

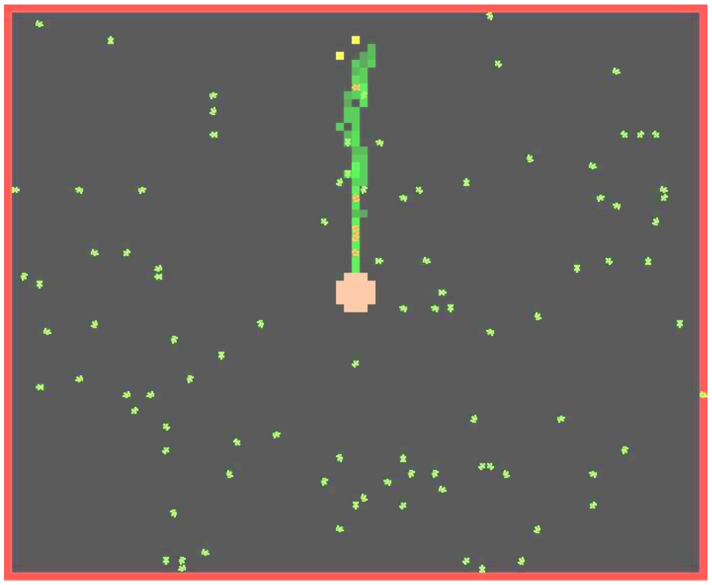


Figure 52. StarLogo simulation (N=100, d=20, t=213)

The following figures demonstrate the two most common exploitation patterns in this set of trials. Figure 53 shows concurrent exploitation of all three sources. Notice that the decay rate is quite slow (d = 15) in this trial, which allows the virtual ants to divide their resources between all three sources. When the decay rate is slow, trail lifetime is extended, decreasing the critical density of ants required for trail maintenance. For a higher decay rate (d = 25), the colony can only support two concurrent trails (Figure 54).

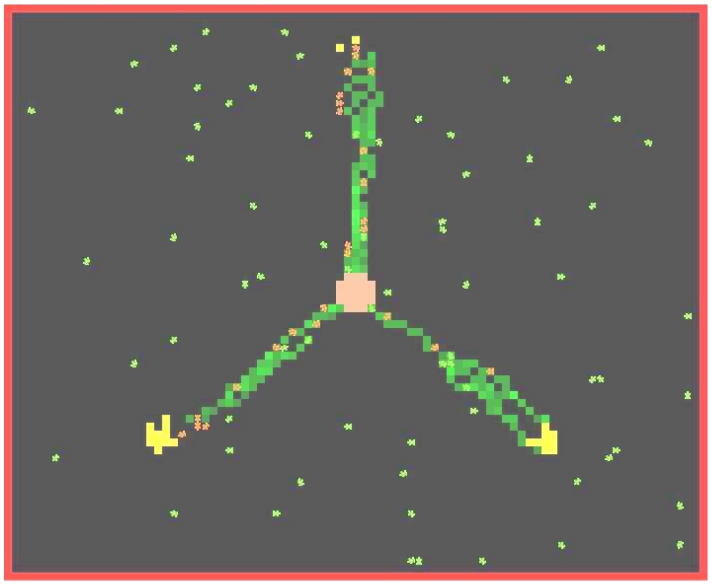


Figure 53. StarLogo simulation (N=100, d=15, t=140)

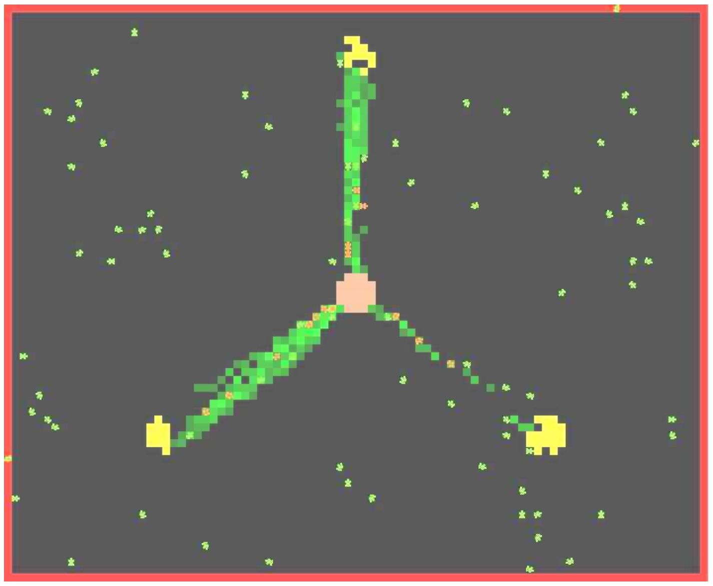


Figure 54. StarLogo simulation (N=100, d=25, t=74)

The time evolutions of food collection for each of the cooperative behavioral modes are shown below. The first plot, Figure 56, shows the effect of a high pheromone decay rate. The colony focuses on each food source in a sequential foraging strategy. Figure 55 is a clear example of utilizing two sources, while virtually ignoring the third. Note the time delay between completion of the first two source (t = 120) and the dramatic increase in collection rate to the remaining source (t = 190). During this intervening time, ants are still following the two dominant trails. It takes a short time for those trails to decay and for the wayward ants to commence searching for new food sources. Finally, in Figure 57, with a slow decay rate (d = 25), the colony forages from all three sources at the same time.

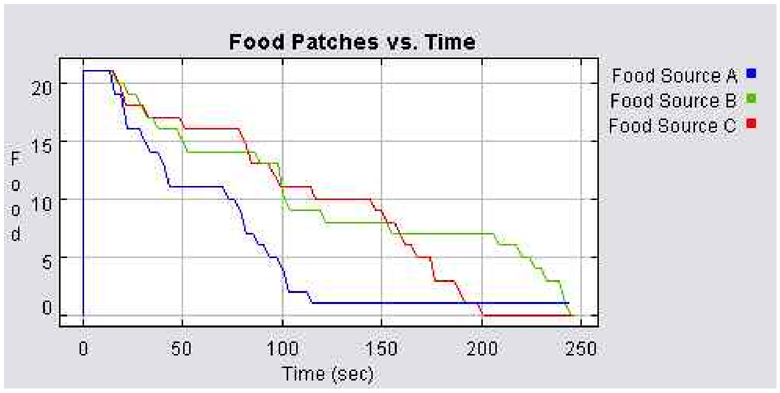


Figure 55. Time evolution of food remaining (N=100, d=50)

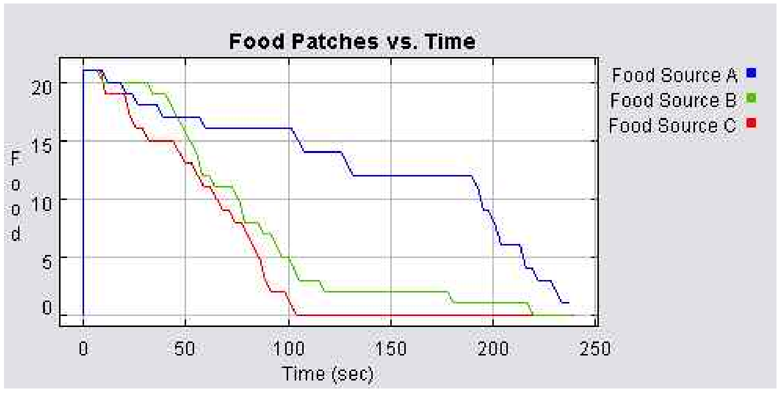


Figure 56. Time evolution of food remaining (N=100, d=40)

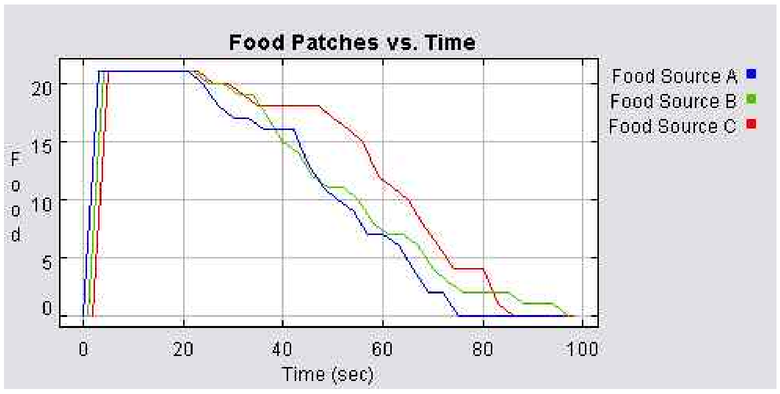


Figure 57. Time evolution of food remaining (N=100, d=25)

The pheromone decay rate, as shown by the results above, plays an important role in the group foraging strategy. Figure 58 summarizes these results, by showing the task completion time as a function of decay rate. There is a strong negative correlation between task efficiency and decay rate. As the decay rate increases, the group is forced to collectively choose to abandon one or two sources and focus on the other two or one sources, respectively. Note that the graph in Figure 58 is divided into regions corresponding to the three exploitation schemes.

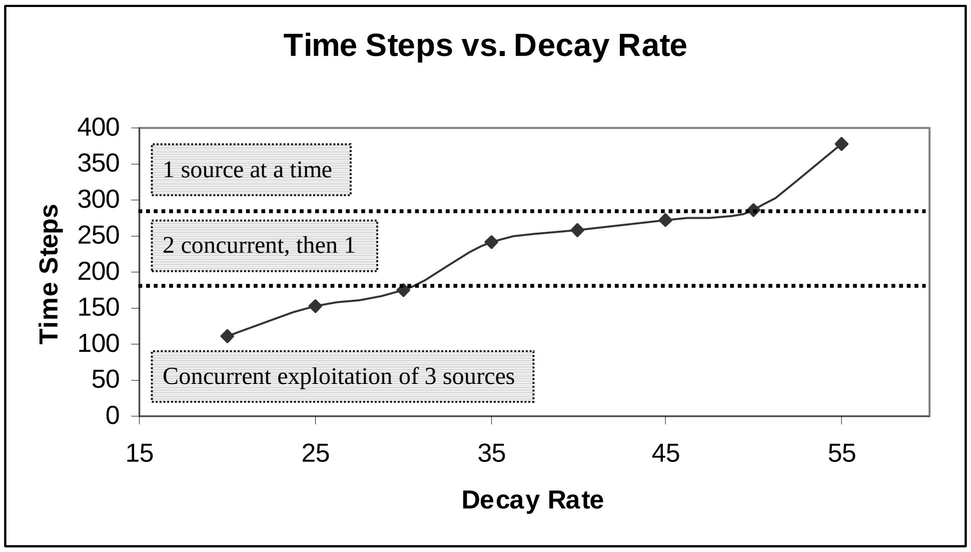


Figure 58. Completion time steps versus decay rate (N=100)

Subsection 5.5.4: Case 5

Case 5 features four identical food sources, equidistant from home. Each source is placed near a corner of the foraging field. This group of simulation trials was run to demonstrate the existence of several stable foraging states. The values of N (group size) and d (decay rate) were chosen to achieve a variety of exploitation schemes. This case is similar to Case 4, which included three sources, but with added complexity in the possible exploitation behaviors. A typical example is presented in the following successive images from one trial. Figure 59 reveals the initial exploitation of the two sources on the left side, with the dominant trail leading to the top left source. Notice that only incomplete trails lead to the other two sources. After depletion of the top left source, the dominant trail shifts to the lower left source, but the other two sources remain weakly exploited (Figure 60). In Figure 61, the lower left source has been completed and focus has shifted to the top right source. Finally, in Figure 62, the colony moves on to the one remaining source.

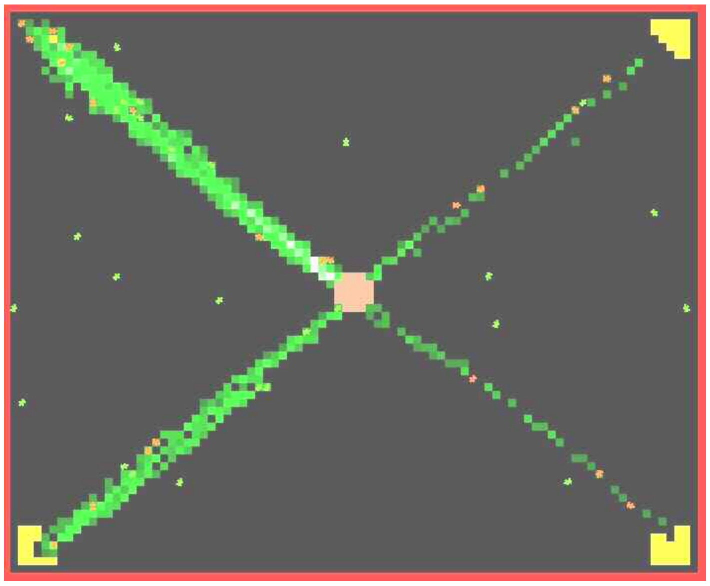


Figure 59. StarLogo simulation (N=50, d=20, t=50)

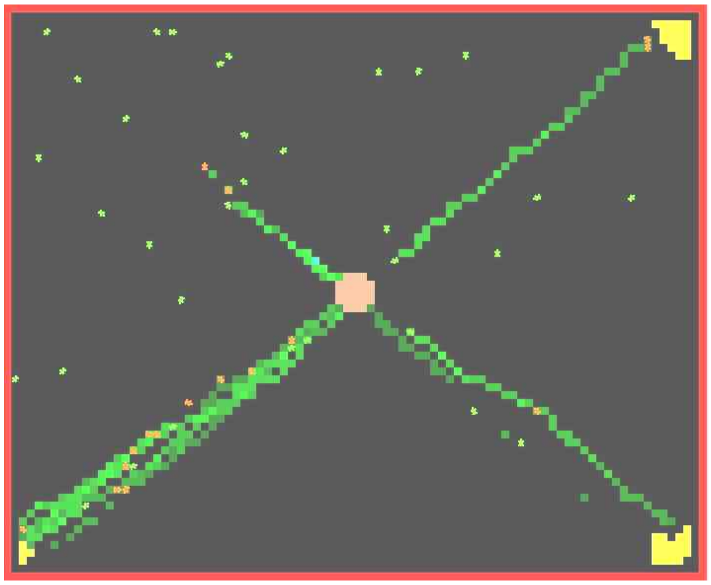


Figure 60. StarLogo simulation (N=50, d=20, t=127)

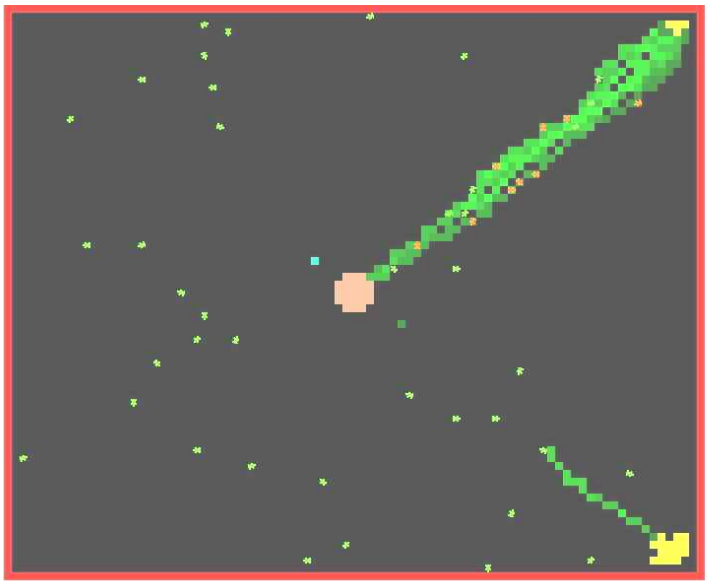


Figure 61. StarLogo simulation (N=50, d=20, t=230)

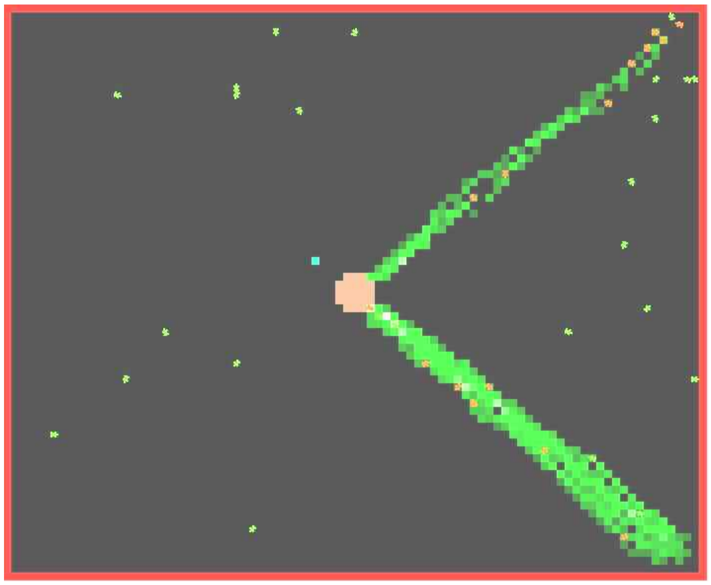


Figure 62. StarLogo simulation (N=50, d=20, t=302)

The following plots each represent a distinct foraging strategy observed in the simulation. These figures do not represent an exhaustive listing of the possible behaviors, as a great deal of complexity exists with four sources. This complexity arises from the possibility of switching between exploitation schemes during the course of one trial. The examples below only present some of the possible foraging patterns. The simplest cooperative behavior is shown in Figure 63, as all four sources are exploited at approximately the same rate.

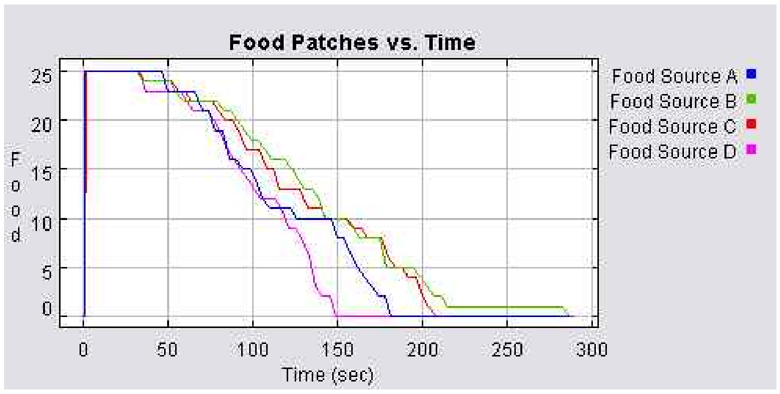


Figure 63. Time evolution of food remaining (N=70, d=15)

The following two figures demonstrate the concurrent exploitation of two (Figure 64) and three (Figure 65) food sources. Notice that, in Figure 65, Food Source C was not even discovered until long after the other three sources were depleted. With so many ants committed to foraging at Sources A, B, and D, very few remained to discover new sources. Source C was only discovered after many ants were finished with trail-following.

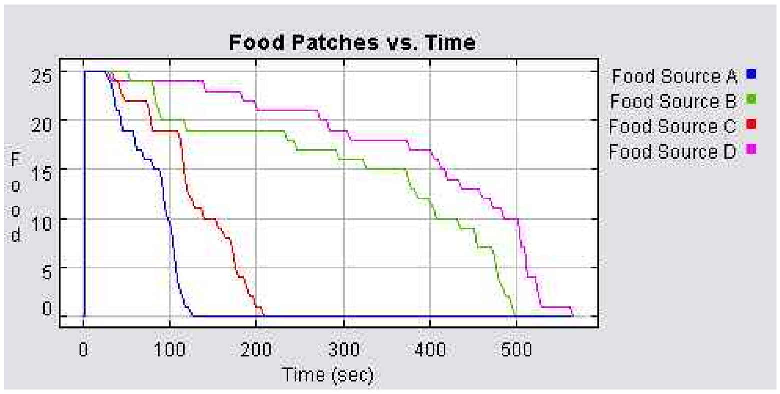


Figure 64. Time evolution of food remaining (N=70, d=20)

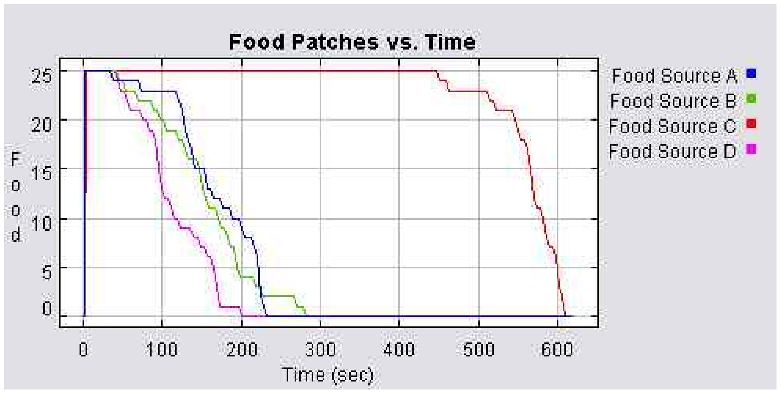


Figure 65. Time evolution of food remaining (N=50, d=15)

Figure 66 and Figure 67 reflect more complex examples of system behavior. In Figure 66, the colony discovers all four sources early in the trial, but soon abandons Food Source A. Sources B and C are moderately used, but the colony focuses on Source D. After approximately 400 time steps, Source D is completed, but the trails to Sources B and C evaporated completely. Meanwhile, positive reinforcement established a strong trail to Source A, for which the collection rate rapidly increased. Upon completion of Source A, the remaining two sources (B and C) were exploited concurrently.

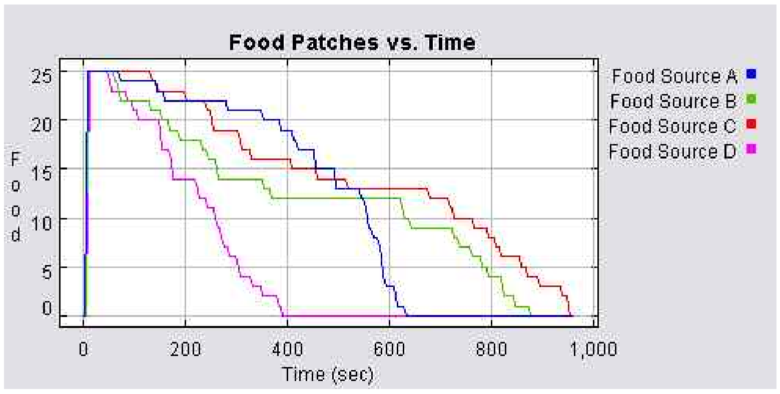


Figure 66. Time evolution of food remaining (N=30, d=15)

In Figure 67, the colony first collectively chooses Sources A and D, but the trail to Source D evaporates before completion. At approximately t = 700, a strong trail is established to Source B, which is quickly finished. Eventually, Source D is rediscovered and a trail is established to Source C, the only remaining source.

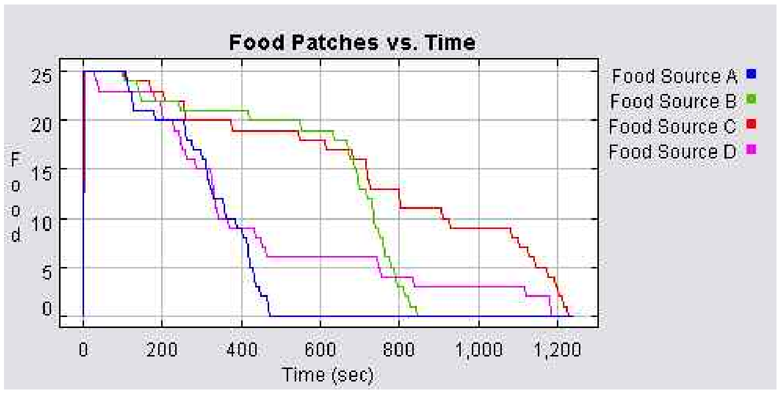


Figure 67. Time evolution of food remaining (N=30, d=20)

Clearly, when four food sources are present, the colony has the potential to utilize a number of exploitation schemes. While the system behavior shows some dependence on group size and decay rate, a number of states exist even for one set of system parameters. Running several trials for fixed values of N and d resulted in a variety of foraging patterns.

Chapter 6: Experimental Design

Section 6.1: Experimental Motivations and Goals

Given the theoretical models and simulation tools available to a swarm intelligence researcher, one may question the utility of experimentation with real robots. After all, physical implementation is more expensive and less conceptually enlightening than simulation. It is usually impractical, for a number of reasons, to assemble a robotic swarm for research purposes large than 10 to 20 robots. From a theoretical perspective, one could argue, there is little to gain from physical implementation. The motivations for physical implementation in this work arise in response to such notions. The five major motivations for experimentation with real robots are discussed in the following paragraphs.

The first motivating factor for this experimentation is the current lack of experimental examples in swarm intelligence literature. While theoretical modeling and simulations have generated a wealth of promising results and led to the imagination of productive real-world robotic applications, most researchers have stopped at this juncture. There are a few notable exceptions to this generalization, most of which are related to development for military application. In such an immature field of study, it is reasonable to lay a strong foundation in theory before forging ahead into baseless attempts at implementation. At this point, however, further theoretical study pales in importance without manifestation in experimentation. The field has matured to the point were further advancement requires a combined focus in both theory and practice. This experimental effort is an attempt to initiate just such a transition to greater balance in swarm intelligence research.

Secondly, the only experimental attempts to design functional robotic foraging systems currently present in the literature employ communication schemes other than trail-based foraging. A number of other methods have been used, with varying success. Most notably, beacons and tactile chains have enabled cooperative foraging. Other examples have succeeded in foraging and/or clustering tasks via stigmergic cooperation. Conspicuously absent is the utilization of trail-laying and following as the mechanism of cooperation. Given that the examples above draw direct inspiration from biological systems, including social insects, the absence of trails is a conspicuous void. One would expect that the predominant method employed by ants would be one of the first choices for a mode of communication. After all, the incredible efficiency of mass recruitment in ants is a testament to the efficacy of self-organization.

While this experimentation is motivated by the lack of trail-based experiments with real robots, as mentioned above, the third major motivation is the weaknesses of current trail-laying and following in tasks other than foraging. With limited success, a small number of researchers have attempted to develop robots capable of laying and following trails. None of these attempts have been motivated by robotic foraging, but they are the best examples of trails in robotics. Unfortunately, each of the proposed systems has important weaknesses, especially if applied to a foraging task. From high power consumption to slow chemical sensing to impractical trail material, each example has critical flaws. In order to develop a functional foraging system, it will be necessary to significantly improve the properties of the trail itself, as well as the deposition and sensing capabilities of individual robots.

Novel technological concepts often generate unrealistic expectations in terms of real-world application. This could be true in the world of robotics and AI more than any other technological field. Proclamations made in the media or even in respected published literature tend to overstate the extent, imminence, and value of potential real-world applications. This is certainly the case in the swarm intelligence literature. Claims are often made on the basis of theory and simulation, but these claims sometimes ignore the difficulties of physical implementation. The step from the simulated on-screen world to the real world is often a greater gap than expected. While it is admitted that swarm intelligence implementation tends to be fairly simple in terms of the technological requirements for each robot, system development remains a significant hurdle. At this point, the swarm intelligence community has largely avoided a thorough investigation of the difficulties of physical implementation. These difficulties can only be addressed as they are encountered by groups attempting physical realization.

One of the central tenets of swarm intelligence is simplicity. This is one of the main advantages proclaimed in the argument against traditional artificial intelligence. Swarm intelligence researchers contend that behavior-based robotic systems exhibit complex behavior, despite the simplicity of each individual. Few would argue against the simplicity of a subsumption architecture, which greatly reduces the processing requirements of the robot and the programming complexity. These claims of simplicity seem to indicate that a group of technologically rudimentary robots should be sufficient to exhibit swarm intelligence. In part, this experimental effort is motivated by an attempt to test the limits of the doctrine of simplicity.

The primary goals of experimentation follow directly from the five major motivations listed above. While each goal does not necessarily correspond to a single motivation, there is a general correspondence that should be obvious. The following four goals determined many of the subsequent choices regarding experimental design:

Experimental Goals

(1) Develop a system of robots capable of utilizing trail-laying and following in completion of a central-place foraging task.

(2) Introduce a novel trail substance that is non-toxic, time-decaying, and easily deposited and sensed.

(3) Demonstrate characteristics of a swarm-intelligent system [3]

   a. Emergent pattern formation

   b. Multistability

   c. Increased task efficiency due to cooperation

(4) Compare experimental task efficiency to theoretical predictions and simulation results.

Section 6.2: LEGO® Mindstorms™

In 1998, LEGO introduced a revolutionary product in the arena of educational toys--Robotics Invention System (RIS). This robotics kit was developed as the flagship product of a new Mindstorms brand name and released with the intended primary audience of children. Within a few months of the release, however, it became apparent

[3] Note that there is not an attempt to demonstrate bifurcation phenomena, due to the difficulties of demonstrating this with few robots and on short time scales.

that the RIS appealed to a far broader audience than LEGO had anticipated. Hobbyists, robotics enthusiasts, engineers, and researchers in academia became interested in the product on personal and professional levels. Just over a year later, LEGO introduced RIS, version 1.5, which was quickly followed in 2001 by RIS 2.0. Each kit, which retails for approximately $200, contains over 700 individual pieces, including a variety of standard LEGO bricks, TECHNICS parts, two motors, two touch sensors, a light sensor, and the RCX (see Figure 68 below). These sensors are encased in fully functional LEGO bricks and designed to interface seamlessly with the RCX "programmable brick." The RCX brick functions as the on-board computer for custom robotic creations. It is a programmable, micro-controller based brick capable of simultaneously operating three motors, three sensors, and one infrared (IR) serial communications interface. When a consumer obtains this kit, all that is required is a personal computer to create (and program) an autonomous mobile robot. LEGO provides software for installing a graphical, icon-based programming environment known as RCX Code. This kit also includes an IR transmitter required to download programs from the PC to the RCX.

LEGO
MINDSTORMS
ROBOTICS
INVENTION
SYSTEM
Ages 12+
Cont. 718 PCS
CREATE THOUSANDS OF ROBOTIC INVENTIONS
BUILD & PROGRAM ROBOTS THAT DO WHAT YOU WANT!
BUILD
PROGRAM
RUN

Figure 68. LEGO Robotics Invention System (RIS) 2.0

The system architecture developed by LEGO facilitates ease-of-use that even a child can master. At the lowest level is the processor, a Hitachi H8300, which executes machine code instructions. The processor cooperates with additional components that convert signals from the ports into digital data, using chips providing memory for data and program storage. Above the processor and circuit layer is the ROM code, which provides functionality to the basic input/output system (BIOS) of the RCX, allowing it to startup and interface with peripherals. On top of the ROM code layer runs the firmware, which functions as the operating system of the RCX. The firmware is downloaded and "installed" into the RAM of the RCX when the user purchases the product. The topmost layer is the user-created RCX Code defining instructions for a particular program. The software provided by LEGO translates the program into bytecode before sending it to the RCX, and finally the bytecode is interpreted by the firmware and converted to machine code. The RCX processor executes the machine code to run the program. Refer to Table 6 for a summary of the RCX specifications.

Table 6. Technical Specifications for LEGO RCX [111]

Processor	Hitachi H8
RAM	32 Kb
ROM	16 Kb
I/O	3 sensors / 3 actuators
Communication	IR transceiver

The RCX represented an important innovation in the world of custom and amateur robotics. By linking sensing and actuation so tightly with the mechanical architecture of a robot, the RCX completed an excellent platform for robotics education and investigation. The RCX was inspired by the MIT Programmable Brick developed in MIT's Media Lab in the mid-1990s [112], which received sponsorship from LEGO. The

RCX was an independent creation, however, designed "from the ground up" and in no way based on the technical elements of MIT's brick. Through ingenious reverse engineering, the community of Mindstorms addicts came to fully understand the internals of the RCX [111] (see Figure 69 for an external view of the RCX). This understanding quickly led to the introduction of custom sensors (see [29], and references therein), actuators, and, most importantly, alternatives for programming the RCX. From alternative programming languages to firmware replacement, the full capabilities of the RCX were unlocked as a number of individuals created these alternatives to the restrictive RCX Code. For a full description of these alternatives, see [29].

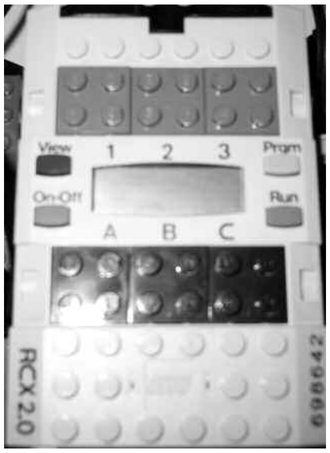


Figure 69. LEGO RCX 2.0

One of the most popular alternatives to the software provided by LEGO is Bricx Command Center, or BricxCC, an integrated development environment (IDE) that fully replaces the functionality of the software provided by LEGO while enhancing programming, debugging, and testing capabilities. BricxCC was developed by John Hansen and Mark Overmars as a complete program development interface between a PC and the RCX [113]. Refer to Appendix C for BricxCC documentation. BricxCC supports a variety of alternative programming languages to RCX Code.

Despite the simplicity of LEGO’s RCX Code, it has several major drawbacks as a programming language:

- The set of instructions is very limited, and doesn’t disclose the full power of the RCX processor.
- The graphical interface is unsuitable for large programs and becomes unmanageable.
- RCX Code severely limits the resolution of sensor readings (especially for the light sensor).
- Functions, subroutines, and arrays cannot be created.
- The use of variables, counters, and timers is limited.

Because of these important limitations, most advanced users turn to one of several alternative languages. For this research, NQC (Not Quite C) was chosen as the best alternative. Developed by Dave Baum, NQC is a streamlined version of C, augmented with a variety of macros and commands specific to programming the RCX [114]. The advantages of NQC include the following:

- It is based on the original LEGO firmware, ensuring reliability.
- It is multiplatform on the host side (PC, Mac, Linux) and target side (supports all LEGO programmable bricks: RCS, Scout, Cybermaster)
- It is supported by a number of IDEs, including BricxCC
- NQC is free software released under the Mozilla Public License.

While NQC closely resembles C in terms of syntax, there are some important differences. Refer to [114] or the NQC documentation in Appendix D for more detailed information.

Section 6.3: Disappearing Ink

In evaluating candidate substances for a trail, there were several important considerations. The most important factor was the property of time decaying trail "concentration," similar to pheromone evaporation in ant foraging. This property is necessary in a fully functional foraging system, as it provides negative feedback to the system. Without a time-decaying substance, an established trail would remain constant even after depletion of a food source (or a cluster of targets). While this robotic implementation is not intended for a specific real-world application, certain other trail attributes are desirable:

- Non-toxic
- Non-damaging to common indoor flooring materials
- Imperceptible after complete trail decay
- Inexpensive
- Chemically inert and nonflammable

Of course, two other major considerations are the ease with which the substance can be deposited and perceived by individual robots.

An original solution to the problem of choosing a trail substance is "disappearing ink." Commonly found as a toy or a novelty item, "disappearing ink" actually possesses many of the properties listed above. Of course, "disappearing ink" is known for its characteristic ability to fade from a dark blue color to colorless in a matter of seconds or minutes. In addition, the ink is non-toxic in terms of contact with human skin. It does not damage or stain a variety of hard floor materials. Carpet would not be a feasible substrate for a number of reasons, making the issue of carpet damage irrelevant. It is quite inexpensive, as it is composed of three very common chemicals [115]. The only drawback to this substance is the issue of flammability. The chemicals in this ink are flammable, making it unsuitable for some applications. However, for a wide range of applications, this would not be a problem. Disappearing ink is not highly reactive, but it would react with a number of chemicals. Again, this would likely not be a factor in a wide array of real-world applications.

The chemistry involved in disappearing ink is quite simple, which is another favorable attribute of this substance. Only three elements are required to make the ink: a pH indicator, an alcohol, and a base. These elements combine to create the disappearing behavior of the ink. The most common formulations of disappearing ink utilize the following specific chemicals:

- thymolphthalein indicator, $C_{28}H_{30}O_4$
- ethanol (ethyl alcohol), C_2H_6O
- sodium hydroxide, NaOH

This particular pH indicator is dark blue in a moderate to strong basic solution ($pH > 10$) and colorless in solutions with $pH < 9.3$ [115]. These three chemicals are combined to form a basic solution (due to the addition of NaOH). When the ink is spread thin on a piece of paper or cloth, the increased surface area accelerates the following chemical reaction:

$$2\ NaOH + CO_2 \rightarrow Na_2CO_3 + H_2O$$

As sodium hydroxide is consumed by this reaction with carbon dioxide in the air, the pH of the solution decreases. The pH indicator, thymolphthalein, reflects this change in pH by steadily changing from blue to colorless. The color change can be accelerated by the addition of an acid to the solution. By varying the initial concentration of NaOH, the time required to transition from blue to colorless can also be tailored.

For the purposes of this experiment, the desired time interval between ink application and complete disappearance was on the order of several minutes. Through trial and error, the following process achieved the appropriate initial NaOH concentration to produce a complete decay time in the range of 2-8 minutes:

Step 1: Add 20 ml indicator/ethanol solution (0.05% w/v) to small beaker.

Step 2: Add 12-25 drops NaOH aqueous solution (50% w/w) to produce the desired decay rate.

Step 3: Add approx. 0.75 grams solid granular thymolphthalein indicator to obtain darker shade of blue.

The last step in the process allows one to adjust the initial shade of blue without altering the time required for color change. In order for the trail to be dark enough to sense using the LEGO light sensor, it was necessary to increase the concentration of indicator using this additional step. In Step 2, note that the range of 12-25 drops of sodium hydroxide corresponds to a complete disappearance time of 2-6 minutes.

The ink solution was tested on a number of surfaces, with a final choice of poster-board as the substrate. The non-glossy paper surface provides enough ink absorption to prevent the robot wheels from tracking ink across the foraging field. It also prevents the ink from drying prior to color change, which can occur with more absorbent materials. An unexpected benefit to the use of disappearing ink is trail reinforcement. When a new trail is deposited across an older trail (even if already colorless), the overlapping region is darker than the rest of the trail. This region also takes slightly longer to disappear, creating the effect of increased "pheromone concentration" along high-traffic routes.

Section 6.4: Mechanical Design

The mechanical design of LEGO Mindstorms robots is considered by some to be more of an art than a science [116], which could be said of mechanical design in general. A number of books have been written for the single purpose of aiding LEGO designers (e.g., see [29]). The thriving online Mindstorms community provides a wealth of information for anyone intending to design a LEGO robot, which is of course considerably different from traditional robot design. Fortunately, designing in LEGO has certain distinct advantages over the use of traditional mechanical components:

- LEGO is fast – From design concept to finished product, creation of a basic model takes only a matter of hours.
- LEGO is cost-effective – The basic RIS 2.0 kit costs only $200, which enables one to create robots of moderate complexity. Additional parts are readily available and fairly inexpensive compared to traditional robotics components.
- LEGO is flexible – Redesigns and modifications are relatively easy to perform, given the ease of disassembly.

The main mechanical disadvantage to using LEGO Mindstorms is reliability and structural integrity. Without taking special care to reinforce each mechanical subsystem, LEGO parts can detach from one another due to vibration and reaction forces applied to motors. The following subsections describe the design requirements, provide sensor and motor information, and detail the final robot design. Finally, a few design challenges encountered during the design process are addressed.

Subsection 6.4.1: Design Requirements

The first step in the engineering design of a mechanical system is to perform functional decomposition of the expected task and generate a set of functional requirements [117]. In this case, the task is collective foraging, which can be subdivided into seven subtasks:

(1) Searching for targets

(2) Obstacle avoidance

(3) Object retrieval

(4) Homing

(5) Object release

(6) Trail following

(7) Trail deposition

In order to complete each task, a number of functional requirements are needed from the final design. Some of the functional requirements are necessary to complete more than one subtask, while others are coupled. For example, the subtask of trail deposition must occur immediately after object retrieval. To complicate the issue, the design is limited to three motors and three sensors due to the input/output capabilities of the RCX. From this list of subtasks, a preliminary set of functional requirements is obtained, as shown in Table 7.

Table 7. Generation of Functional Requirements for a Foraging Robot

Subtask	Functional Requirement(s)
search for targets	▪ locomotion ▪ ability to sense target
obstacle avoidance	▪ ability to sense obstacle or collision ▪ locomotion ▪ turning
object retrieval	▪ method of picking up target
homing	▪ ability to know relative position of home
object release	▪ method of dropping target
trail following	▪ locomotion ▪ ability to sense trail
trail deposition	▪ locomotion ▪ mechanism of depositing ink

The preliminary list of functional requirements (Table 8), after eliminating duplicates, is as below:

Table 8. Preliminary Functional Requirements

locomotion
turning
obstacle or collision sensing
target sensing
pick up target
release target
trail sensing
trail deposition
knowledge of home location

This list can be refined by consideration of certain design constraints. As mentioned above, one of the major constraints is the limitation of three motors and three sensors. Another self-imposed constraint relates to the turning requirement. While none of the subtasks require rotating (turning in place without translation), the maneuverability afforded by this ability should be considered essential. Performing obstacle avoidance in a system of autonomous robots would be extremely difficult without the ability to rotate in place. Yet another self-imposed constraint applies to the functional requirement of knowing the location of home. In reality, the robot is only required to know its position relative to home. There are four main solutions to this problem of mobile robot positioning: (1) odometry and dead-reckoning, (2) beacon referencing, (3) landmark-based navigation, and (4) map-based positioning. Dead-reckoning is notoriously difficult due to accumulated odometry error. Odometry would also require the use of rotation sensors, which would violate the constraint of using only three sensors. Map-based positioning requires a priori knowledge of the foraging field and sophisticated on-board representations of the environment, which is impossible using LEGO robots and violates the philosophy of a swarm-based system. Landmark-based navigation could be useful, except for the lack of physical landmarks in the experimental environment and the need for some type of robot vision. Having eliminated all other options, beacon referencing becomes the obvious solution. With these constraints in place, the final list of functional requirements takes shape (

Table 9):

Table 9. Functional Requirements

Actuation	**locomotion**
	rotation
	pick up target
	release target
	trail deposition
Sensing	obstacle/collision sensing
	target sensing
	trail sensing
	home beacon sensing

Having separated the functional requirements into the categories of sensing and actuation, the input/output limitations of the RCX become problematic. Four functional requirements require sensors, while only three input ports are available. Similarly, five requirements require motor actuation, but only three outputs are present. For this reason, the functional requirements become coupled, introducing added complexity to the mechanical design process. In order to solve this design problem, it was necessary to utilize a single light sensor to satisfy two functional requirements (target sensing and trail sensing). One motor was used to satisfy three functional requirements (pick up target, release target, trail deposition). The details of the design are found in Subsections 6.4.3 and 6.4.4, but Table 10 illustrates resource (sensors and motors) allocation according to functional requirement.

Table 10. Allocation of Motors and Sensors

Actuation	**two motors**	**locomotion**
		rotation
	one motor	pick up target
		release target
		trail deposition
Sensing	touch sensor	obstacle/collision sensing
	light sensor	target sensing
		trail sensing
	light sensor	home beacon sensing

Subsection 6.4.2: LEGO Sensors and Motors

In this section, a brief technical description is provided for the LEGO sensors and motors. While a number of websites sell or provide instructions for building custom LEGO-compatible sensors, this design only utilizes Mindstorms brand motors and sensors.

LEGO currently produces three types of 9V DC electric motors that are compatible with the RCX: the ungeared motor, the geared motor (Figure 70), and the micromotor. The geared motor is a newer product and has several advantages over the other two models (Table 11). It features low current consumption, low RPM due to an internal multistage gear reduction, much higher efficiency than the ungeared motor, and far more torque and higher reliability than the micromotor. The 9V geared LEGO motor is extremely reliable and fairly resistant to damage under stall situations [29].

Table 11. Properties of LEGO Motors (taken from [29], p. 44)

Properties	Ungeared Motor	Geared Motor	Micromotor
Maximum voltage	9V DC	9V DC	9V DC
Minimum current (no load)	100 mA	10 mA	5 mA
Maximum current (stall)	450 mA	250 mA	90 mA
Maximum speed (no load)	4000 rpm	350 rpm	30 rpm
Speed under typical load	2500 rpm	200 rpm	25 rpm

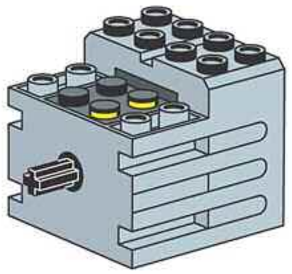


Figure 70. LEGO 9V geared motor (taken from [118])

Two types of LEGO sensors are used in this design: light and touch (see Figure 71). The touch sensor is a very simple mechanical contact that opens or closes a circuit internal to the RCX. It has a binary state, *on* or *off*, which is read by the RCX. The LEGO light sensor is more complicated, but still a very basic sensor compared to conventional sensing hardware. Individuals have succeeded in reverse engineering the light sensor and discovering the internal circuitry (refer to [119] for details). The main elements of the sensor are a phototransistor for sensing light intensity and a red LED, which allows the sensor to measure reflected light as well. The phototransistor is also sensitive to light in the IR spectrum. The sensor can be employed to sense light intensity, color, or relative distance to an object [29]. The RCX reads a raw value between 0 and 1023, depending on the intensity of light incident on the phototransistor.

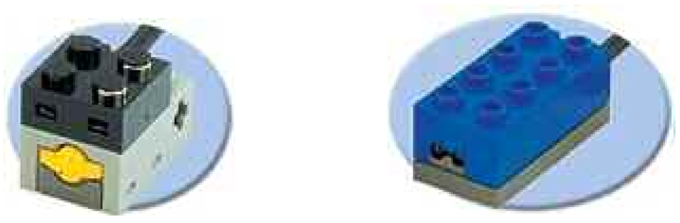


Figure 71. LEGO touch sensor (left) and light sensor (right); (taken from [118])

Subsection 6.4.3: Physical Architecture

The system architecture of the completed design consists of several subsystems. These subsystems combine to form the final product, which is a robot satisfying all of the functional requirements listed in

Table 9 (see Figure 72 for photographs of the completed robot). Each subsystem is described below, along with its relationship to the entire system architecture.

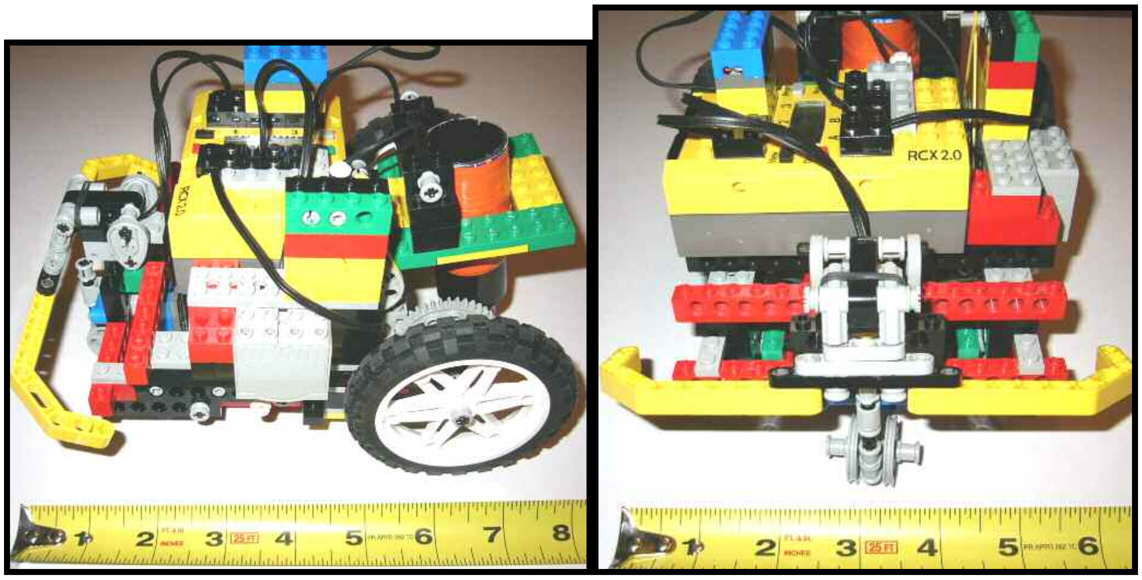


Figure 72. Side (left) and front (right)views of final robot design

Prior to designing the mechanical architecture of the robot, it was necessary to first decide on a system of locomotion. As described in Chapter 2, there are a variety of available options for the drive system of an autonomous mobile robot. Each option has distinct advantages and disadvantages. One of the functional requirements is rotation (without translation), which immediately eliminates certain drive systems (steering, tricycle, articulated, and pivot drives). Of the remaining four options (differential, dual differential, synchro, and skid-steer), the differential drive was selected due to simplicity of construction, the maintenance of an open undercarriage (not allowed by the synchro drive), and a consistent axis of rotation (unreliable in a skid-steer drive). The selection of a differential drive to some extent dictated the basic form of the robot. As with all differential drive arrangements, the driving wheels are arranged on opposite lateral sides of the robot and driven independently. One or more caster wheels at the front or rear of the robot provide stability and free rotation.

After choosing a differential drive, the robot system architecture begins to take shape. The first subsystem provides the foundation upon which all others are built: the chassis (Figure 73). The robot chassis is a generally rectangular framework of beams providing the structural integrity of the robot. The chassis was designed to withstand the forces caused by collisions and those imposed as reaction forces on the motors. It was also designed to resist internal detachment due to vibrations caused by ground contact and motor operation. Redundant bracing and double-width longitudinal beams result in a firm foundation for additional subsystems.

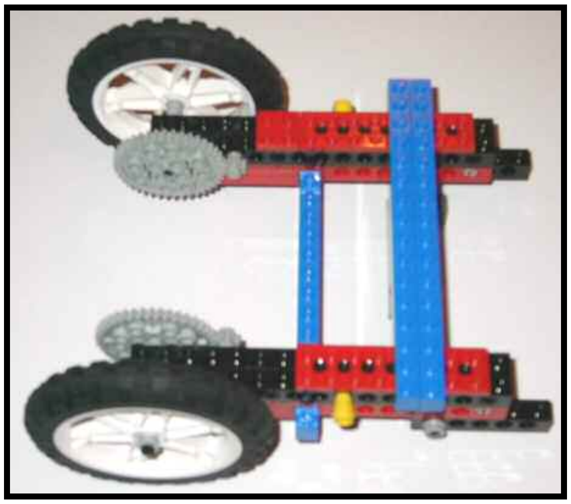


Figure 73. Robot chassis (with rear wheels attached)

The chassis is followed by the drive subsystem, which includes three basic elements: (1) a caster wheel, (2) two rear drive wheels, and (3) two motors framed together (Figure 74). The caster wheel, located at the front of the robot, connects modularly into the main front crossbeam of the chassis. It is composed of two small wheels (without tires) coupled on one axle and oriented symmetrically with respect to a vertical axle. The vertical axle serves as the axis of rotation for the caster during turns. Each rear drive wheel is covered with a rubber tire. The axle for each wheel passes through a double-beam of the chassis and ends at a 48-tooth drive gear. The two drive motors are braced together, back-to-back, in a frame solidly containing both motors. This frame is connected to the chassis at several locations. Each motor has an 8-tooth gear on the output shaft that drives the 48-tooth gear on each rear-wheel axle.

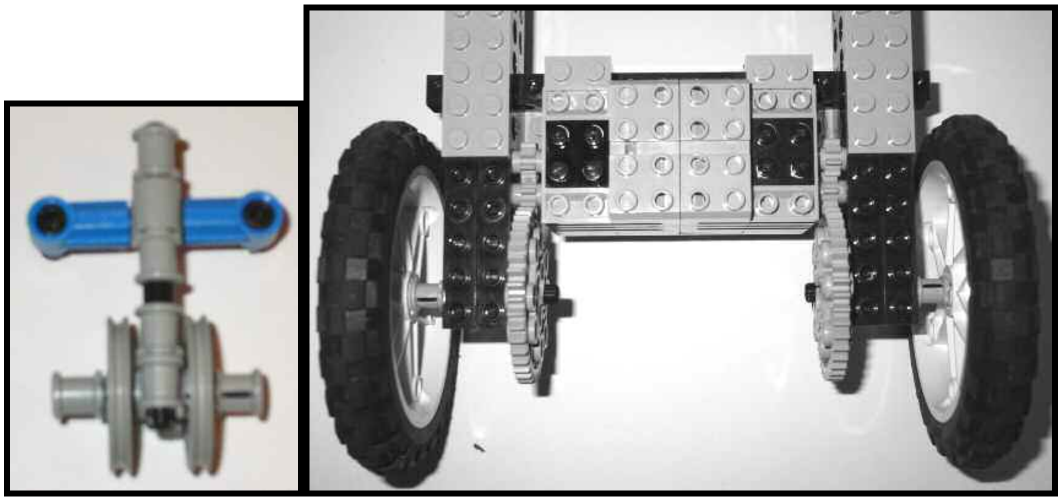


Figure 74. Caster wheel (left) and drive system (right)

The next subsystem is the front bumper assembly (Figure 75), which attaches directly to two horizontal crossbeams on the front of the chassis. The bumper assembly consists of one touch sensor that is closed upon contact with an obstacle. Two long, angled lift-arms extend laterally from the center of the robot. These lift-arms rotate about a horizontal axis in order to depress the touch sensor contact point. A rubber band maintains the "open" state of the touch sensor by slightly rotating the lift-arms off the contact.

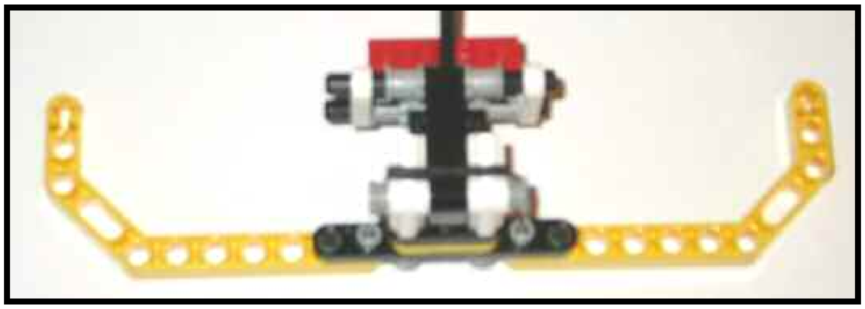


Figure 75. Front bumper subassembly

At the rear of the robot is the pen holder and drive subsystem (Figure 76), which controls the ink deposition process. A long drive axle is mounted laterally across the two main drive motors, with a large pulley on the end of the axle. The axle is supported at two points and generates the rotation of the pen holder frame, which is connected to the axle via short lift-arms. The square pen holder frame has an opening in the center for marker insertion and an elevated axle from which a rubber band connects to the RCX. This band ensures that the default position of the pen holder is up (not depositing a trail). The drive pulley connects by a rubber belt to a motor mounted on the right side (facing the front of the robot) and near the center (front-to-back) of the robot. The motor is mounted and braced solidly to the chassis, preventing dislodgement.

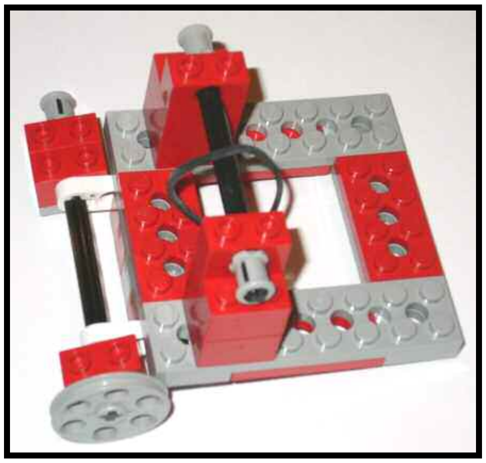


Figure 76. Pen holder and drive assembly

The mechanism responsible for capture and release of targets is least connected to the chassis. A permanent magnet adhered to two LEGO racks (flat gears) is held in tension by opposing rubber bands anchored on opposite side of the chassis (Figure 77). The racks are driven by an 8-tooth gear on a laterally-oriented axle passing through the chassis. This axle is driven by a rubber belt from the third (non-drive-system) motor mounted directly above the end of the axle. The magnet assembly is guided in a vertical slot constructed using small LEGO bricks and short axles connected to the underside of the RCX. A large plate, on which the targets lay flat during magnetic suspension, covers the magnet apparatus and is connected on four corners to the chassis (Figure 78). A downward-facing light sensor connects to the chassis immediately in front of the flat plate.

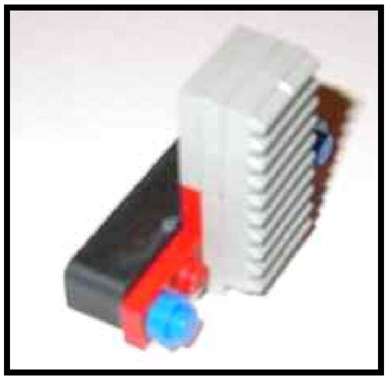


Figure 77. Magnet apparatus

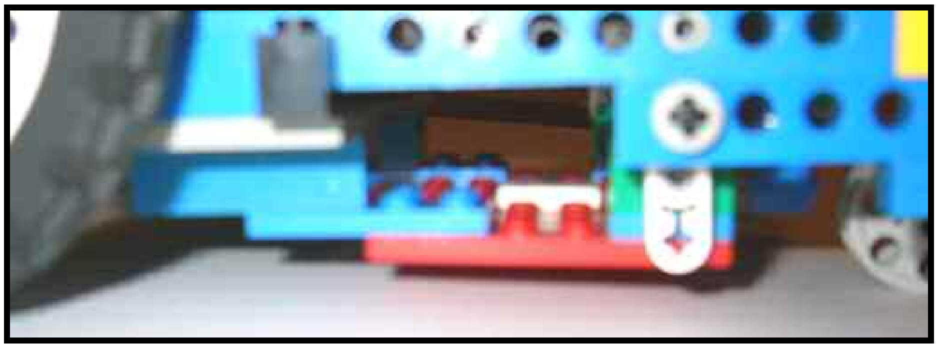


Figure 78. View of robot undercarriage (note the red cover plate)

Finally, the last subsystem is the RCX, to which all motors and sensors are wired (Figure 79). A forward-looking light sensor is mounted on top of the RCX, along with an anchor point to which the pen-holder rubber band attaches. The RCX provides support to the third (non-drive-system) motor, as well as the main pen-drive axle.

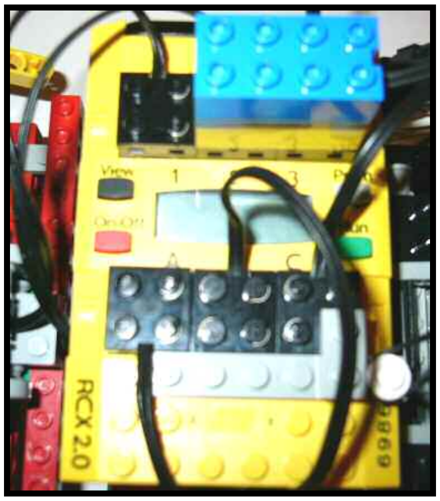


Figure 79. RCX Subassembly

Subsection 6.4.4: Assembly

This section describes the assembly procedure followed to build each of the seven individual robots. Refer to Appendix F for the specific steps required to assemble each subsystem described in the previous section. The integration of the subsystems into a completed robot is the focus of this section. Robot assembly can be divided into 6 major steps, as listed below:

Step 1: Assemble each individual subsystem (see Appendix F), except for the target capture/release mechanism, which cannot be fully assembled ahead of time.

Step 2: Connect the drive subsystem (caster, drive wheels, drive motors) to the chassis (Figure 80).

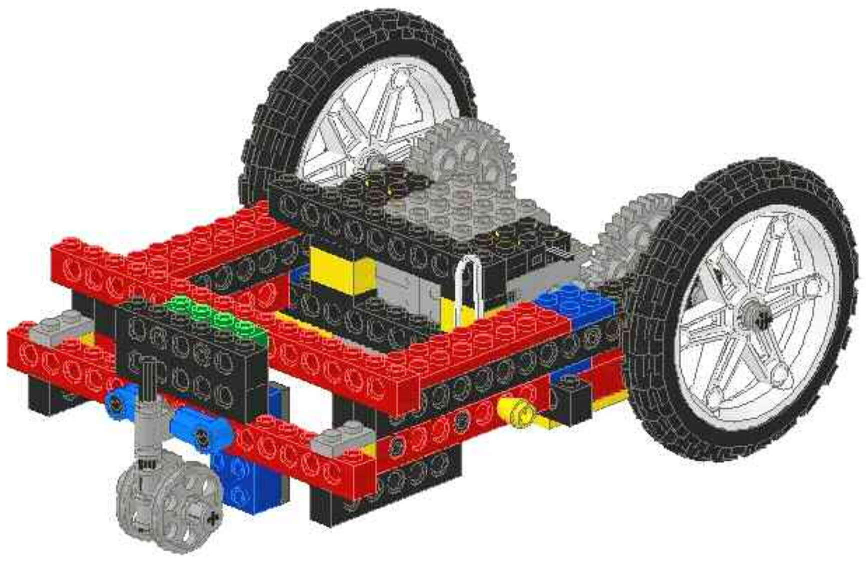


Figure 80. Chassis, caster, rear wheels, and drive motors

Step 3: Attach the third motor to the chassis, followed by the pen holder and drive mechanism. The rubber belt must be included between the motor output shaft and the pen drive pulley. The second rubber belt (that connects to the magnet apparatus drive axle) should also be included on the motor shaft.

Step 4: Attach the front bumper assembly to the chassis.

Step 5: Attach the RCX subsystem to the chassis and connect all wires. Connect the rubber band from the RCX anchor point to the suspended pen holder axle.

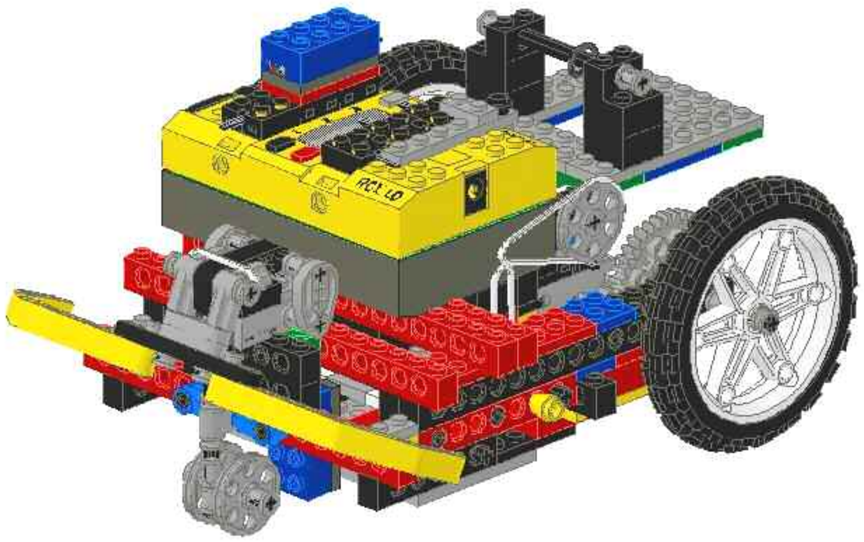


Figure 81. Pen holder, bumber, and RCX subsystems added to assembly

Step 6: Construct the target capture/release mechanism onto the underside of the RCX, including the opposing rubber bands anchored at points on the chassis (Figure 82). Connect the rubber belt already on the third motor to the pulley on the magnet drive axle. Attach the downward-facing light sensor to the chassis. Then, attach the flat cover plate to the chassis.

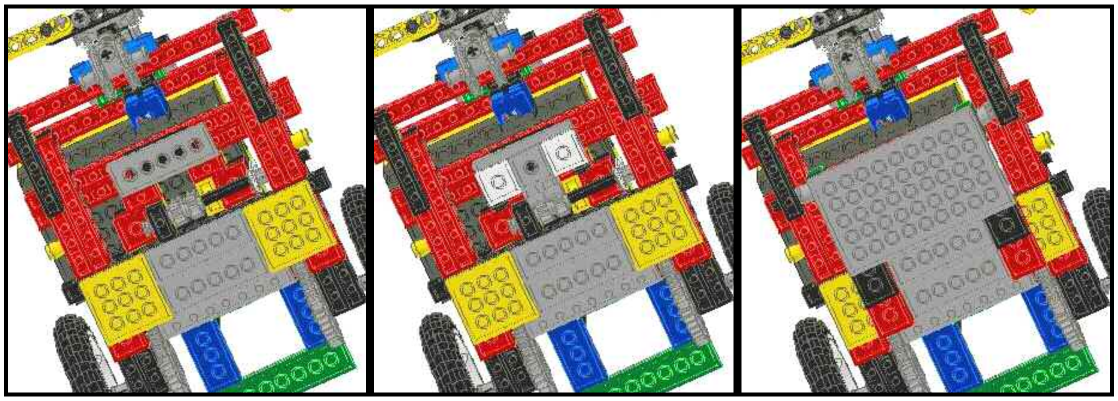


Figure 82. Steps for adding magnet apparatus and cover plate

Subsection 6.4.5: Operation

Mechanical operation of the robot can be described in the context of each of the "active" subsystems (excludes the chassis and RCX). The remaining four subsystems contain at least one mechanism with moving parts. The operation of these subsystems is described below:

- Drive System – The differential drive consists of two large, driven rear wheels and one front caster wheel. During movement in a straight line (forward or reverse), both drive motors operate in the same direction and at the same speed. In this case the caster does not rotate about the vertical support axle, but simply provides stability. Turning is accomplished by disparity in the power applied to the drive motors. Varying this disparity alters the turning radius. Rotation in place about the midpoint of the rear drive axles occurs when the rear wheels are driven with equal power in opposite directions. The caster wheel freely rotates to one side or the other to allow for this rotation. By braking one motor and powering the other, rotation can also occur about either rear drive wheel.
- Front Bumper – When the robot makes frontal contact with an obstacle of appropriate height, the lift-arms are caused to rotate about the pivot axle. After a small angular displacement, the touch sensor contact is depressed. When the robot moves away from the obstacle, the lift-arms are pulled to their original position by the rubber band in the bumper assembly. The touch sensor button returns to its normal position.

- Pen Holder/Drive – The third motor is connected to the pen drive via a belt and pulley system. In order to deposit an ink trail, this motor drives the large pulley, causing the pen holder frame to rotate with the axle toward the ground. Contact with the ground provides sufficient resistance to cause the belt to slip on the motor pulley. The result is constant, even pressure between the marker tip and the floor. The motor reverses direction to lift the marker tip from the ground. It is held in this “up” position by the rubber band anchored to the RCX, allowing the motor to turn off.
- Object Capture/Release – In order to pick up a metal washer, the third motor mounted above the main drive axle turns a belt connected to the axle pulley. The 8-tooth gears on this same axle interface with the racks on the magnet apparatus, generating linear vertical motion toward the floor. The magnet apparatus is stopped by the cover plate, causing the belt to slip on the motor pulley. A metal washer is suspended on the underside of the cover plate due to forces applied by the magnetic field. To release the washer, the motor turns in the opposite direction, moving the magnet upward and away from the cover plate. This reduces the strength of the magnetic field and allows the washer to drop to the ground.

Subsection 6.4.6: Design Challenges and Solutions

A number of significant design challenges were encountered during the development of this robot. These difficulties tested the limits of the imagination and the capabilities of LEGO Mindstorms robotics. In some cases, the eventual solution was sub-optimal, but could not be improved upon with the available materials and financial limitations. Each of these challenges, although presented individually, actually formed an interdependent set of design problems. The solution to one problem often rendered another unsolvable, necessitating an iterative process to solve all of the problems in a cohesive and functional design.

The most pervasive design challenge was the issue of robot size. Given a limited space in which to perform experimental testing, compactness became a design necessity. The length of the robot became the most important issue, as the width is largely determined by the width of the RCX and two motors. A robot of great length encounters maneuverability problems and suffers from inefficient obstacle avoidance due to the large radius swept by the front end of the robot. This problem was solved solely through design ingenuity and multiple design iterations. The vertical nature of the target release mechanism also proved invaluable for conserving length.

The caster design also presented a number of challenges. First of all, the most efficient caster designs contain two wheels capable of independent rotation [29]. This allows the caster to turn freely without wheel slippage. However, designing a caster of this type forces the caster to sweep through a circle of greater diameter during turns. This causes clearance issues and increases the overall length of the robot. Casters also rotate more freely under lighter loads. Unfortunately, creating a lighter burden for the caster by moving it forward again corresponds to increased length of the robot. The final design represents a compromise between robot length and weight distribution. The suboptimal caster design, with two wheels coupled on one axle, was also chosen for the sake of compactness.

The constraint of having only three available motors created a significant design challenge. As mentioned in reference to Table 10 in Subsection 6.4.1, one motor was allocated necessarily for three functional requirements. A single motor needed to pick up an object, release an object, deposit an ink trail, and stop ink deposition. Furthermore, each pair of these tasks is coupled, as they need to occur simultaneously. Ink deposition must begin at the same time an object is picked up. The solution to this problem involved the creative implementation of belt and pulley systems. Allowing belts to slip acts as a torque-limiter, preventing motor damage in stall situations. Rubber bands were also utilized to set default positions for the pen holder and magnet apparatus.

Releasing the metal washer (target) upon arrival at “home” presented yet another design challenge. Using a magnet to pick up the washer proved to be a simple mechanism to design. However, forcing the washer permanently out of the influence of the magnetic field is far more difficult. Initial attempts to solve this problem were hindered by the strong forces required for initial separation between the magnet and the washer. It later became apparent that direct contact between the washer and magnet was unnecessary. By introducing a space between the magnet and washer, and by moving the magnet away from the washer rather than vice versa, this problem was eventually solved.

The last major challenge was the ink deposition mechanism. In order for each robot to deposit a trail of ink capable of being followed, the line needed to be at least two centimeters wide. Furthermore, the trail needed to be fairly uniform in terms of width and ink concentration. It was essential to develop a deposition mechanism that did not leave “pools” of ink on the surface, even if a robot was momentarily stationary during homing behavior. This problem was solved by using a common, store-bought marker. The ink was removed from a large washable-ink marker, which was then modified slightly to fit into the LEGO pen holder. The original marker tip served as an excellent ink repository. The tip absorbed plenty of ink for an entire trial, and deposited the ink with excellent uniformity. A belt drive was used to apply constant pressure between the marker tip and the floor. The belt slips on the pulley at a set amount of torque, preventing the motor from stalling, but maintaining pressure on the marker.

Section 6.5: Behavior-based Programming

Subsection 6.5.1: Control Architecture Overview

A subsumption architecture was employed as the overarching control structure of the robot. This is the signature architecture of behavior-based programming, as popularized by Brooks [9, 10, 18] and Mataric [27, 61] (see Subsection 2.3.1). Subsumption architecture is based on a system of layered *levels of competence*, constructed in sequence from the simplest to most complex. Each competence level forms a complete and fully-functional system of control. These layers are stacked in parallel, with each higher level subsuming the lower levels. The layers are composed of a small set of *basis behaviors*, a term introduced in [61]. While each basis behavior is mentioned briefly here, a more thorough description of each is provided below in

Subsection 6.5.2. In order to accomplish the foraging task, four basis behaviors were identified:

(1) Avoid – Collision detection and response

(2) Wander – Searching randomly for targets

(3) Follow – Following an ink trail to the food source

(4) Homing – Picking up a target, moving to home, and dropping the target.

Appropriate implementation of these four behaviors makes it possible for each individual to complete the task object location and retrieval. Notably, these basis behaviors are identical to those used in simulation (see Subsection 5.3.1), with the necessary addition of Avoid. In both simulation and experimentation, as expected, the behaviors generate the interaction necessary for cooperation and resulting self-organization. It is important to note that, while arbitration between the behaviors is based primarily on sensory input, this is not strictly a reactive control architecture. Mataric makes a distinction between reactivity and behavior-based operation of a mobile robot, but notes that “behavior-based systems embody some of the properties of reactive systems, and usually contain reactive components” [27], p. 3. The obstacle avoidance behavior is one example of a reactive component to the control system. The motor commands are a direct reaction to a specific environmental stimulus (touch sensor depression).

Three levels of competence are designed within this control architecture, Levels 0, 1, and 2. During the program development, a sequential “layered approach” was employed to develop each competence level. The zeroth competence level was first written, debugged, and tested to flawless operation before adding the first level. This process was repeated until the robot achieved "Level 2 competence." The basis behaviors are arranged below according their respective competence level:

Zeroth Level:	Avoid
First Level:	Wander
Second Level:	Follow, Homing

In order to elucidate the concept of subsumption, each competence level is represented schematically below in Figure 83, Figure 84, and Figure 85.

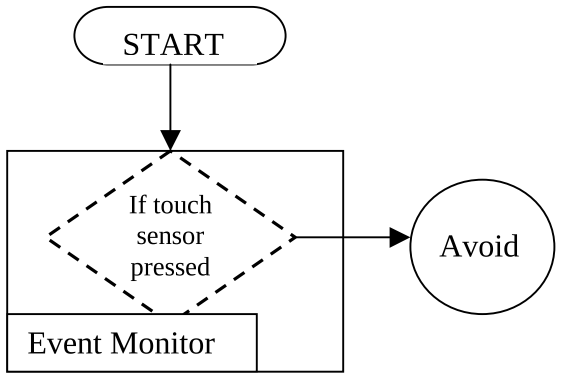


Figure 83. Competence Level 0

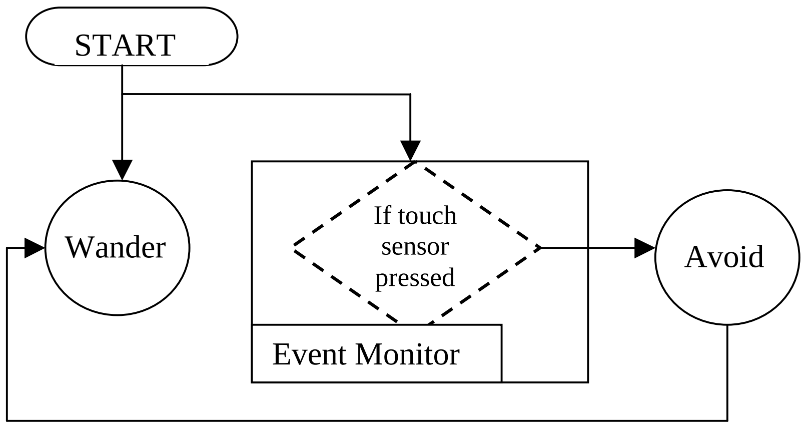


Figure 84. Competence Level 1

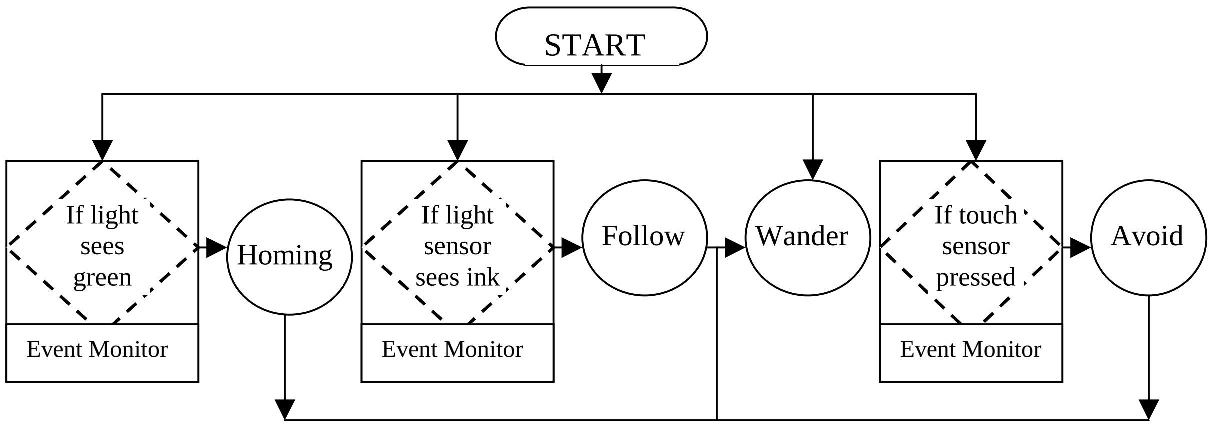


Figure 85. Competence Level 2

Note, for example, the inclusion of Level 0 functionality in the Level 1 diagram. Level 0, however, operates as normal with no knowledge of the existence of first level behaviors. One also observes that arbitration between behaviors occurs on the basis of sensory input. While the notion of behavior priority, in terms of access to motor control, is not clearly defined in such an architecture, the basic order of priority, from highest to lowest, is (1) Avoid, (2) Homing, (3) Follow, and (4) Wander. With the necessary exception of avoiding obstacles, behaviors directly related to object retrieval possess highest authority. This priority ordering is based on the biological example of foraging in ants. Food retrieval to the nest takes the highest priority, as the colony seeks to maximize energy yield.

Subsection 6.5.2: Basis Behaviors

The basis behaviors identified above represent all of the necessary functionality required to complete a foraging task. They were directly inspired by foraging behavior in social insects such as ants, especially those engaging in mass recruitment. Choosing an appropriate set of basis behaviors is not a rigorously defined process, and the concept of an optimal set of behaviors is rejected by Mataric [61]. Brooks notes that one important characteristic in selecting basis behaviors is simplicity: “Behaviors tend to be simple so that computational ‘depth’—the computational path from sensor to actuator—is minimized to maintain a high degree of interactivity with the environment” [9], p. 17. Mataric goes further, providing the following criteria for determination of suitable basis behaviors:

> *A basis behavior set should contain only behaviors that are* necessary *in the sense that each either achieves, or helps achieve, a relevant goal that cannot be achieved with other behaviors in the set and cannot be reduced to them. Furthermore, a basis behavior set should be* sufficient *for accomplishing the goals in a given domain so no other basis behaviors are necessary. Finally, basis behaviors should be simple, local, stable, robust, and scalable.* [61], p. 4

Directed by these criteria and inspired by the example of “basis behaviors” employed by individual worker ants, the final set of four basis behaviors was determined. The specifics of each behavior are presented schematically in Figure 86. The complete robot program was written in NQC (see Section 6.2) and is found in Appendix E.

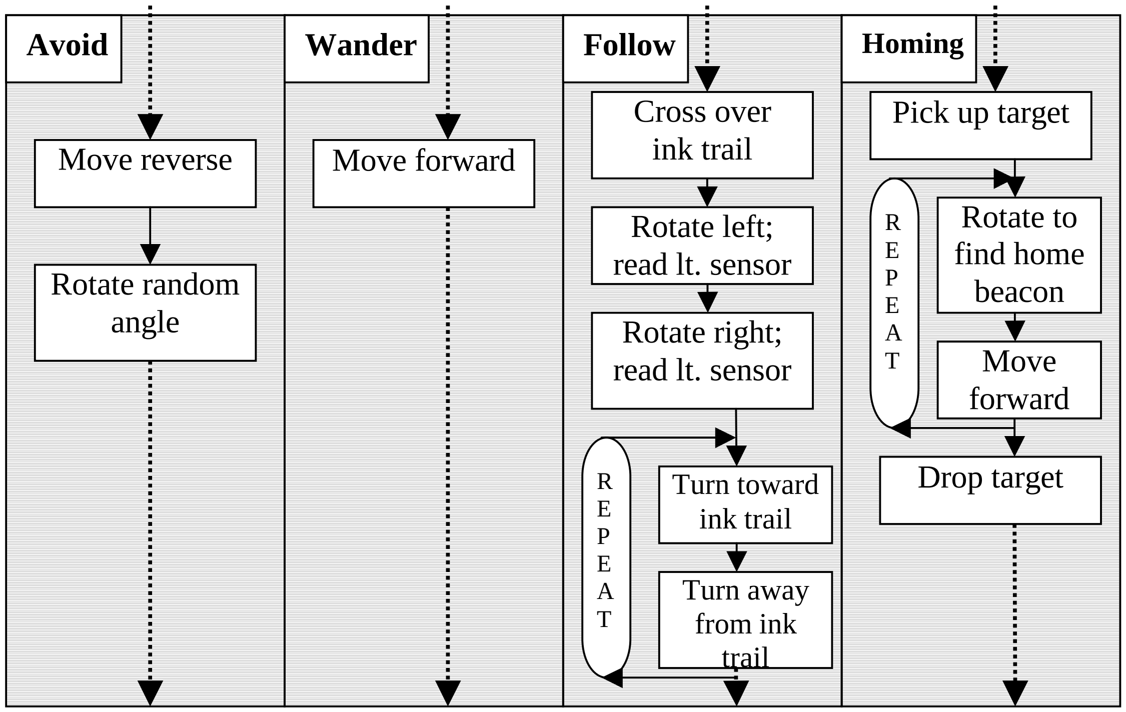


Figure 86. Actions taken in each basis behavior

The simplest behavior, Avoid, is responsible for collision detection and avoidance. Obstacles in the foraging environment include the perimeter boundary and other robots, which are not distinguished from immobile obstacles. Collision detection is achieved via the touch sensor. In response to this stimulus, the robot moves directly backward for a short distance and rotates counter-clockwise through a random angle. During the evasive maneuvers, the touch sensor is monitored, allowing initiation of avoidance within execution of the avoid behavior.

Wandering fulfills the requirement of searching the foraging field for targets. An extremely simple search method consists of following a straight path until a collision occurs. The Avoid behavior then produces a new initial heading, at which point straight motion resumes. All three sensors (both light sensors and the touch sensor) are monitored during Wander: the touch sensor senses collisions, the forward-looking light sensor looks for home to prevent wandering close to home, and the downward-facing light sensor sees either a target or an ink trail. Wandering can be interrupted at any time by any of the other three basis behaviors.

Follow instructs the robot to follow an ink trail sensed during Wander. The process of following a trail is composed of two processes. Initially, the robot decides which direction leads toward the food source. This accomplished by orienting itself approximately parallel to the trail in opposite directions, in turn, and taking light intensity readings from the forward-looking light sensor. Whichever direction produces a lower intensity (high intensity light emanates from home) is the direction of the target source. After choosing a direction, the robot engages in line-following along either the right or left edge of the trail, depending on the initial direction of approach (see Figure 87). A thorough discussion of line-following techniques in LEGO robots is found in [29]. For following the left edge of a trail, the robot turns right until it sees the trail, turns right to move ahead and off the trail, and repeats the process. The number of iterations is determined by the distance to home, which is based on the reading of the forward-looking light sensor taken during the decision on following direction. Upon completion of these iterations, wandering behavior occurs. Wandering starts in the immediately vicinity of a target source, causing Homing to occur with a high probability after completion of Follow.

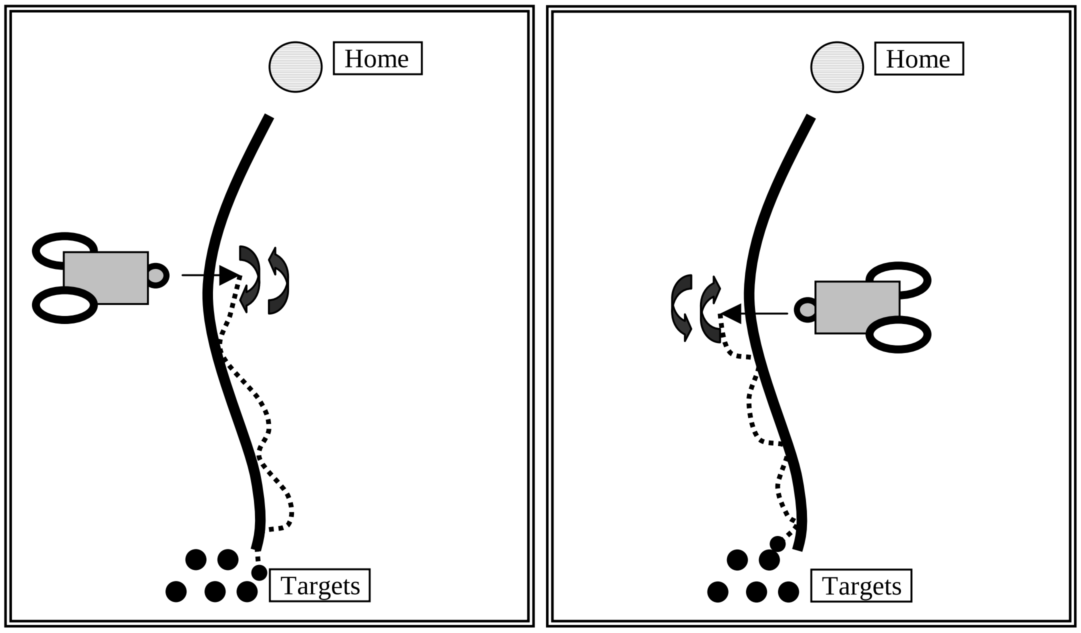


Figure 87. Robot line following, approaching trail from left or right

Homing consists of three sub-behaviors, all of which are required to return the target to home. Having sensed the target on the ground, the robot must pick it up magnetically. This is done by moving forward a small amount and descending the magnet apparatus, which is coupled via the non-drive-system motor to rotation of the pen holder mechanism into the down position. The next step is homing, which takes place by reading the forward-facing light sensor and setting a heading toward the highest intensity. Along the way, ink deposition continues through continuous driving of the pen holder mechanism. The path to home is corrected periodically along the way, but the robot considers itself home when a certain maximum light intensity is reached. At this point, the pen and magnet apparatus are elevated, releasing the target. The robot then rotates approximately 180° (back toward the food source) and commences with Wander.

As a complete control scheme, these four basis behaviors produce the desired foraging behavior. They also facilitate stigmergic communication via trail-laying and following. The result is an operational system of mass recruitment-based and self-organizing foraging autonomous robots

Section 6.6: Experimental Setup

The experimental testing environment is described in this section. Each experiment consisted of an initial spatial distribution of targets and 1-7 robots positioned at home and directed radially toward the perimeter of the foraging field. For experiments with multiple robots, approximately even angular spacing was maintained between individuals at the start of the experiment. In cases involving clusters of targets (Cases 2 and 3), one robot was positioned facing each cluster to ensure rapid initial discovery of each cluster. This practice avoids the possibility of an extended initial searching phase before discovery of a cluster. Power limitations make it impractical to allow such an initial search phase. Each test was initiated by starting a timer and quickly pressing "Run" on each RCX. From that point forward, the robots operated autonomously except in the event of a correctable malfunction.

Mechanical malfunction occurred in a number of tests, but over 90% of the trials occurred without error. In rare cases, two robots would become entangled without the ability to separate. Also, very specific angles of approach to corners sometimes caused stagnation, rendering an individual immobile. Most of these cases were correctable during testing, with no residual influence on the test results. In a handful of tests, mechanical failure of an individual robot rendered the test invalid. Such tests were immediately stopped and the results discarded.

Prior to each test, the battery level of each robot was checked to ensure equivalent operation across all individuals. If the battery level at the beginning of a test was below 8.7 V (max 9.5 V), all six AA alkaline batteries were replaced.

Two trials were conducted for each test scenario. This provides some estimation of variance within a particular scenario and provides an improvement in data credibility over a single data point. Unfortunately, financial constraints made more thorough testing impossible. Detailed statistical analysis is unfounded with only two data points per test, but battery and ink consumption placed limitations on the number of tests. During the course of experimentation, over 200 AA batteries were used. Due to the sensitive dependence of robot operation on voltage supply and an approximate 20 minute battery lifetime (under continuous operation), battery replacement for a single robot occurred every two or three trials.

Data collection was performed by two primary means: robot data-logging and direct observation. The RCX supports data logging, which was uploaded from each robot to a PC following each test. Every robot logged a variety of data during each test, including the time spent in each behavior, the number of obstacles encountered, etc. Global data was collected by hand, including overall collisions, pick-up and drop-off times for each target, and the total task completion time.

Subsection 6.6.1: Targets

In order to conduct robotic foraging experiments, it is necessary to select an object that corresponds to items of food in the world of social insects. Depending on the ant species and the foraging environment, the nature of food items varies. Some species feed primarily on discrete food items carried one-at-a-time, such as seeds. Workers from other species are able to carry multiple discrete food items in a single foraging trip. Food sources can also be pools of liquid (e.g., nectar, sugar water), such as those commonly presented to ants in laboratory experiments [82]. Nearly all physical experiments of foraging with real robots employ discrete targets carried one-at-a-time back to home (e.g., [12, 25, 58, 69]). This approach is far simpler to implement than simulating other types of biological food sources.

A novel mechanism for target acquisition is used in this experimental design: magnetic attraction. As described in Subsection 6.4.5, a vertically-adjustable permanent magnet attached to the underside of the robot causes metal objects to levitate onto the flat cover plate. Such a mechanism requires that at least some component of the target is a ferrous metal. In addition to containing ferrous metal(s), each target must be visible to the robot. The downward-facing LEGO light sensor must be able to distinguish the target from the white floor. The final requirement is that targets have a profile low enough such that they don't represent an obstacle to robotic locomotion. Most importantly, the small caster wheel must roll smoothly over the targets. The design requirements for the targets are summarized below:

- contain ferrous metal(s)
- visible to LEGO light sensor
- low-profile

A simple and inexpensive solution is provided by the use of thin, stainless-steel washers. Each washer was painted a dark-green color with a flat finish to provide contrast to the white substrate. The washers are a standard hardware component, with one inch diameter and a thickness of 1/16” (see Figure 88). Each washer also has a small hole in the center, which provides no operational advantage or disadvantage.

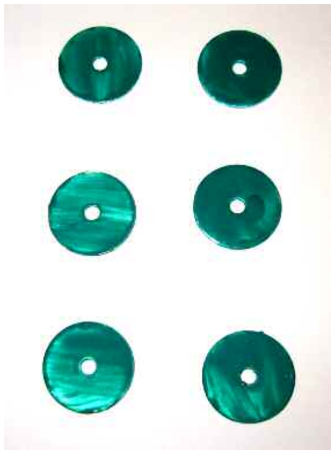


Figure 88. Washers used in experimentation

Subsection 6.6.2: Foraging Area

Foraging experiments were conducted within an enclosed foraging field surrounded by an octagonal perimeter. The perimeter was formed by eight straight sections of 4" diameter PVC pipe joined at the ends by eight 135° PVC angle-joints. The octagonal side length was approximately 52 inches, providing a maximum distance from center to perimeter of 68 inches (minimum distance = 63”, see Figure 89). A cylindrical metal column from floor to ceiling at the geometric center of the field is considered "home." The cement floor was covered by ¼" thick foam-board, which was adhered directly to the floor. Seams between sections of foam board were taped together. The foam-board substrate was covered by standard white poster-board, with the glossy surface facing down. Poster-board was stapled to the foam board and replaced every few tests to prevent problems of trail reinforcement (see Section 6.3) occurring between trials.

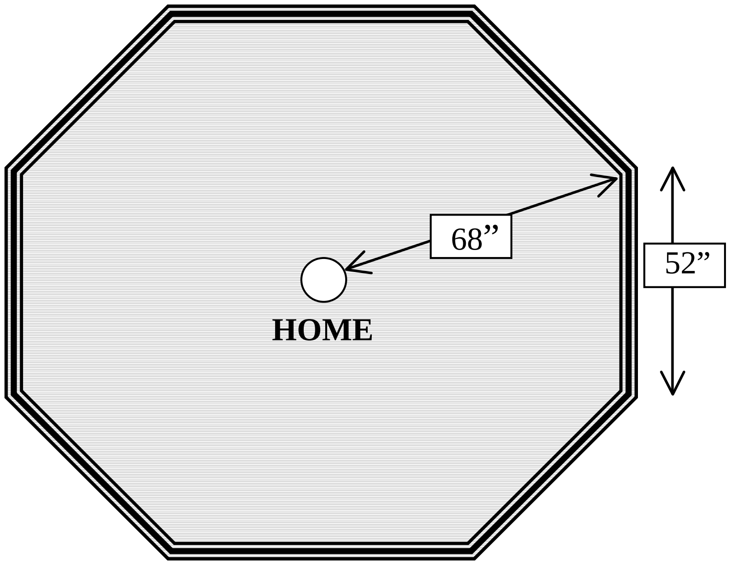


Figure 89. Experimental foraging field

Since beacon-based navigation was required for the homing process, a source of light was located near the ground at home. Two standard 60 W light bulbs were suspended approximately 6" above the floor, providing light intensity visible to the forward-facing light sensor (Figure 90). General lighting was supplied by five 60 W flood lights approximately 7 feet above the surface. Ambient light from external sources (windows, etc) was shielded in order to maintain consistent lighting across all trials.

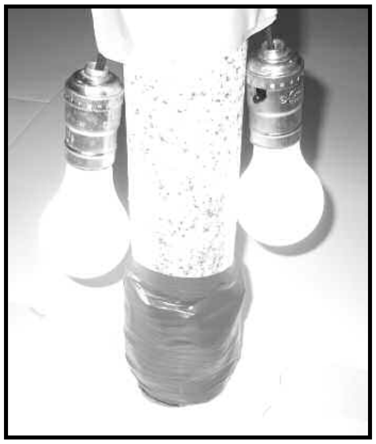


Figure 90. Two light bulbs are home beacon

Section 6.7: Experimental Cases

Three experimental scenarios were designed to meet the major goals of experimentation (see Table 12), listed in Section 6.1. The three cases are distinguished only by the target distribution and the definition of task completion. In Case 1, ink deposition and following were not used, due to their obvious irrelevance in a case without clustered targets. Otherwise, robot operation and data collection were constant in all three cases.

Table 12. Summary of Experimental Scenarios

Scenario	No. of Robots	Target Distribution	No. of Targets	Task Completed When
Case 1	1-7	random	48	25% collected
Case 2	2-7	one cluster	16	75% collected
Case 3	2-7	two clusters	4, 12	75% of large cluster collected

Subsection 6.7.1: Case 1

This first case is designed for use as a basis of comparison for the subsequent two cases. This case also corresponds to the first simulation case. It demonstrates completion of the foraging task without cooperation. It is also designed to show that task efficiency improves in a non-cooperative task, but the improvement trend is unlike that characteristic of swarm intelligent systems. Targets (48) were distributed randomly throughout the foraging field, but no targets were placed within two feet of home. This case was conducted with one to seven robots, with task completion defined as collecting 25% of the targets. It is common to make such a definition of task completion, due to the fact that target retrieval slows considerably with decreasing target density. When only a few targets remain in the foraging field, the probability is very low of each robot finding one. Given a limited power supply (and consequent running time), the test was deemed complete after retrieval of 12 out of 48 targets.

Subsection 6.7.2: Case 2

Case 2 features a single cluster of 16 targets located approximately 3 inches from a corner of the octagonal perimeter. This case was designed to demonstrate the mechanisms of positive and negative feedback, the characteristic formation of emergent patterns, and increased task efficiency due to cooperation. Positive feedback reinforces an initial trail and causes increasingly frequent target retrieval, while negative feedback is manifested in trail disappearance over time. A well-established ink trail emerges as a result of stigmergy and positive feedback. The case was conducted with 2-7 robots and each test ended with retrieval of 12 out of 16 targets. In order to demonstrate foraging system performance with clustered target distribution but without cooperation, this case was also tested without trail-laying and following.

Subsection 6.7.3: Case 3

Case 3 features two clusters of unequal size: one of 4 targets and one of 12 targets. The clusters were located on opposite sides of the foraging field, each approximately 3 inches from a corner of the octagonal perimeter. This case was designed to demonstrate the mechanism of negative feedback, as well as the system property of multistability. Negative feedback allows the system to abandon the small target cluster upon complete depletion. The potential to exploit either target cluster exclusively, or both simultaneously, represents the existence of three stable states (multistability). The case was conducted with 2-7 robots and each test ended with retrieval of 12 out of 16 targets.

Section 6.8: Results

Subsection 6.7.1: Case 1

The first experimental case tests the foraging capability of group of robots without cooperation. All robots used in this case were incapable of laying or following trails. Clearly, for the case of random target distribution, trail-laying and following does not improve performance. Each trail would simply lead to an empty space where a target was formerly located. The experimental setup is shown in Figure 91, with a group of three robots prepared for the foraging task. Notice the random distribution of 48 targets and the bright light emanating from home.

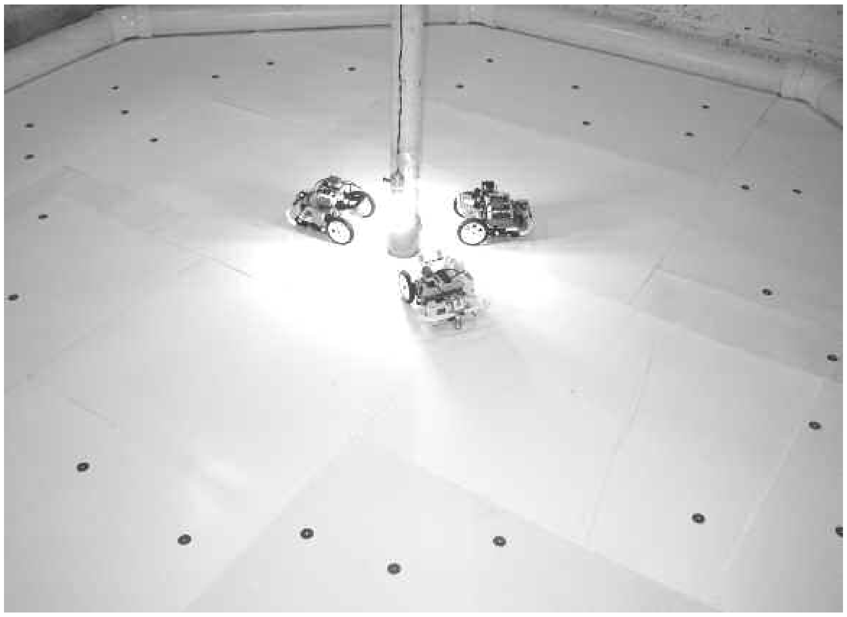


Figure 91. Experimental setup (random distribution, N=3)

As in the presentation of simulation results (see Section 5.5), the time evolution of object collection provides an understanding of the system behavior. In Figure 92 below, the number of targets remaining until task completion is plotted against time. Each curve corresponds to a different group size, ranging from one to seven robots. As a group, the curves follow a roughly linear trend throughout the test. For a given group size, the collection rate is nearly constant from the beginning of the task until the sixteenth target is retrieved. Due to the fact that only 25% of the targets were collected before each trial was stopped, the change in target density barely influences collection rate. If the trials were allowed to continue longer, these collection rates would diminish in response to decreased target density.

The rate of target collection shows a strong dependence on group size. When a single robot attempts the foraging task, it exhibits the slowest collection rate, as expected. This corresponds to the longest time required for task completion, as shown in Figure 93. The task completion time decreases with increasing group size, up until N = 5. For larger group size, the trend reverses, and the foraging time increases. The task completion time for N = 7 is the second highest, behind only the case of one robot.

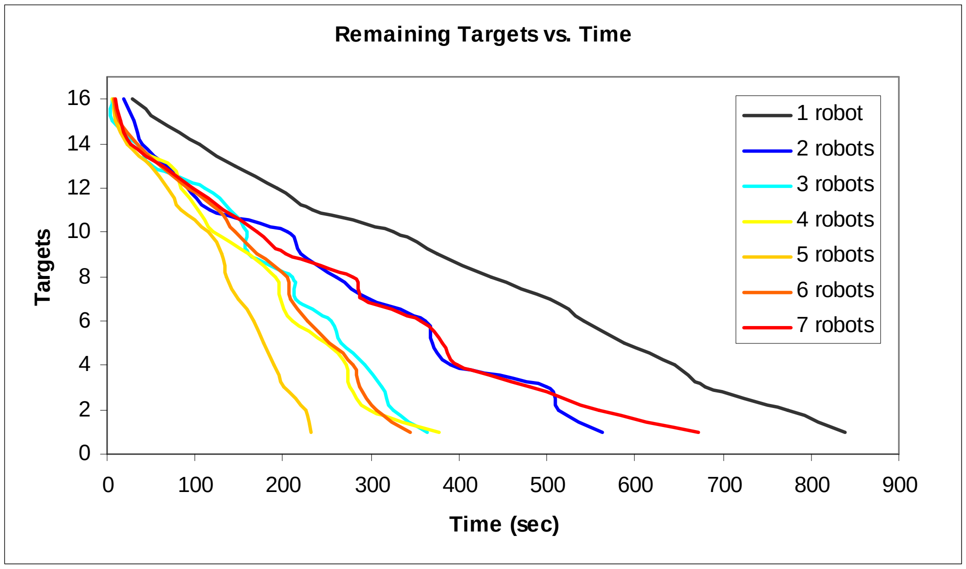


Figure 92. Time evolution of target collection (random dist.)

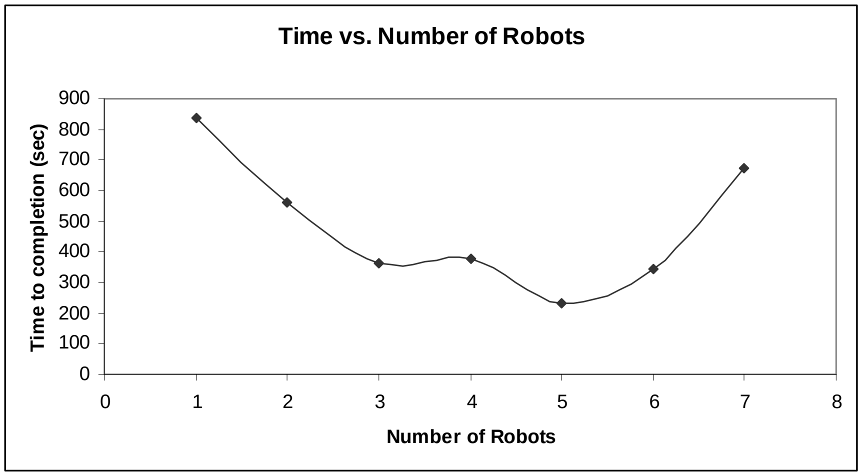


Figure 93. Collection time versus number of robots (random dist., avg. of 2 trials[4])

Figure 94 illustrates the collision frequency, in collisions per minute, versus the number of robots. As expected, the collision frequency increases with group size, due to the increased spatial density of robots. The high collision rates for larger group size explain

[4] All subsequent graphs of this type reflect the average of two trials.

the decreasing task efficiency for N > 5. For N = 7, each robot collided with another robot every 12 seconds, on average. Robots experiencing frequent collisions waste time in collision avoidance; that time could otherwise be used in retrieving targets. This lost time reduces the overall efficiency of the group.

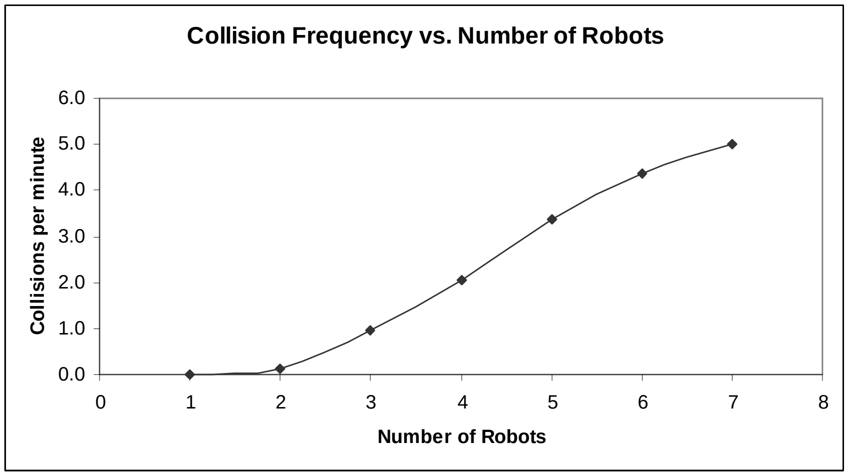


Figure 94. Collision frequency versus number of robots **(random dist., avg. of 2 trials[5])**

Subsection 6.7.2: Case 2

Case 2 includes two variations of the same target distribution: a single cluster of 16 targets. Presented first is the non-cooperative foraging method. This is similar to Case 1 (see Subsection 6.7.1), but with a spatially concentrated target distribution. Second, the results are presented for cooperative foraging robots capable of laying and following ink trails. Testing this case with and without cooperation provides a method of determining the efficacy of the cooperative strategy (as compared to the nominal case of non-cooperation).

[5] All subsequent graphs of this type reflect the average of two trials.

NO INK TRAILS

When ink trails were not used by the foraging robots, the rate of target retrieval was quite slow compared to results from the randomly distributed case, as shown in Figure 95 (cf. Figure 92). Due to the high spatial concentration of targets in a small region of the foraging field, the probability of each robot finding a target was low. The evolution of target collection was erratic in each case, due to the significant random component in searching for a small cluster. Note that each curve below is characterized by long periods of searching in which no targets were collected. In general, the collection rate increased with group size.

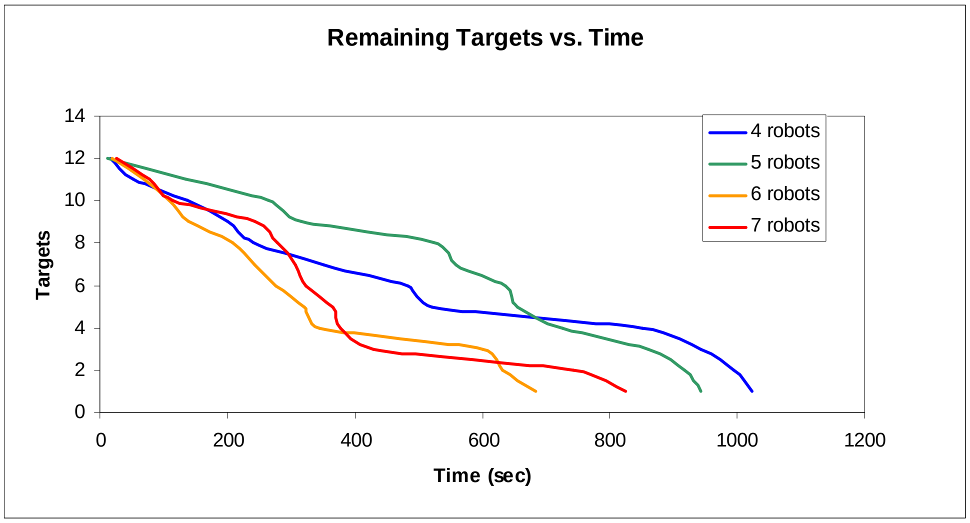


Figure 95. Time evolution of target collection (1 cluster, no trails)

The total task completion time improved between N = 4 and N = 6, due to increased robot density (see Figure 96). However, for N = 7, the high frequency of interactions between robots caused the completion time to increase. This trend is similar to that shown in Figure 93. Figure 97 illustrates the expected increase in collision frequency with group size.

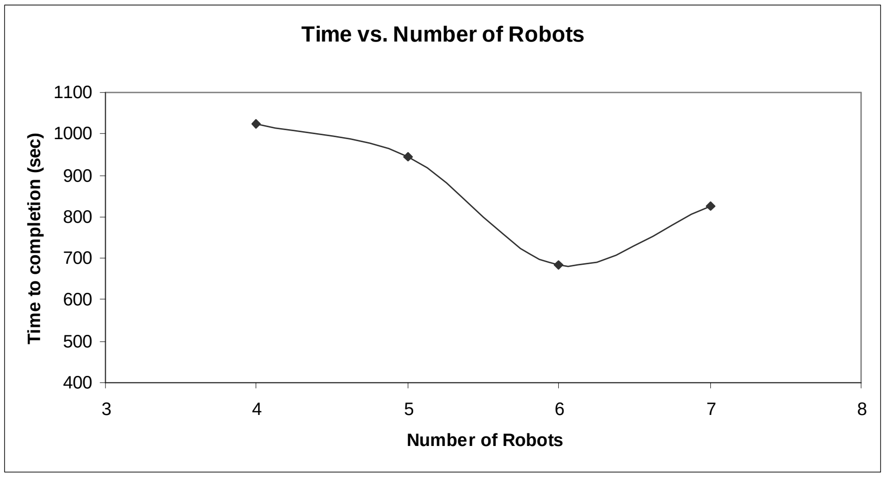


Figure 96. Collection time versus number of robots (1 cluster, no trails)

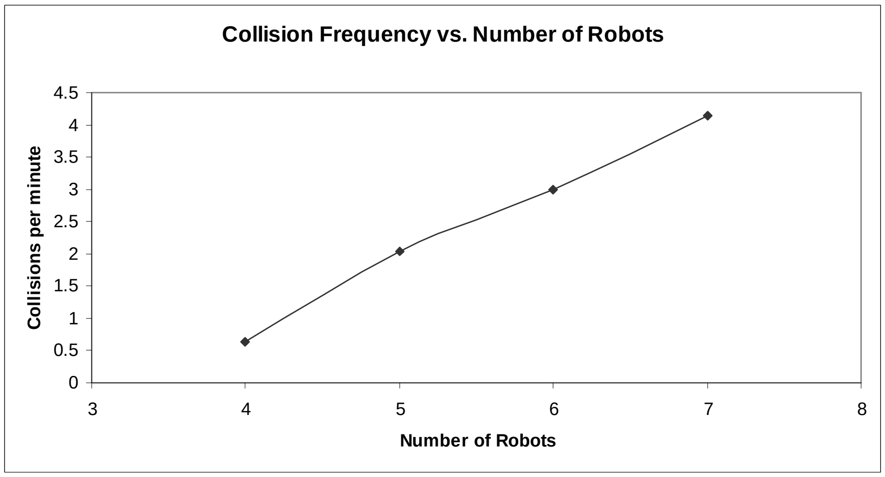


Figure 97. Collision frequency versus number of robots (1 cluster, no trails)

WITH INK TRAILS

For items distributed in clusters, the mass recruitment foraging strategy is shown in insects to be an effective mode of cooperation [101]. This is tested with real robots in this section. The experimental setup is pictured in Figure 98, with four robots prepared to start the test. Note the cluster of 16 targets in the top right corner of the photograph. A series of photographs taken during one trial are presented in Figure 99 (on the next page). The pictures are viewed in sequence from top left to bottom right. At t = 200, three targets have been retrieved, as evidenced by the three ink trails leading to home. Note that each target was removed from home upon retrieval in order to prevent it from being collected a second time by a robot wandering near home. At t = 400, the initial three trails have weakened significantly, beyond the point of being followed. However, notice that a robot is following a trail deposited since t = 200. At t = 600, only six targets remain in the cluster, and a number of trials are present. The oldest ink trails have disappeared completed, but recently deposited trails remain clearly visible. One robot is depositing a trail on the way to home and another is following a trail to the target cluster. At t = 700, the final target is collected. Only two strong trails remain visible.

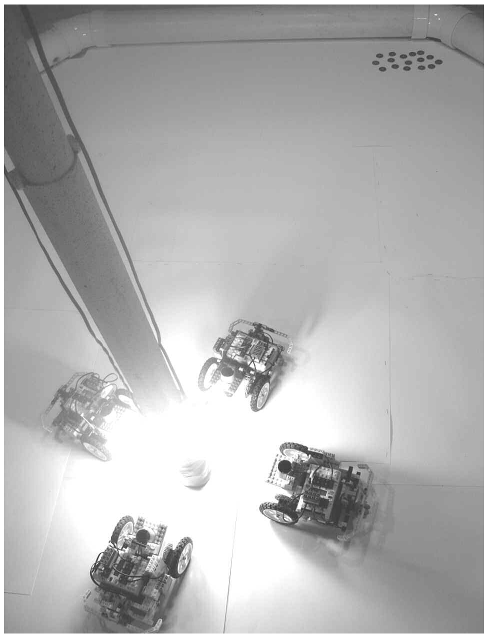


Figure 98. Experimental setup (1 cluster, N=4)

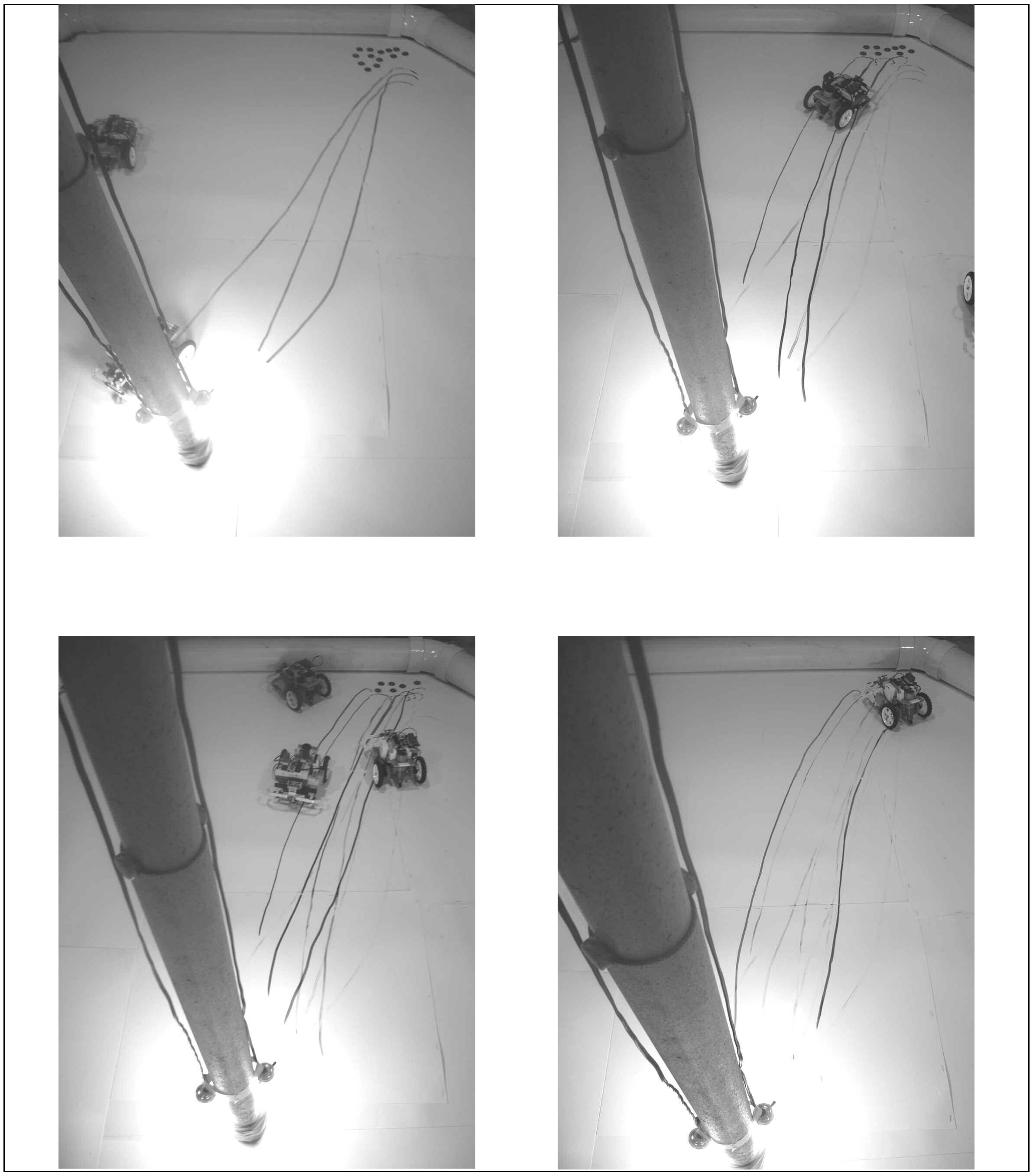


Figure 99. Series of testing photographs (from top left to bottom rt., t=200, 400, 600, 700 sec)

Figure 100 shows the time evolution of target collection for each group of robots tested. In comparison to the results without ink trails (see Figure 95), these curves are less erratic. The collection rates are more constant, but still exhibit a fair degree of variation in time. For N = 5, one observes the downward concavity for 50 < t < 400 seconds that is a signature of positive reinforcement and effective cooperation. The same type of curve shape resulted from StarLogo simulations of foraging with a single food cluster (e.g., Figure 28).

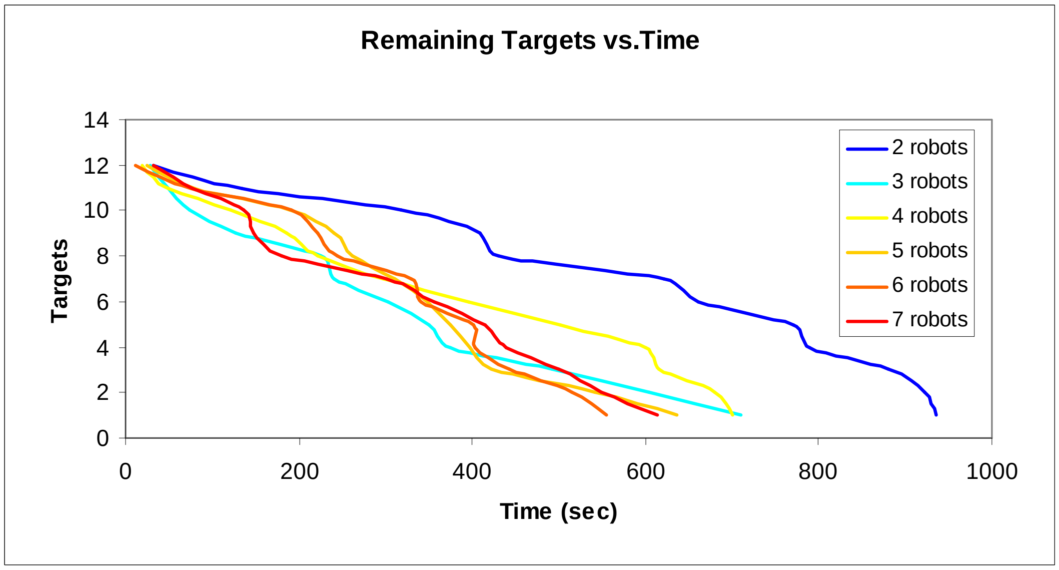


Figure 100. Time evolution of target collection (1 cluster)

The total collection time decreases steadily from N = 2 to N = 6. For N > 4, the collection time begins to level out, and it actually increases for N = 7. In order to observe the benefit of cooperation, the results without trails are included in Figure 101. Clearly, when trails were used as a foraging strategy, the foraging efficiency increased significantly. The completion time decreased by at least 19% (N = 6) and at most 33% (N = 5). In addition, the detriment of high collision frequency was mediated by cooperation. This is reflected in the fact that system performance was only 11% worse (from N = 6 to N = 7) when trails were used, as compared to 21% without trials.

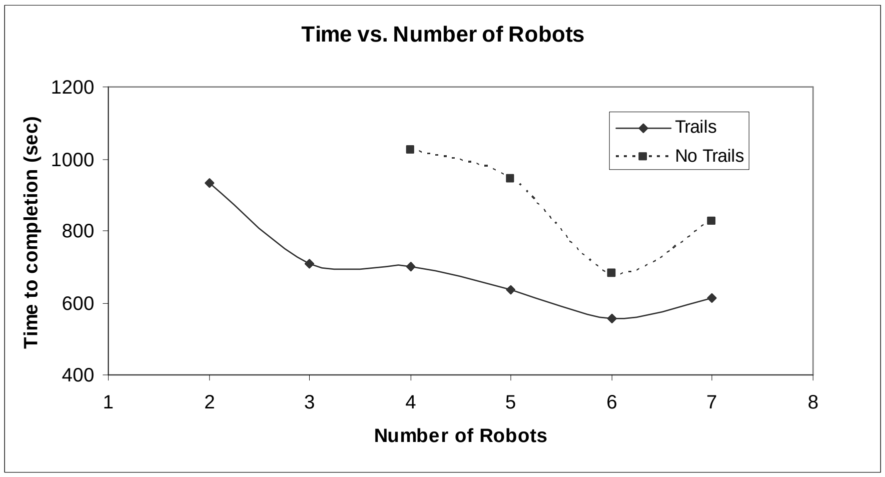


Figure 101. Collection time versus number of robots (1 cluster)

As discussed previously, the collision frequency increases with robot density (Figure 102). A similar trend is observed when cooperation is introduced, but it appears that cooperation slows the rate at which the collision frequency rises. This could be due to the fact that self-organized behavior is occurring when cooperation is introduced. Robots engaged in following and target retrieval tasks are less likely to collide with other robots. These tasks involve slower motion and less area coverage within the foraging field when compared to the default wandering behavior.

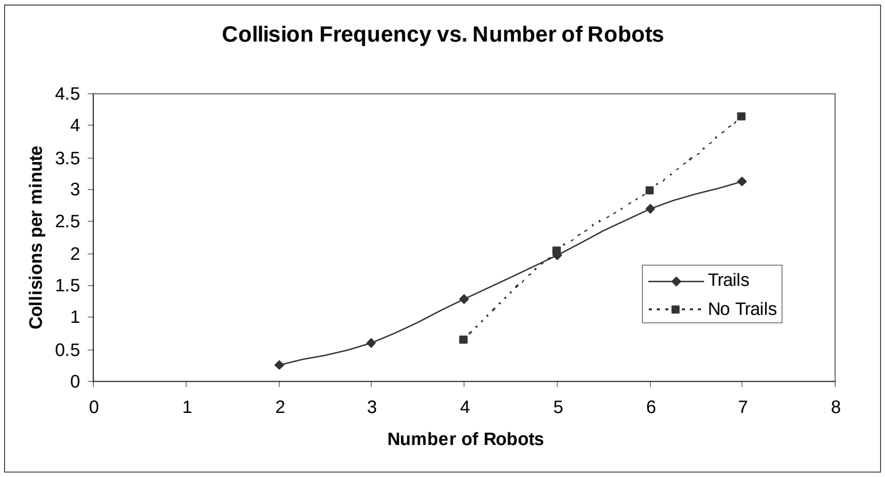


Figure 102. Collision frequency versus number of robots (1 cluster)

Subsection 6.7.3: Case 3

In this case, two target clusters were present in the foraging field. With a maximum of seven foraging robots, a full investigation of this case is not possible. Preliminary tests revealed that seven robots are not sufficient to demonstrate cooperative foraging behavior in the presence of two sources. Robot density is not high enough to establish a stable trail to either cluster. For this reason, the test was designed to demonstrate only one specific system behavior: adaptability. One of the two sources contains only four targets (see Figure 103), which ensures that it is completed before the other source (contains eight targets). After the small source is depleted, the trails eventually disappear, allowing the system to focus only on the remaining cluster. This behavior was observed in 9 out of the 10 trials.

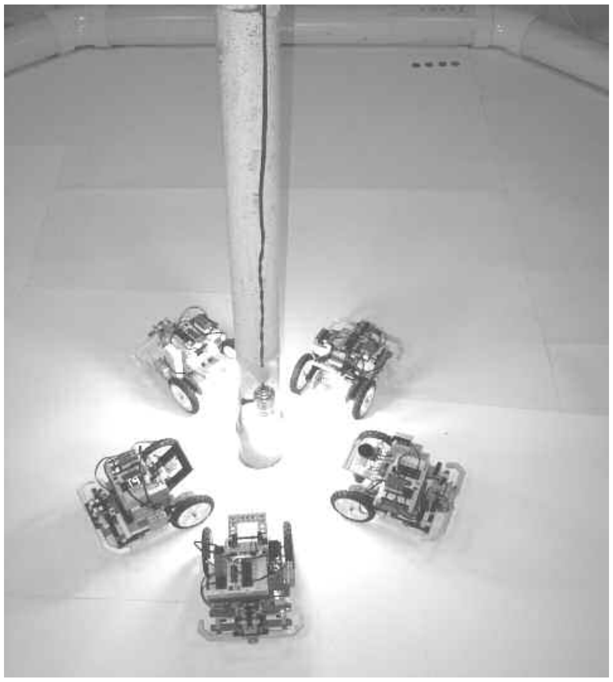


Figure 103. Experimental setup (2 clusters, N=5)

The plot of task completion time versus group size (Figure 104) for this case follows the usual trend, except for the case of N = 7. One would expect the completion time to be higher than for N = 6, due to higher collision frequency (see Figure 105). The discrepancy was caused by one of the two trials in which seven robots operated. In this trial, the seven robots briefly established trails to both food sources, which explains the improvement in performance compared to N = 6. The relationship between collision frequency and group size also exhibits the expected trend. For N = 5, the collision frequency is somewhat higher than expected, but this could simply be a function of random elements in each trial. Since only two trials are averaged for each data point, the trends can only be examined in a general sense.

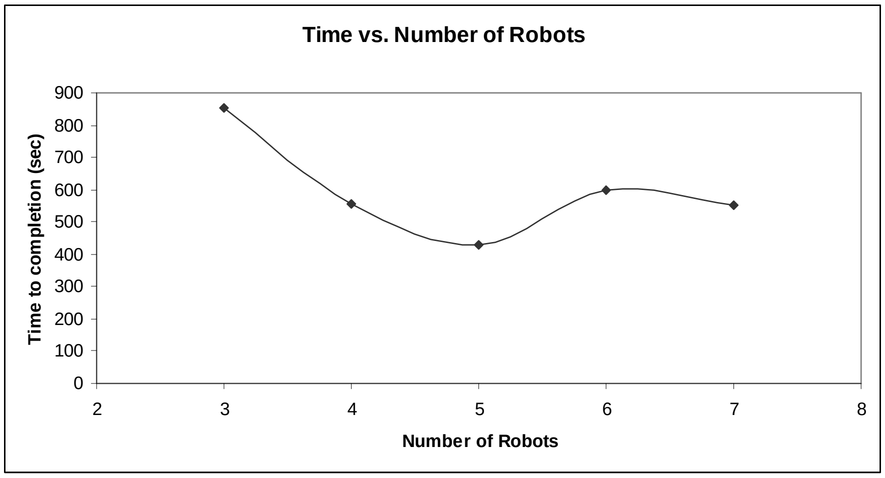


Figure 104. Collection time versus number of robots (2 clusters)

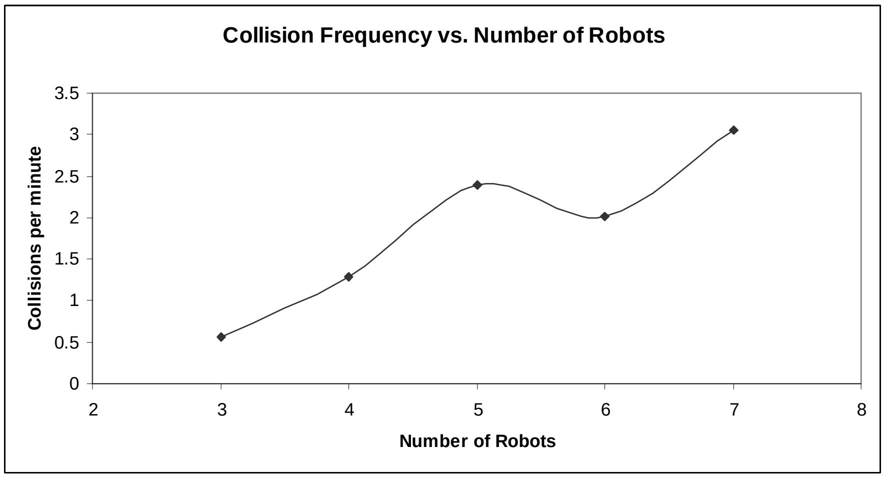


Figure 105. Collision frequency versus number of robots (2 clusters)

Figure 106 shows the time evolution of target collection from the cluster of eight targets. The most interesting feature of these curves is the increase in collection rate after t ≈ 150 seconds. This corresponds approximately to the time at which the cluster of four targets is fully depleted. After a short delay, during which the unneeded ink trails disappear, more robots are able to contribute to cooperative foraging at the only remaining food source. This system transition produces the increase in collection rate, seen most clearly for N = 4. Figure 107 illustrates the progression of target collection from both sources for N = 7. As in most of the trials, the small cluster is depleted first, followed by the remainder of the large cluster.

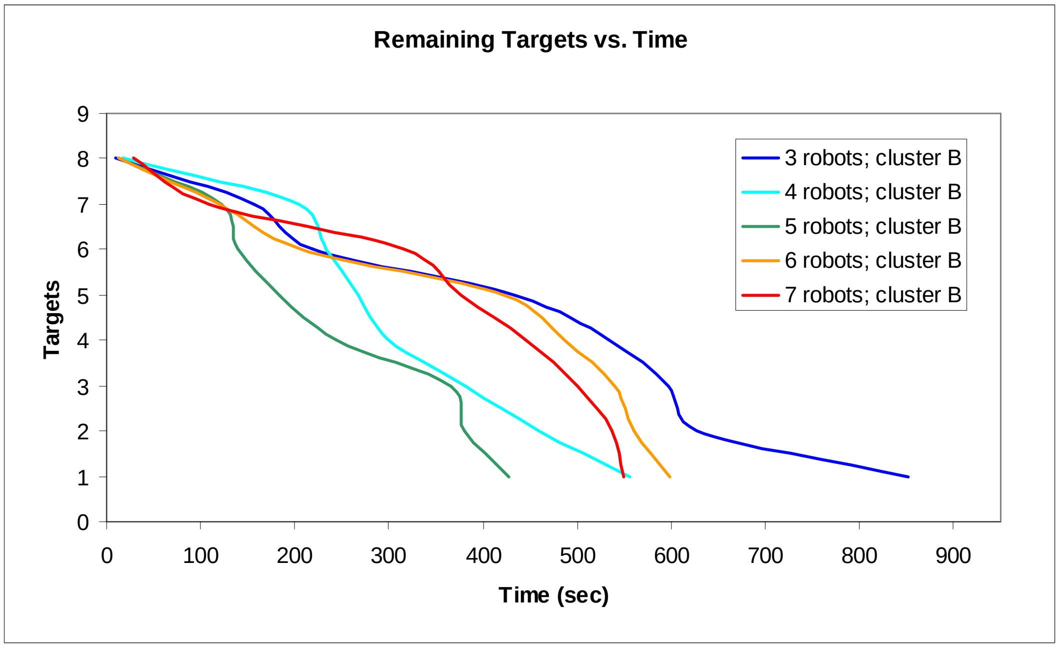


Figure 106. Time evolution of target collection from source B (2 clusters)

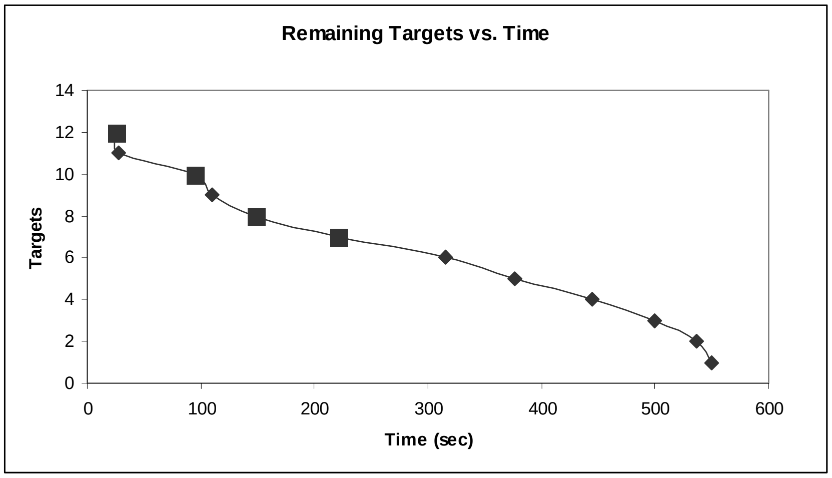


Figure 107. Time evolution of target collection **(2 clusters, N=7; square markers indicate targets from source A)**

Chapter 7: Analysis and Discussion

The following chapter is devoted to a discussion of the simulation and experimentation results from a broader perspective than the results sections of Chapter 5 (Section 5.5) and Chapter 6 (Section 6.8). For both simulation and experiments, the results are discussed below in terms of the most important general findings. These key results are followed by a brief discussion of the weaknesses and possible improvements in the StarLogo simulation and in experimentation with real robots. Finally, the simulation and experiments are evaluated based on the achievement of the goals stated in Chapter 5 (Section 5.1) and Chapter 6 (Section 6.1). To conclude the chapter, the results of simulation and experimentation are taken as a whole and compared to the theoretical predictions from Chapter 4 (Section 4.6).

Section 7.1: Discussion of Simulation Results

Subsection 7.1.1: Important Results

The first key lesson derived from the results of simulation is the significance of power law relationships in the system behavior. These types of trends were most visible in Case 1, which featured a random distribution of food items throughout the foraging field. It was observed in Section 5.5 that the relationships between completion time and group size, as well as between collision frequency and group size, appeared to follow the rough trend of a power law. The simulation results for this case are presented again below, but with power law equations fitted to the data (see Figure 108 and Figure 109). Note that the correlations are quite strong, especially in Figure 109.

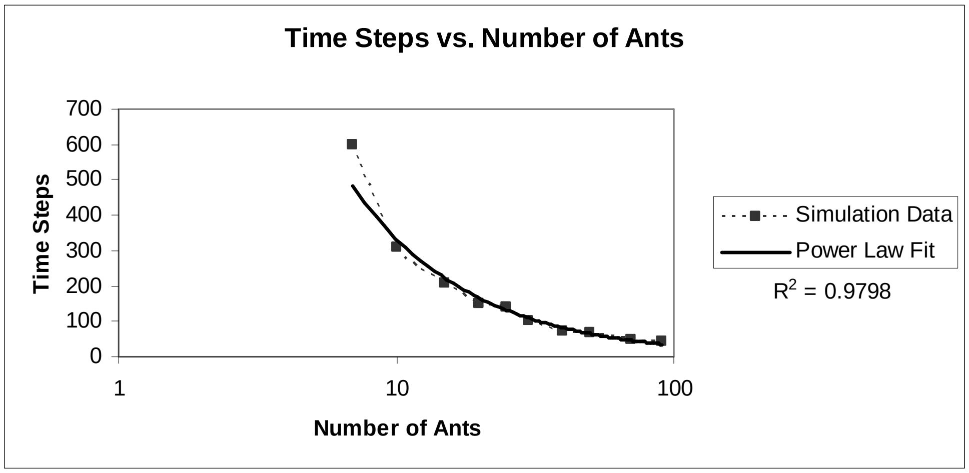


Figure 108. Power law approximation of simulation results (rand. dist.)

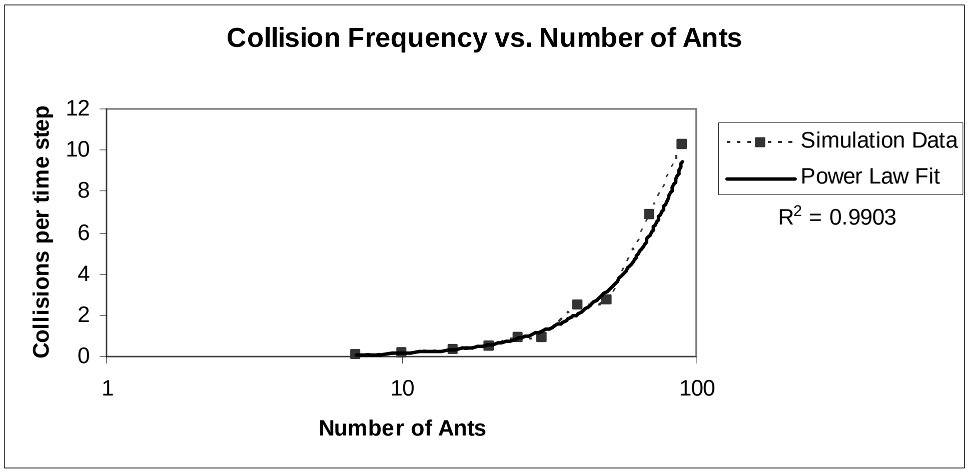


Figure 109. Power law approximation of simulation results (rand. dist.)

Based on these curve fits, it seems that a power law provides an accurate approximation of the system behavior, at least in the case of non-cooperative foraging with a random food distribution. The question then arises, does the power law relationship hold for other simulated scenarios? This question is answered below by the analysis presented in Table 13 and Table 14, which include the coefficient and exponent of the fitted power law for each set of data. The correlation coefficient is also given for each data set, in order to evaluate the closeness of fit provided by a power law relationship. In Table 13, the R^2 values are very high for foraging without cooperation (first two rows of the table).

In these cases, it appears that a power law provides a close approximation of the data. However, for cooperative foraging from one source, the power law approximation becomes less accurate. While the correlation coefficients remain above 0.91, these values reflect an important difference introduced by cooperation. At least with respect to the task completion time, a power law relationship is only a moderately accurate description of the dependence on group size. The table also includes the coefficient, c_3, and exponent, c_4, in each fitted equation. By observing the values of c_4 across all cases, it becomes apparent that they are clustered in the vicinity of $c_4 = -1$. As a result, one can conclude that the completion time is roughly inversely proportional to the number of virtual ants:

$$T_c \ \alpha \ \frac{1}{N} \tag{7-1}$$

Table 13. Power Laws Fit to Simulation Data

Completion Time vs. Number of Ants $T_c = c_3 * N^{c4}$			
Trial	c_3	c_4	R^2
random distribution	1317.7	-0.9885	0.9956
1 source, no trails	7568.1	-0.9787	0.9873
1 source (d=10)	1438.1	-0.8113	0.9541
1 source (d=20)	3740.6	-1.0161	0.9798
1 source (d=35)	40204	-1.5966	0.9179

For the relationship between collision frequency and group size, the correlation coefficients are again very high for both non-cooperative scenarios. In general, the power law curve is less close-fitting to the data for the cooperative cases. For all cases, though, the R^2 value is above 0.956, indicating that a power law relationship is a fairly accurate approximation to this relationship (with or without cooperation). By examining the power law exponents, c_2, one notices a larger variance than in Table 13. The average of these values is $c_{4,\ avg} = 2.18$. If more data was collected, one could obtain a more accurate value for this exponent. The collision frequency is roughly proportional to the square of group size:

$$C_f \ \alpha \ N^2 \tag{7-2}$$

Table 14. Power Laws Fit to Simulation Data

Collision Frequency vs. Number of Ants $C_f = c_1 * N^{c2}$			
Trial	c_1	c_2	R^2
random distribution	0.0004	2.5191	0.983
1 source, no trails	0.0014	2.0973	0.9973
1 source (d=10)	0.0018	1.9217	0.966
1 source (d=20)	0.0022	1.8561	0.9903
1 source (d=35)	0.0001	2.5616	0.9568

The plots of the time evolution of food collection revealed another key result from simulation. By comparing the plots from non-cooperative experiments to those cases in which trails were used, a significant difference in curve shape became apparent. In cases without trails, the collection rate remained nearly constant, with only a slight, gradual decrease throughout. When trails were employed, the collection rate increased, often dramatically, throughout the trial. This created a noticeably difference in concavity of the curves. Positive feedback allowed the group of virtual ants to accelerate food collection, resulting in the signature downward concavity of the curves.

Decay rate was an important parameter in many of the simulation trials. The dependence of system behavior on decay rate was the focus of Case 4, which included three food sources. It was observed that slow trail decay facilitated the establishment of a higher number of concurrent stable trails. On the other hand, a rapid rate of decay inhibited cooperative foraging. With three sources present, for example, a high value for the decay rate could force the group to focus on one source at a time, completing the task in sequence rather than in parallel. A rapid decay rate corresponds to a higher number of foragers necessary to efficiently exploit multiple food sources.

The group size was varied in each major simulation scenario, as it was the primary variable under investigation. It was observed, as expected, that task completion time improved super-linearly with group size. Below a certain density of virtual ants, the group was unable to establish even one stable pheromone trial. As the group size increased, the number of possible concurrent trails increased as well. Large groups were able to forage simultaneously from several sources.

The final key result of simulation is the effect of cooperation on task efficiency. In general, it was observed in Chapter 5 that trail-laying and following resulted in faster task completion compared to no communication. This result is confirmed below with a more clear illustration of the improvement in task efficiency as a result of cooperation (see Figure 110). The task efficiency is defined as the inverse of task completion time. As the group size increases in both curves, the one corresponding to cooperation climbs at a significantly faster rate than the non-cooperative case. Notably, when the graph is compared to the generalized definition of swarm intelligence (see Section 2.2, Figure 1), one must conclude that the group of foraging virtual ants exhibited swarm intelligence by utilizing a mass recruitment strategy.

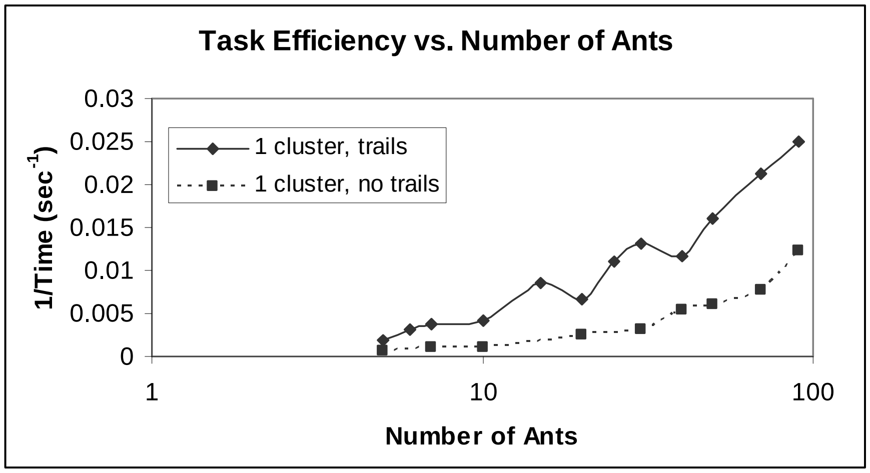


Figure 110. Task efficiency versus number of ants (1 food source)

Subsection 7.1.2: Simulation Weaknesses

As with any computer simulation, this implementation of StarLogo to simulate mass recruitment in an ant colony had certain weaknesses. Of course, the simulation did not incorporate every detail of true foraging behavior in social insects. One important factor missing from simulations was the effect of direct interactions between individuals. Collisions between virtual ants were not simulated. This allows for analysis of the results without the complications introduced by collisions, which can be considered an advantage from a theoretical perspective. It is also similar to the behavior of an ant colony, as the “collisions” between individuals are handled so efficiently that they are not of great significance. However, in the realm of robotics applications, these interactions are important. An accounting was made of collisions between virtual ants, but each interaction had no impact on the behavior of the individuals.

The second important weakness of these simulations is the volume of data. Many of the relationships between important variables and parameters could have been examined in more detail, and with more statistical backing, given a higher volume of data collection. A wider range of decay rates and group sizes could have been explored, for example. The level of detail was beyond the scope of this work, however, which attempted only to demonstrate the existence and qualitative nature of certain dependencies.

Subsection 7.1.3: Achievement of Simulation Goals

To conclude the discussion of the simulation results, it is fitting to consider the degree to which stated goals were accomplished. The goals of simulation, as stated in Section 5.1, are reproduced here for convenience:

- To investigate the predictions of theoretical modeling.
- To visualize system behavior to enhance understanding.
- To extend theoretical knowledge of scenarios for which mathematical modeling is intractable.

The first goal is discussed in detail below in Section 7.3. In general, this goal was accomplished, due to the observation of many theoretical hypotheses. For example, the simultaneous existence of a number of stable foraging states was observed, especially with three or four food sources. The goal of visualizing system behavior was achieved, as represented by the figures in Section 5.5. The use of StarLogo provided excellent visualization of system behavior, which enhanced understanding of foraging dynamics. Theoretical models of foraging fail to capture the complexity of system behavior observed with four food sources. Results from that case briefly introduced complexity for which mathematical models have not been developed.

Section 7.2: Discussion of Experimental Results

Subsection 7.2.1: Important Results

In the previous section, which discussed the results of simulation, the importance of power laws was discussed. It was shown through data analysis that a power law relationship existed between collision frequency and group size, as well as between task completion time and group size. These relationships were also investigated in experiments with real robots. The data was analyzed to determine the degree to which power laws approximate the data. In Figure 111 below, the power law is fitted to the experimental relationship between collision frequency and number of robots. The curve is fitted to the data corresponding to the cooperative case. Notice that the approximation is less accurate than for the simulation data. In Table 15, the power law coefficient, c_1, exponent, c_2, and correlation coefficient R^2, are provided for three experimental cases. The correlation coefficients reveal that a power law relationship does not describe the system behavior with a high degree of accuracy. The fit is only moderately accurate for each of the cases. The power law exponent, c_2, is consistent among the two non-cooperative cases, but is significantly lower for the cooperative case. When cooperation is present, the exponent is approximately $c_2 = 2$, agreeing with the results from simulation (see Table 13).

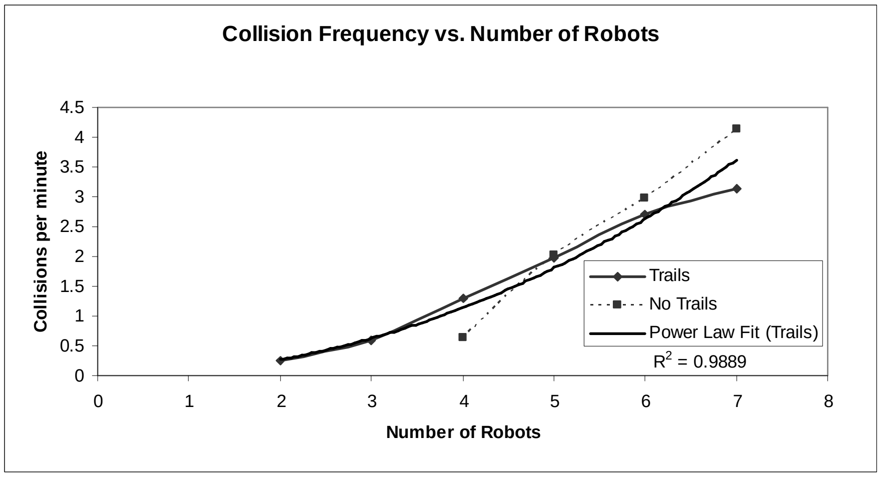


Figure 111. Power law approximation to experimental data

Table 15. Power Laws Fit to Experimental Data

Collision Frequency vs. Number of Robots $C_f = c_1 * N^{c2}$			
Trial	c_1	c_2	R^2
no clusters	0.0149	3.2859	0.968
1 cluster, no ink	0.0082	3.2649	0.9433
1 cluster	0.064	2.0735	0.9889

The second important result from experimentation is the influence of direct interactions between robots on system performance. As discussed in Section 6.8, the task completion time decreased above a certain critical group size. At some point, the detrimental effect of collisions between robots negated the beneficial effect of cooperation. This effect is due to the amount of time wasted by each robot in obstacle avoidance behavior. Figure 112 illustrates the point by providing the percentage of total run-time spent in the avoid behavior. This data was logged by each individual robot and the graph below represents the average values from all robots in a given trial. As expected, the percentage of time spent avoiding obstacles increased with group size. The robots were unable to distinguish collisions with robots from collisions with the perimeter boundary. Therefore, these percentages are biased by the amount of time spent responding to collisions with the perimeter.

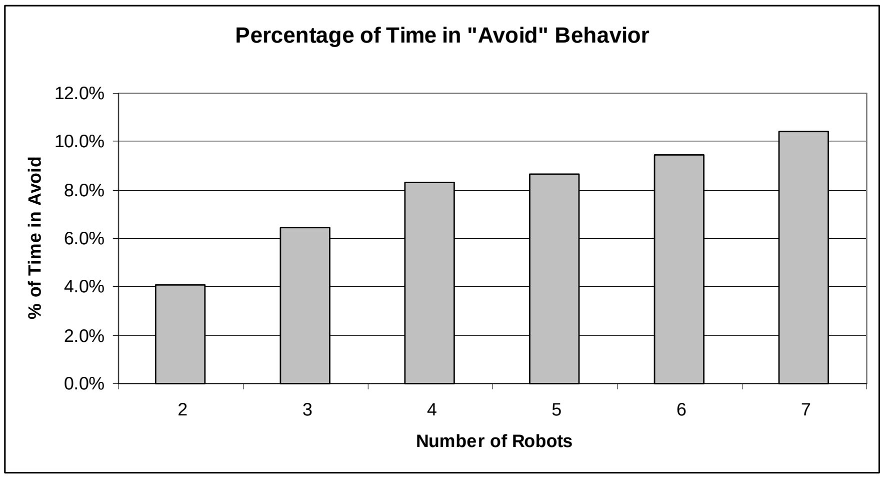


Figure 112. Percentage of time avoiding obstacles (1 cluster, with trails)

As in the simulations, the system dependence on group size is another important lesson from analysis of the experimental results. The robotic system was also unable to establish strong trails to a food source below a certain critical density. Generally, with less than three robots, the group was unable to effectively communicate via trail-laying and following. The task completion time in all experimental cases improved with group size, but showed a reversal of the trend for $N \geq 6$. As mentioned above, the increased frequency of collisions between robots caused this reduction in foraging performance.

The last key result from experimentation is the efficacy of cooperation. It is necessary to answer the question of whether or not stigmergic cooperation between robots effectively increased completion efficiency. As shown in Figure 113, the group of robots performed the completion task significantly faster when using trails. While the benefit of cooperation is not as drastic as in simulation (see Figure 110), the improvement in performance remains clear.

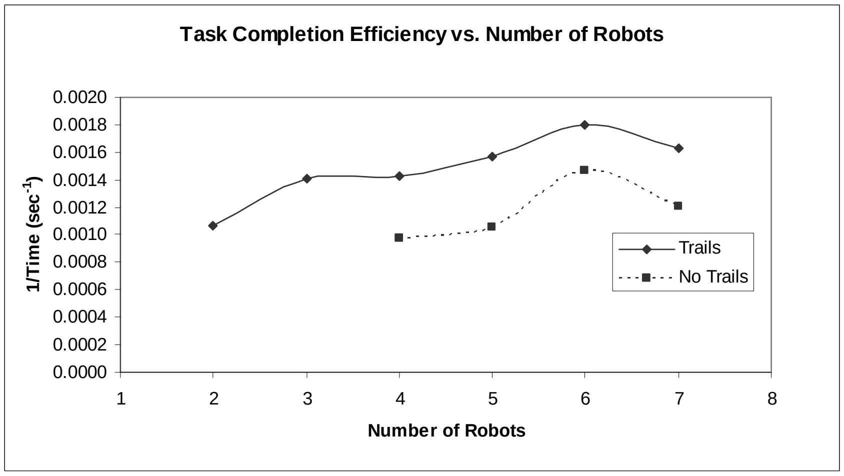


Figure 113. Task efficiency versus number of robots (1 cluster)

Subsection 7.2.2: Experimental Weaknesses

A number of weaknesses were present in the operation of the experiments. Despite the general success of the experiments, there will always be possible improvements in a system of real robots. Perfection cannot be obtained while working in the real world and dealing with the uncertainty and unpredictability of reality.

Due to spatial and financial limitations, the scale of the experiments was limited to seven robots. This number is adequate for demonstrating a limited set of behaviors. However, it is insufficient for thorough investigation of behaviors requiring large group size. Swarm intelligence in biological system relies on *multiple* communicative interactions between individuals, making it difficult to produce swarm intelligence with such a small swarm population.

Experimental errors invariably have some effect on the results of an experiment. Due to unpredictability in the operation of each robot, the system behavior demonstrated a significant degree of variance between trials. For example, robots did not follow trails with an extremely high degree of accuracy. Targets were sometimes passed over without being collected. Robots occasionally became deadlocked at a corner of the foraging field or interlocked with another robot. These types of errors resulted in inconsistent data exhibiting a moderately high degree of variability between tests.

The volume of data collected is another element of experimentation that could be improved by future research. Financial limitations made it impossible to run more than two tests for each data point. With only two trials per data point, it is difficult to perform meaningful statistical analysis. Furthermore, the means obtained from two trials are obviously less valid than those consisting of a high number of tests. Despite limited volume of data, a number of trends were identified and significant goals were met.

Subsection 7.2.3: Achievement of Experimentation Goals

In conclusion, the degree to which stated goals were accomplished will be discussed. The goals of experimentation, as outlined in Section 6.1, are reproduced here for reference:

- To develop a system of robots capable of utilizing trail-laying and following in completion of a central-place foraging task.
- To introduce a novel trail substance that is non-toxic, time-decaying, and easily deposited and sensed.
- To demonstrate characteristics of a swarm-intelligent system

    - Emergent pattern formation
    - Multistability
    - Increased task efficiency due to cooperation
- To compare experimental task efficiency to theoretical predictions and simulation results

The first two goals amount to successful proof of concept by means of designing a functional system of foraging robots. These goals were accomplished, as the robots performed cooperatively using mass recruitment via disappearing ink trails. A novel trail substance was utilized that is non-toxic and time-decaying. The main drawback to this trail substance is flammability and reactivity. The results of experimentation successfully demonstrated some elements of self-organization and swarm intelligence. Emergent pattern formation occurred on a limited scale, taking the form of well-defined ink trials to a cluster of targets. Because of the small number of robots, demonstration of multistability was inconclusive. It was not possible for seven robots to concurrently exploit two or more clusters. The only sense in which multiple stable states were observed is in the case of one cluster with three or four robots. In those trials, the robots were sometimes unable to establish and maintain trails; instead, the system operated in an essentially uncooperative state. The same trials sometimes demonstrated successful utilization of trail-laying and following. The two states—cooperative and non-cooperative foraging—existed simultaneously, a possible example of multistability. Task efficiency clearly improved as a function of cooperation, as shown in Figure 113. Overall, the experimental goals were accomplished, with the exception of demonstrating multistability.

Section 7.3: Comparison of Results to Theoretical Predictions

Chapter 4 summarized the most substantial contributions to central-place foraging theory related to mass recruitment. In Section 4.6, a number of theoretical predictions were listed. These predictions follow directly from the discussion of mathematical foraging models. The results of simulation and experimentation have been presented and analyzed from a narrow perspective, but it is essential to relate the results to the current body of theoretical work in the field of swarm intelligence. This section describes the extent to which theoretical predictions were borne out by simulation and experimentation. The hypotheses presented in Section 4.6 were classified according to the three main characteristics of self-organized behavior. They are reproduced below, with a brief discussion of the veracity of each prediction with respect to experimental and simulated results.

- Emergent Pattern Formation
    - *A critical density of foragers is required to generate some trail patterns.* This attribute of pattern formation occurred in StarLogo simulations. Below a certain critical density, which depends on the trail decay rate and number of food sources, the virtual ants were unable to maintain pheromone trails. A similar result occurred in the experiments with real robots. For $N < 3$, the robots were unable to establish a stable trail.
    - *Formation of an established trail (or trails) may follow an initial "disordered phase" of unpredictable duration.*

A “disordered phase” was observed in some of the time evolution plots of food collection from simulation (e.g., Figure 35). In some trials, the colony exploited multiple sources before collectively choosing to focus on a subset of the total number of sources. In experiments, this characteristic was difficult to observe, due to small population size and short time scales.

- *The nature of the trail pattern depends on parameters such as the number of foragers, food distribution, pheromone decay, etc.*

A number of foraging exploitation patterns were observed in simulation. These patterns demonstrate dependency on group size, food distribution, and pheromone decay, as discussed in Section 5.5 and above in Section 7.1. The foraging robots were only able to establish one stable trail at a time. The existence of the one stable trail was dependent on group size. Pheromone decay rate was not varied as a parameter.

- *Trail formation relies on autocatalysis of random fluctuations in the paths of searching foragers*.

Simulation results confirmed this hypothesis. For example, in trials where the colony chose to forage preferentially from two equidistant food sources, each source was chosen with equal probability. The preferred source was eventually chosen in each case from reinforcement of random fluctuations. A similar situation occurred in the formation of a single trail with real robots. A single trail formed either more slowly or not at all due to the stochastic component of robot searching.

- Multistability

    - *When multiple food sources are present, a number of stable states may exist, ranging from exclusive exploitation of one source to uniform exploitation of all available sources.*

    This prediction was clearly demonstrated to be accurate in simulation, especially in cases with three or four food sources. It could not be investigated with real robots, due to a limited population. The robots were unable to exploit two sources simultaneously.

    - *When multiple food sources are present, under certain conditions, only one stable state exists.*

    This hypothesis was confirmed in both simulation and experimentation. In simulation, the virtual ant colony could only exploit one source at a time for small group size and/or rapid pheromone decay. With two sources present, the foraging robots always exploited only a single source.

    - *The foraging group may collectively choose a suboptimal stable state.*

    This type of behavior occurred exclusively in simulation. When a second food source was presented closer to home during a simulation trial, the colony sometimes continued exploiting the more distant source (see Figure 48).

- Bifurcation Phenomena
    - *Variations in key system parameters may correspond to dramatically different food source exploitation schemes.*

    The results of StarLogo simulations show conclusively that system behavior exhibits sensitive dependence on pheromone decay rate and group size. Slight variations in either parameter correspond to significant alterations in foraging exploitation strategy.

- *Oscillations in certain parameters (especially in the number of active foragers) may allow a system near a bifurcation point to switch from a suboptimal to an optimal solution.*

Figure 46 and Figure 47 are two examples of a simulated system switching to an optimal foraging strategy. While parameters were not intentionally caused to oscillate during these trials, the system dynamics may have caused the number of “active” foragers to oscillate in the vicinity of a bifurcation point (refer to [86] for a detailed discussion of active/inactive foragers in biological systems). Active foragers are those searching the foraging field for a new food source (not following a trail or carrying food to home).

Chapter 8: Summary and Conclusions

The topic of swarm-intelligent robotics has been investigated from an experimental perspective in this thesis. This field of research emerged from the broader field of cooperative mobile robotics, as discussed in Chapter 2. Swarm intelligence was discussed in a conceptual framework of cooperative robotics and the historical foundations of the field were reviewed. In Chapter 3, the biological inspirations of swarm intelligence were presented by discussing self-organization in a variety of biological systems. Next, foraging theory was discussed in detail in Chapter 4. Three important types of mathematical models were introduced, including a detailed development of at least one model from each category. Chapters 5 and 6 describe investigations of swarm-intelligent foraging via computer simulations and experimentation with real robots, respectively. Finally, in Chapter 7, a synthesis of the results is provided, along with a discussion of their relationship to the predictions of theoretical foraging models.

In Chapter 1, the chief aims of the thesis were presented. In conclusion, it is essential to evaluate the degree to which the goals were achieved. The primary objective was to apply swarm intelligence theory to the development of a robotic foraging system. The conjecture was made that such a system could be developed with the following constraints: (1) no direct communication between robots, (2) decentralized control, and (3) behavior-based control. It has been demonstrated that the system described in Chapter 6 successfully obeys these constraints and accomplishes the task of cooperative foraging through mass recruitment. While conclusive demonstration of swarm intelligence in the robotic system was not achieved, cooperative foraging represents an important step along the path toward a conclusively swarm-intelligent robotic system. The secondary goal was to utilize computer simulations to enhance conceptual understanding of swarm intelligence and investigate system dependence on group size. Visualization produced by computer simulations contributed significantly to a deeper understanding of the mechanisms of swarm intelligence. Importantly, the simulated swarms exhibited conclusively swarm-intelligent improvement in task efficiency when compared to non-cooperative groups. In summary, the research provided herein fully defends the primary conjectures put forth in Chapter 1.

This work offers three primary novel and substantial contributions to the swarm intelligence research community. First, this is the first experimental demonstration of a multi-robot system capable of cooperative foraging by means of trail-laying and following. The group of robots complete the foraging task more efficiently with mass recruitment than without cooperation. The second contribution closely follows from the first one. A novel trail-laying and following system featured "disappearing ink," a simple chemical solution sensed optically. This type of trail performed well in experimentation, and exists as a promising substance for future utilization in robotic systems. Finally, the StarLogo computer simulations represent the third substantial and novel contribution. While computer simulations of ant colonies have been performed, this work includes a broad analysis of performance dependent on group size and pheromone decay rate. In addition, the behavioral complexity exhibited with three, and especially four, food sources is an original result. Combined, these three original contributions form the core of this research effort.

In closing, a number of suggestions are provided for future research. As in many scientific investigations, this effort raises as many new questions as it answers. A number of theoretical issues remain to be explored. A comprehensive mathematical model of foraging via mass recruitment does not exist. Development of a new model or extension of current models is an important area requiring further study. In particular, future modeling work could focus on the importance of pheromone decay rate. Also, a comprehensive model would necessarily describe the behavioral complexity possible in the presence of multiple food sources. Theoretical models should also be developed from within the robotics community, making theoretical results more pertinent to robotic implementation of swarm intelligence. As experimental work with real robots is still an emerging field, numerous research possibilities exist for the future. Systems must be developed with large numbers of robots in wide spatial foraging fields in order to study a wider range of foraging phenomena. Considerable work remains in developing effective trail substances for a variety of indoor or outdoor applications. Furthermore, a number of other communication modes exist, and not all rely solely on stigmergy. Investigating other forms of indirect or direct communication would be a valuable contribution to the research community. Finally, it is vital that researchers begin testing robotic swarms in real-world, dynamic environments. A great deal of work remains in that direction before the bold claims regarding real-world applications of swarm intelligence are realized.

APPENDIX A – StarLogo Documentation

The following material is excerpted from [110] See that reference for more complete documentation on StarLogo v. 2.0, which was used in this research.

INTRODUCTION

StarLogo is a programmable modeling environment for exploring the behaviors of decentralized systems, such as bird flocks, traffic jams, and ant colonies. It is designed especially for use by students.

In decentralized systems, orderly patterns can arise without centralized control. Increasingly, researchers are choosing decentralized models for the organizations and technologies that they construct in the world, and for the theories that they construct about the world. But many people continue to resist these ideas, assuming centralized control where none exists--for example, assuming (incorrectly) that bird flocks have leaders. StarLogo is designed to help students (as well as researchers) develop new ways of thinking about and understanding decentralized systems.

StarLogo is an extension of the Logo programming language. With traditional versions of Logo, you can create drawings and animations by giving commands to a graphic "turtle" on the computer screen. StarLogo extends this idea by allowing you to control thousands of graphic turtles in parallel. In addition, StarLogo makes the turtles' world computationally active: you can write programs for thousands of "patches" that make up the turtles' environment. Turtles and patches can interact with one another. For example, you can program the turtles to "sniff" around the world, and change their behaviors based on what they sense in the patches below. StarLogo is particularly well-suited for modeling complex decentralized systems--systems which traditionally have not been available to people without advanced mathematical or programming skills.

This document provides a short introduction to the Java version of StarLogo. It describes the basic features of StarLogo, and it suggests initial activities for using and learning StarLogo. At the end are pointers to other documents that provide a more complete description of the StarLogo language.

CAST OF CHARACTERS

StarLogo includes three main types of "characters":

Turtles. The main inhabitants of the StarLogo world are graphic creatures known as "turtles." You can use a turtle to represent almost any type of object: an ant in a colony, a car in a traffic jam, an antibody in an immune system, a molecule in a gas. Each turtle has a position, a heading, a color, and a "pen" for drawing. You can add more specialized traits and properties. In StarLogo (unlike traditional versions of Logo), you can control the actions and interactions of *thousands* of turtles in parallel.

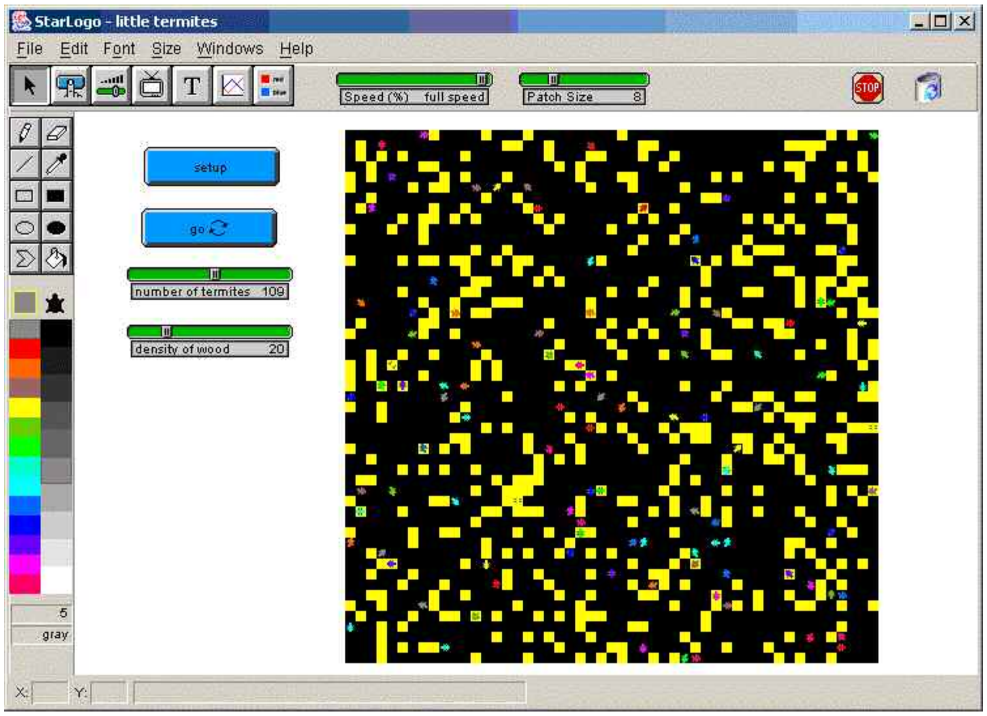
Patches. Patches are pieces of the world in which the turtles live. Patches are not merely passive objects upon which the turtles act. Like turtles, patches can execute StarLogo commands, and they can act on turtles and patches. Patches are arranged in a grid, with each patch corresponding to a square in the Graphics area.

Observer. The observer "looks down" on the turtles and patches from a birdâs eye perspective. The observer can create new turtles, and it can monitor the activity of the existing turtles and patches.

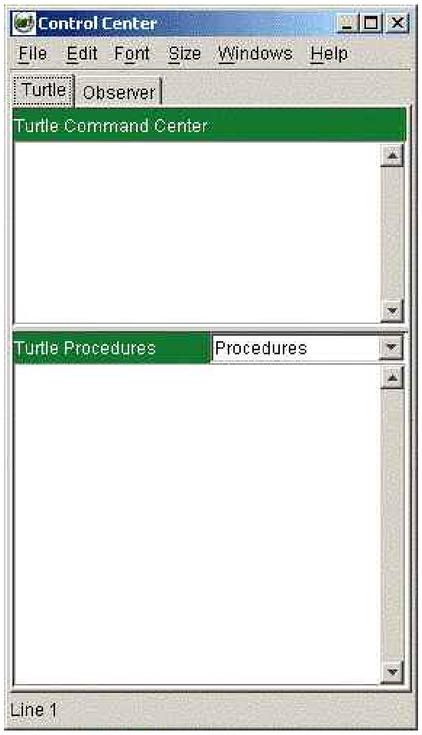
STARLOGO USER INTERFACE

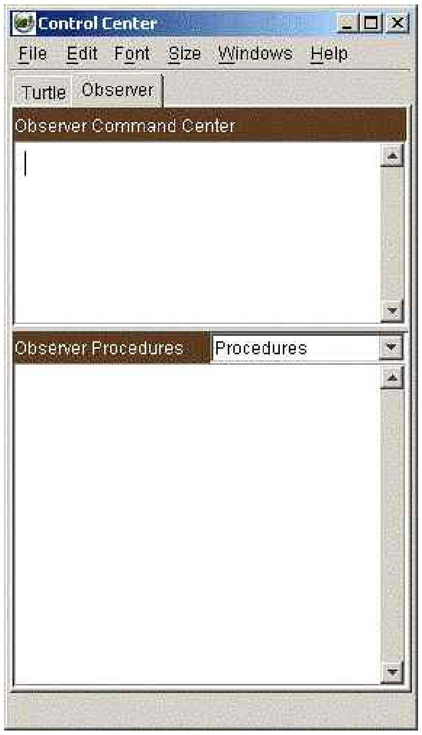
The StarLogo interface consists of several main windows:

***StarLogo window*:** This window is divided into several sections. The initially black Graphics area is where the StarLogo turtles move and draw. The turtles move on top of a grid of patches. When a turtle moves off the screen, it "wraps" around to the other side. You can move a turtle directly by dragging it with the mouse. The white area on the left of the Graphics area is the Interface area. This area is where you will create buttons, sliders, and monitors that allow you to interact directly with StarLogo programs. To create and inspect interface objects (that is, buttons, sliders, and monitors), use the Main Toolbar, located in the gray bar across the top of the StarLogo window. To create a new interface object, choose the appropriate tool from the Toolbar and drag out a rectangle in the Interface area. To inspect the underlying behavior of an existing interface object, choose the appropriate tool from the Toolbar and then click on the object. (You can also inspect an object by holding down control and double-clicking on the object with the standard arrow tool.) The Paint and Color Toolbar appears above the Graphics area when you click the paintbrush icon from the Main Toolbar. From here, you can select which color you want to draw with. In addition, you can select different types of drawing tools.



***Control Center window*:** The Command Center area of the Control Center window is where you type commands for StarLogo. You may run a command again by moving the cursor to that line and pressing return. The Procedures area is where you write your own StarLogo procedures. By clicking back and forth between "Turtle" and "Observer" you can distinguish between those commands which are specific to turtles and those which can only be executed by the observer. Refer to the Reference Manual for a more complete description of the different command types available to each StarLogo character.





***Other windows*:** The menu bar at the top of the Control Center window allows you to access several other windows. The Information window is intended for explanatory notes, commentary, and instructions. The Output window is used for printing and recording data generated by a StarLogo project. The print command prints to this window. The Plot window is where you can create real-time graphs as your StarLogo project is running.

***Turtle Monitors*:** If you double-click on a turtle in the Graphics area, a Turtle Monitor will appear, showing the turtle's variables, both standard and user-defined, and their values. The values update in real-time as your StarLogo project is running. You can also use the Turtle Monitor to directly change the value (state) of the variable.

HOW TO RUN STARLOGO

To use StarLogo, you will need to have a version of Java 1.1 (with Swing) or higher installed on your computer.

Our installer for Windows includes a Java 1.4 runtime from Sun Microsystems. This version will not interfere with any other versions of Java installed on your machine, and you won't have to worry about compatability. We do not use or require any other version of Java to be installed on your computer.

The Mac OS 9 version requires MRJ 2.2.5 or higher. If you don't yet have MRJ 2.2.5 (or are not sure) and have Mac OS 9, use the Software Update control panel to install any updates from Apple. One of these will be MRJ 2.2.5 or higher. If you use Mac OS 8.1 or higher, you can download MRJ from Apple's Java web site at http://developer.apple.com/java/download.html.

We support MacOS X 10.1 and higher. If you get the Java Update 1.3.1 for OS X 10.1, be advised that multi-monitor support is broken. There are still quite a few problems with MacOS X Java, even on 10.2, so feel free to email at bug-starlogo@media.mit.edu when bad things happen.

On Unix, we recommend using your vendor's Java, as long as it is compatible with Java 1.2 or higher. We include a shell script to run StarLogo, and include a special shell script for Solaris. On Linux, we recommend Sun's JDK over Blackdown because of far fewer bugs in the AWT implementation. If you can get Sun's JDK 1.4, that'll be even less buggy. A note: there are interactions between some window managers and Java which cause new windows to open in a really small size. You might try a different window manager first, to see if it is the problem, but if not, tell your window manager to allow you to place the window manually when it pops up, rather than automatically. This will give you the opportunity to resize it.

After you have successfully installed StarLogo, follow these instructions to run StarLogo on your computer:

On a Windows PC

- In the Start Menu, look for the StarLogo 2.0 program group. Click on the StarLogo application file to run StarLogo.
- StarLogo should start up and pop two windows up on the screen. If you don't see them, they might be hidden behind other windows. Check your taskbar for them.

On a Mac

- Make sure that you have MRJ 2.2.5 or higher (from http://developer.apple.com/java/download.html) installed on your Macintosh before trying to run StarLogo.
- Double-click on the StarLogo application.
- StarLogo should start up and pop two windows up on the screen.

On Unix

- Install your favorite Java 1.2.x, or Java 1.3.x, Java 1.4.x.
- Go to the StarLogo distribution directory.
- Execute the unix-run-starlogo shell script. (Note: If you use Solaris, execute the solaris-run-starlogo shell script instead).
- Note: We've seen that StarLogo requires that you set Java to use a larger stack size. This is necessary with Sun's Java; we're not sure whether this is necessary on other Unixes.

APPENDIX B – StarLogo Simulation Code

The StarLogo simulation code (with trails, two food sources) is presented below. Minor changes were made to the program for other simulation scenarios (e.g., three or four food sources). Comments are set off with semicolons along with right-hand column.

Observer Procedures

```
;Declare variables
turtles-own [follow-time]
patches-own [pheromone]
globals [collide]
setcollide 0

;Defines simulation setup
to setup
clear-all                                ;Clear screen
setup-field                              ;Starts setup-field procedure
crt turtles                              ;Creates turtles
ask-turtles [setc green fd 5]            ;Sets color green, forward
ask-patches [setpheromone 0]             ;Sets pheromone = 0 for patches
end

;Sets up boundary and home
to setup-field
create-and-do 1 [setc red                ;Create one turtle
fd screen-half-height rt 90 pd           ;Draw red boundary
fd screen-half-width rt 90
fd screen-height - 1 rt 90
fd screen-width - 1 rt 90
fd screen-height - 1 rt 90
fd screen-half-width]
ct                                       ;Clear turtle
create-and-do 30 [setc brown pd fd 2]    ;Create 30, draw home
ct                                       ;Clear turtles
create-and-do 50 [setc yellow            ;Create 50, draw food
          setxcor -38 setycor 30 pd fd 3 bk 4 pu
          setxcor 25 setycor -20 pd fd 3]
ct                                       ;Clear turtles
end

;Causes pheromone decay
to decay
ask-patches [setpheromone pheromone - 5] ;Decrements pheromone
wait 100 / decay-rate                    ;Wait
decay                                    ;Decay again
```

```
end

;Updates color of patches
to color-patches
ask-patches[
if pheromone < 0 [setpheromone 0]            ;Reset pheromone to min
if pheromone > 95 [setpheromone 95]          ;Reset pheromone to max
if pheromone = 0 and pc > 59 and pc < 62 [setpc 0]  ;Update to black
if pc > 59 and pc < 72                       ;Update to shade of green
[setpc (pheromone / 10) + 59.5]]
wait 1                                       ;Wait
color-patches                                ;Color again
end
```

Turtle Procedures

```
; Defines search behavior for each ant
to search
setc green                                   ;Set color to green
seth heading + 30 - random 60                ;Set random heading
step                                         ;Move forward one unit
wait 1 / srch-spd                            ;Wait
if pc = red                                  ;If patch is red...
[rt 180 step]                                ;turn 180 degrees and step
if pc = yellow                               ;If patch is yellow...
[go-home]                                    ;start go-home procedure
if pc > 59 and pc < 70                       ;If patch is green...
[setfollow-time 0 setc orange follow]        ;follow trail
if count-turtles-here > 1 [setcollide collide + 1] ;Keep track of collision
search                                       ;Search again
end

; Defines behavior to go home
to go-home
seth towards 0 0                             ;Set heading toward home
ifelse pc-ahead = yellow                     ;If yellow patch is ahead
[stamp black]                                ;color it black
[setpheromone-at 0 0 pheromone + 10          ;increment pheromone
if pheromone > 95 [setpheromone 95]          ;reset if over max value
stamp (pheromone / 10) + 59.5]               ;color shade of green
step                                         ;Move forward one unit
if count-turtles-here > 1 [setcollide collide + 1]        ;Keep track of collisions
wait 1 / srch-spd                            ;Wait
repeat 100                                   ;Repeat following commands 100 times
[if pc = black                               ;If patch is black
[setpheromone-at 0 0 pheromone + 10          ;increment pheromone
stamp (pheromone / 10) + 59.5]                ;color patch green
if pc > 59 and pc < 72                        ;If patch is green
[setpheromone-at 0 0 pheromone + 10           ;increment pheromone
stamp (pheromone / 10) + 59.5]                ;color patch green
if pc = brown                                 ;If patch is brown
[seth heading + 200 - random 40               ;Set heading approx. 180 degrees
search]                                       ;Resume search
seth towards 0 0                              ;Set heading toward home
seth heading + 25 - random 50                 ;Set random heading
wait 1 / srch-spd                             ;Wait
step]                                         ;Move forward one step
search                                        ;Resume search
end

; Define path following behavior
to follow
setfollow-time follow-time + 1                ;Increment follow-time
if follow-time > 6 [step search]              ;Resume search if lost
setc orange                                   ;Set color orange
repeat 36                                     ;Repeat commands 36 times
[rt 10 - random 2                             ;Turn random angle
if pc-ahead = yellow                          ;If yellow patch ahead...
[step setc green go-home]                     ;Start go-home
if pc-ahead > 59 and pc-ahead < 70 and        ;If patch ahead is green and farther
fromhome
xcor ^ 2 + ycor ^ 2 <
(xcor + sin (heading)) ^ 2 + (ycor + cos (heading)) ^ 2
[step                                         ;Move forward one unit
if count-turtles-here > 1 [setcollide collide + 1]]]      ;Keep track of collisions
follow                                        ;Follow again
end
```

APPENDIX C – Bricx Command Center Documentation

The following material is excerpted from [113] and is written by John Hansen. BricxCC, version 3.3, was used in this research.

Bricx Command Center was written to more easily work with the Lego MindStorms and CyberMaster robot system. It is built around NQC (Not Quite C Compiler), written by Dave Baum, that makes it possible to program the RCX, the Scout, and the Cybermaster programmable bricks in a language that is close to C. For more information on NQC, see the tutorial and NQC reference guide provided.

The basis of Bricx Command Center consists of an editor in which you can write your NQC programs with the integrated ability to compile the programs and download them to the brick. In case of errors they are reported at the bottom of the editor window such that they can be corrected easily. Besides the editor, Bricx Command Center contains tools to control your robot directly, to watch what is happening in the brick, to download (new) firmware, etc.

To run Bricx Command Center it is not necessary to have spirit.ocx (the ActiveX control used with the RIS 1.0 and 1.5 sets) registered on your system.

Bricx Command Center can be used free of charge. Please note that NQC and it documentation are copyrighted by Dave Baum. Bricx Command Center supports version 2.3r1 or later of NQC. (This version is NOT compatible with version 1.x. You can though set the program in compatibility mode in the Preference menu. There is also an option on the Edit menu to translate an old style program into a new style program.)

APPENDIX D – Not Quite C (NQC) Documentation

The following information is excerpted from Dave Baum's "NQC Programmer's Guide." The complete version of this guide can be found in [114]. Some sections have been omitted for the sake of brevity. This is meant to be a basic introduction to the programming language.

Introduction

NQC stands for Not Quite C, and is a simple language for programming several LEGO MINDSTORMS products. Some of the NQC features depend on which MINDSTORMS product you are using. This product is referred to as the *target* for NQC. Presently, NQC supports five different targets: RCX, RCX2 (an RCX running 2.0 firmware), CyberMaster, Scout, and Spybotics.

All of the targets have a bytecode interpreter (provided by LEGO) which can be used to execute programs. The NQC compiler translates a source program into LEGO bytecodes, which can then be executed on the target itself. Although the preprocessor and control structures of NQC are very similar to C, NQC is not a general purpose language - there are many restrictions that stem from limitations of the LEGO bytecode interpreter.

Logically, NQC is defined as two separate pieces. The NQC language describes the syntax to be used in writing programs. The NQC API describes the system functions, constants, and macros that can be used by programs. This API is defined in a special file built in to the compiler. By default, this file is always processed before compiling a program.

This document describes both the NQC language and the NQC API. In short, it provides the information needed to write NQC programs. Since there are several different interfaces for NQC, this document does not describe how to use any specific NQC implementation. Refer to the documentation provided with the NQC tool, such as the *NQC User Manual* for information specific to that implementation.

For up-to-date information and documentation for NQC, visit the NQC Web Site at http://www.baumfamily.org/nqc

Program Structure

An NQC program is composed of code blocks and global variables. There are three distinct types of code blocks: tasks, inline functions, and subroutines. Each type of code block has its own unique features and restrictions, but they all share a common structure.

Tasks

The RCX implicitly supports multi-tasking, thus an NQC task directly corresponds to an RCX task. Tasks are defined using the `task` keyword using the following syntax:

```
task name()
{
      // the task's code is placed here
}
```

The name of the task may be any legal identifier. A program must always have at least one task - named "main" - which is started whenever the program is run. The maximum number of tasks depends on the target - the RCX supports 10 tasks, CyberMaster supports 4, and Scout supports 6.

The body of a task consists of a list of statements. Tasks may be started and stopped using the `start` and `stop` statements (described in the section titled *Statements*). There is also a NQC API command, `StopAllTasks`, which stops all currently running tasks.

Functions

It is often helpful to group a set of statements together into a single function, which can then be called as needed. NQC supports functions with arguments, but not return values. Functions are defined using the following syntax:

```
void name(argument_list)
{
      // body of the function
}
```

The keyword `void` is an artifact of NQC's heritage - in C functions are specified with the type of data they return. Functions that do not return data are specified to return `void`. Returning data is not supported in NQC, thus all functions are declared using the `void` keyword.

The argument list may be empty, or may contain one or more argument definitions. An argument is defined by its *type* followed by its *name*. Multiple arguments are separated by commas. All values are represented as 16 bit signed integers. However NQC supports four different argument types which correspond to different argument passing semantics and restrictions:

Type	**Meaning**	**Restriction**
int	pass by value	none
const int	pass by value	only constants may be used
int&	pass by reference	only variables may be used
const int &	pass by reference	function cannot modify argument

Arguments of type `int` are passed by value from the calling function to the callee. This usually means that the compiler must allocate a temporary variable to hold the argument. There are no restrictions on the type of value that may be used. However, since the function is working with a copy of the actual argument, any changes it makes to the value will not be seen by the caller. In the example below, the function `foo` attempts to set the value of its argument to 2.

The second type of argument, `const int`, is also passed by value, but with the restriction that only constant values (e.g. numbers) may be used. This is rather important since there are a number of RCX functions that only work with constant arguments.

The third type, `int &`, passes arguments by reference rather than by value. This allows the callee to modify the value and have those changes visible in the caller. However, only variables may be used when calling a function using `int &` arguments:

The last type, `const int &`, is rather unusual. It is also passed by reference, but with the restriction that the callee is not allowed to modify the value. Because of this restriction, the compiler is able to pass anything (not just variables) to functions using this type of argument. In general this is the most efficient way to pass arguments in NQC.

NQC functions are always expanded as inline functions. This means that each call to a function results in another copy of the function's code being included in the program. Unless used judiciously, inline functions can lead to excessive code size.

Subroutines

Unlike inline functions, subroutines allow a single copy of some code to be shared between several different callers. This makes subroutines much more space efficient than inline functions, but due to some limitations in LEGO bytecode interpreter, subroutines have some significant restrictions. First of all, subroutines cannot use any arguments. Second, a subroutine cannot call another subroutine. Last, the maximum number of subroutines is limited to 8 for the RCX, 4 for CyberMaster, 3 for Scout, and 32 for Spybotics. In addition, when using RCX 1.0 or CyberMaster, if the subroutine is called from multiple tasks then it cannot have any local variables or perform calculations that require temporary variables. These significant restrictions make subroutines less desirable than functions, therefore their use should be minimized to those situations where the resultant savings in code size is absolutely necessary. The syntax for a subroutine appears below:

```
sub name()
{
    // body of subroutine
}
```

Variables

All variables in NQC are of the same type - specifically 16 bit signed integers. Variables are declared using the `int` keyword followed by a comma separated list of variable names and terminated by a semicolon ('`;`'). Optionally, an initial value for each variable may be specified using an equals sign ('=') after the variable name.

Global variables are declared at the program scope (outside any code block). Once declared, they may be used within all tasks, functions, and subroutines. Their scope begins at declaration and ends at the end of the program.

Local variables may be declared within tasks, functions, and sometimes within subroutines. Such variables are only accessible within the code block in which they are defined. Specifically, their scope begins with their declaration and ends at the end of their code block. In the case of local variables, a compound statement (a group of statements bracketed by `{` and `}`) is considered a block.

In many cases NQC must allocate one or more temporary variables for its own use. In some cases a temporary variable is used to hold an intermediate value during a calculation. In other cases it is used to hold a value as it is passed to a function. These temporary variables deplete the pool of variables available to the rest of the program. NQC attempts to be as efficient as possible with temporary variables (including reusing them when possible).

The RCX (and other targets) provide a number of storage locations which can be used to hold variables in an NQC program. There are two kinds of storage locations - global and local. When compiling a program, NQC assigns each variable to a specific storage location. Programmers for the most part can ignore the details of this assignment by following two basic rules:

• If a variable needs to be in a global location, declare it as a global variable.

• If a variable does not need to be a global variable, make it as local as possible. This gives the compiler the most flexibility in assigning an actual storage location.

Statements

The body of a code block (task, function, or subroutine) is composed of statements. Statements are terminated with a semi-colon ('`;`').

Variable Declaration

Variable declaration, as described in the previous section, is one type of statement. It declares a local variable (with optional initialization) for use within the code block. The syntax for a variable declaration is:

`int` *`variables`*`;`

where variables is a comma separated list of names with optional initial values:

`name``[=`*`expression`*`]`

Arrays of variables may also be declared (for the RCX2 only):

`int` *`array`*`[`*`size`*`];`

Assignment

Once declared, variables may be assigned the value of an expression:

variable assign_operator expression;

There are nine different assignment operators. The most basic operator, '=', simply assigns the value of the expression to the variable. The other operators modify the variable's value in some other way as shown in the table below

Operator	**Action**
=	Set variable to expression
+=	Add expression to variable
-=	Subtract expression from variable
*=	Multiple variable by expression
/=	Divide variable by expression
%=	Set variable to remainder after dividing by expression
&=	Bitwise AND expression into variable
|=	Bitwise OR expression into variable
^=	Bitwise exclusive OR into variable
||=	Set variable to absolute value of expression
+-=	Set variable to sign (-1,+1,0) of expression
>>=	Right shift variable by a constant amount
<<=	Left shift variable by a constant amount

Control Structures

The simplest control structure is a compound statement. This is a list of statements enclosed within curly braces ('{' and '}'):

```
{
        x = 1;
        y = 2;
}
```

Although this may not seem very significant, it plays a crucial role in building more complicated control structures. Many control structures expect a single statement as their body. By using a compound statement, the same control structure can be used to control multiple statements.

The `if` statement evaluates a condition. If the condition is true it executes one statement (the consequence). An optional second statement (the alternative) is executed if the condition is false. The two syntaxes for an `if` statement is shown below.

```
if (condition) consequence
if (condition) consequence else alternative
```

Note that the condition is enclosed in parentheses.

The `while` statement is used to construct a conditional loop. The condition is evaluated, and if true the body of the loop is executed, then the condition is tested again. This process continues until the condition becomes false (or a `break` statement is executed). The syntax for a `while` loop appears below:

```
while (condition) body
```

A variant of the `while` loop is the `do-while` loop. Its syntax is:

```
do body while (condition)
```

The difference between a `while` loop and a `do-while` loop is that the `do-while` loop always executes the body at least once, whereas the `while` loop may not execute it at all.

Another kind of loop is the `for` loop:

```
for(stmt1 ; condition ; stmt2) body
```

A for loop always executes stmt1, then it repeatedly checks the condition and while it remains true executes the body followed by stmt2. The for loop is equivalent to:

```
stmt1;
while(condition)
{
      body
      stmt2;
}
```

The `repeat` statement executes a loop a specified number of times:

```
repeat (expression) body
```

The expression determines how many times the body will be executed. Note that it is only evaluated a single time, then the body is repeated that number of times. This is different from both the `while` and `do-while` loops which evaluate their condition each time through the loop.

A `switch` statement can be used to execute one of several different blocks of code depending on the value of an expression. Each block of code is preceded by one or more `case` labels. Each case must be a constant and unique within the switch statement. The switch statement evaluates the expression then looks for a matching case label. It will then execute any statements following the matching case until either a break statement or the end of the switch is reaches. A single `default` label may also be used - it will match any value not already appearing in a case label. Technically, a switch statement has the following syntax:

```
switch (expression) body
```

The case and default labels are not statements in themselves - they are *labels* that precede statements. Multiple labels can precede the same statement.

The `goto` statement forces a program to jump to the specified location. Statements in a program can be labeled by preceding them with an identifier and a colon. A goto statement then specifies the label which the program should jump to. For example, this is how an infinite loop that increments a variable could be implemented using `goto`:

```
my_loop:
      x++;
      goto my_loop;
```

The `goto` statement should be used sparingly and cautiously. In almost every case, control structures such as `if`, `while`, and `switch` make a program much more readable and maintainable than using `goto`. Care should be taken to never use a `goto` to jump into or out of a `monitor` or `acquire` statement. This is because `monitor` and `acquire` have special code that normally gets executed upon entry and exit, and a `goto` will bypass that code – probably resulting in undesirable behavior.

APPENDIX E – NQC Program Code

The following NQC Code was used to control each robot. In the non-cooperative experiments, the"follow" behavior was removed from the program. Comments are set off by double forward slashes. The program is displayed in two columns, read from the bottom of the left column to the top of the right column.

```
// Mark Edelen
// Foraging Robots Thesis Research
// Main Foraging Program
//     Four main behaviors:
//         (1) wander
//         (2) avoid
//         (3) pick up / go home
//         (4) follow

task main ()
{
  // Initialize sensors
  SetSensor (SENSOR_1,
SENSOR_TOUCH);
  SetSensor (SENSOR_3,
SENSOR_LIGHT);
  SetSensorType (SENSOR_2,
SENSOR_TYPE_LIGHT);
  SetSensorMode (SENSOR_2,
SENSOR_MODE_RAW);

  // Initialize counters
  #pragma reserve 0    // Counter for avoid
behavior
  #pragma reserve 1    // Counter for pick-up
behavior
  #pragma reserve 2    // Counter for follow
behavior

  // Clear counters
  ClearCounter (0);
  ClearCounter (1);
  ClearCounter (2);

  // Check battery level
  if (BatteryLevel() < 8850)
  {
    PlaySound (SOUND_LOW_BEEP);
    Wait (100);
    }

  // Create datalog
  CreateDatalog (100);

  // Play Start Note
  PlaySound (SOUND_CLICK);

  // Make sure pen & magnet are up
  SetPower (OUT_B,4);
  OnFor (OUT_B, 30);

  // Start basic behavior
  start wander;
}


task wander ()
{
   // Stop follow task
   stop follow;

  // Define events to monitor while
wandering
  SetEvent (1, SENSOR_1,
EVENT_TYPE_PRESSED);   // touch
sensor
  SetEvent (2, SENSOR_2,
EVENT_TYPE_NORMAL);    // light2
reads washer or line
  SetLowerLimit (2,SENSOR_2 + 10);
  SetUpperLimit (2,720);
  SetEvent (3, SENSOR_3,
EVENT_TYPE_HIGH);      // light3 reads
bright
  SetLowerLimit (3,50);
  SetUpperLimit (3,75);

  // Start wander timer
  SetTimer (1, 0);

  // Display raw value of light2
  SelectDisplay (DISPLAY_SENSOR_2);

  // Start monitoring 3 events
  monitor( EVENT_MASK (1) |
EVENT_MASK (2) | EVENT_MASK (3))
  {
    SetDirection (OUT_A+OUT_C,
OUT_FWD);
    SetPower (OUT_A+OUT_C,2);
    while (true)
    {
      // Recalibrate lower limit every 4 ms
      SetLowerLimit (2,SENSOR_2 + 8);
      OnFor (OUT_A+OUT_C, 4);
    }
  }
  // If touch sensor is pressed, then...
  catch( EVENT_MASK (1) )
  {
    Off (OUT_A+OUT_C);
    // Beep
    PlaySound (SOUND_UP);

    AddToDatalog (Timer (1));
    start avoid;

  }
  // If robots reads non-white, then...
  catch( EVENT_MASK (2) )
  {
    Wait(4);
    Off (OUT_A+OUT_C);
    // Beep
    PlaySound (SOUND_DOUBLE_BEEP);
    Wait(100);
    if (SENSOR_2 > 700)          // over 700 is
the washer
    {
      // Record wandering time
      AddToDatalog (Timer (1));
      // Pick up washer
      start pick_up;
    }
    else
    {
      // Increment counter for follow behavior
      IncCounter (2);
      // Add counter value to datalog
      AddToDatalog (20);
      AddToDatalog (Counter (2));
      // Record wandering time
      AddToDatalog (Timer (1));
      // Follow the trail
      start follow;
    }
  }
  // If light3 reads bright (robot is close to
home), then...
  catch(EVENT_MASK(3))
  {
    Off (OUT_A+OUT_C);
    // ...avoid home
    start avoid;
  }
}


task avoid ()
{
  // Start avoid timer
  SetTimer (2, 0);

  // Increment avoid counter
  IncCounter (0);
  AddToDatalog (0);
  AddToDatalog (Counter (0));

  // Monitor for touch while avoiding
  monitor (EVENT_MASK (1))
  {
    // Back up
    SetDirection
(OUT_A+OUT_C,OUT_REV);
    OnFor (OUT_A+OUT_C, 20);
    // Rotate random angle
    SetDirection (OUT_C, OUT_FWD);
    SetPower (OUT_A+OUT_C, 4);
    OnFor (OUT_A+OUT_C,
40+Random(80));
    AddToDatalog (Timer (2));
    start wander;
  }
  catch
  {
    start avoid;
  }
}
```

```
task pick_up ()
{
  // Go fwd to get over washer
  OnFor (OUT_A+OUT_C, 35);

  // Lower magnet to pick up washer
  SetPower (OUT_B,6);
  SetDirection (OUT_B, OUT_REV);
  OnFor (OUT_B, 75);
  Wait (50);

  // Increment counter1
  IncCounter (1);
  AddToDatalog (10);
  AddToDatalog (Counter (1));

  // Back up to clear other magnets
  SetPower (OUT_A+OUT_C, 2);
  SetDirection (OUT_A+OUT_C,
OUT_REV);
  OnFor (OUT_A+OUT_C, 30);
  Wait (50);
  Off (OUT_A+OUT_C);

  // Start to go home
  start go_home;
}

task go_home ()
{

  //Display value of light3
  SelectDisplay (DISPLAY_SENSOR_3);

monitor (EVENT_MASK (1))
{

  // Rotate to find initial home direction
  SetDirection (OUT_C, OUT_FWD);
  SetDirection (OUT_A, OUT_REV);
  SetPower (OUT_A+OUT_C, 4);
  do
  {
    OnFor (OUT_A+OUT_C, 10);
    Wait (60);
  }
  while (SENSOR_3 < 54);
```

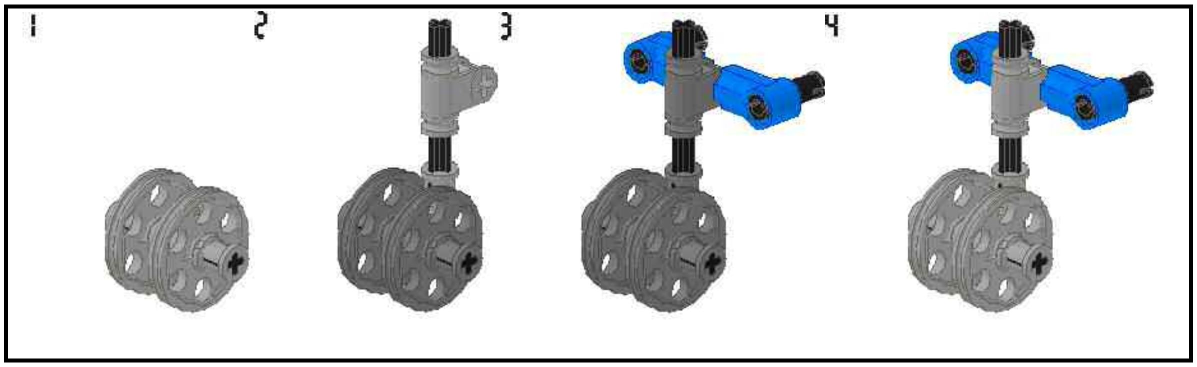
```
  // Lower pen
  SetPower (OUT_B, 8);
  OnRev (OUT_B);

  Off (OUT_A+OUT_C);
  SetDirection (OUT_A+OUT_C,
OUT_FWD);
  SetPower (OUT_A+OUT_C, 2);
  On (OUT_A+OUT_C);
  Wait(30);

  repeat(5)
    {
    // Search for home until light is bright
    SetPower (OUT_C, 1);
    On(OUT_C);
    Wait(100);
    Float(OUT_C);
    Wait(75);

    if (SENSOR_3 > 90) break;
    }


  // Go fwd, lift pen & magnet, go back
  PlaySound (SOUND_CLICK);
  //   Go fwd to get close to home
  SetPower(OUT_A+OUT_C,2);
  OnFor (OUT_A+OUT_C, 70);
  //   Lift magnet
  Off (OUT_B);
  SetDirection (OUT_B, OUT_FWD);
  On (OUT_B);
  Wait (15);
  //   Go back
  SetDirection(OUT_A+OUT_C,
OUT_REV);
  OnFor (OUT_A+OUT_C,100);
  //   Turn
  SetDirection(OUT_C, OUT_FWD);
  SetPower(OUT_A+OUT_C,4);
  On(OUT_A+OUT_C);
  Wait (75 + Random(40));
  Off (OUT_A+OUT_B+OUT_C);

  // Resume wandering
  start wander;
}
catch
{
  Off(OUT_A+OUT_B+OUT_C);
  Wait(300);
  start go_home;
}
}


task follow ()
{
  // Start follow timer
  SetTimer (3, 0);

  //Display raw value of light2
  SelectDisplay (DISPLAY_SENSOR_2);

  monitor (EVENT_MASK (1))
  {
  // Cross over line
  SetDirection (OUT_A+OUT_C,
OUT_FWD);
  SetPower (OUT_A+OUT_C, 2);
  OnFor (OUT_A+OUT_C, 25);
  Wait(100);

  // Turn left small amount to turn caster
  SetDirection (OUT_A, OUT_REV);
  SetPower (OUT_A+OUT_C, 3);
  OnFor (OUT_A+OUT_C, 15);

  // Turn left until light2 sees line
  int floor1;
  int floor2;
  int j;
  j=0;
  do
  {
    floor1 = SENSOR_2;
    OnFor (OUT_A+OUT_C, 5);
    Wait (35);
    floor2 = SENSOR_2;
    j++;
    if (j>30) {PlaySound (SOUND_CLICK);
start wander;}
  }
  while (floor2 <= floor1 + 5);
  Wait(100);

  // Take light reading
  int light_left;
  light_left = SENSOR_3;

  // Turn right small amount to get off line
  Toggle (OUT_A+OUT_C);
  SetPower (OUT_A+OUT_C, 4);
  OnFor (OUT_A+OUT_C, 50);
  SetPower (OUT_A+OUT_C, 3);
  Wait(100);

  // Turn right until light2 sees line
  j=0;
  do
  {
    floor1 = SENSOR_2;
    OnFor (OUT_A+OUT_C, 5);
    Wait (35);
    floor2 = SENSOR_2;
    j++;
    if (j>25) start wander;
  }
  while (floor2 <= floor1 + 5);
  Wait(100);

  // Take light reading
  int light_right;
  light_right = SENSOR_3;

  // Estimate distance from home and assign
number of turns
  int light_home;
  if (light_right < light_left)
  {
  light_home = light_right;
  }
  else
  {
  light_home = light_left;
  }
  int dist_home;
  if (light_home > 77 && light_home <= 85)
dist_home = 8;
  if (light_home > 70 && light_home <= 76)
dist_home = 6;
  if (light_home > 65 && light_home <= 70)
dist_home = 5;
  if (light_home > 52 && light_home <= 64)
dist_home = 4;
  else dist_home = 3;

  // Compare light readings
  if (light_right < light_left)
  // Follow line with line on right
  { repeat(dist_home + 1)
    {// Turn left to get off line
    SetPower (OUT_A+OUT_C, 4);
    SetDirection (OUT_A+OUT_C,
OUT_FWD);
    Off (OUT_A);
    OnFor (OUT_C, 30);
    // Turn right until line
    Off (OUT_C);
    j=0;
    do
    {
      floor1 = SENSOR_2;
      OnFor (OUT_A, 4);
      Wait (20);
      floor2 = SENSOR_2;
      j++;
      if (j>30) start wander;
    }
    while (floor2 <= floor1 + 4);
    }
  }

  else
  {
  // Follow line with line on left
  PlaySound (SOUND_FAST_UP);

    // Turn left small amount
  Toggle (OUT_A+OUT_C);
  OnFor (OUT_A+OUT_C, 55);
```

```
  SetPower (OUT_A+OUT_C, 4);
  SetDirection (OUT_A+OUT_C,
OUT_FWD);
  repeat(dist_home + 1)
    {
    // Turn left until line
    Off (OUT_A);
    j=0;
    do
    {
      floor1 = SENSOR_2;
      OnFor (OUT_C, 4);
      Wait (20);
      floor2 = SENSOR_2;
      j++;
      if (j>30) start wander;
    }
    while (floor2 <= floor1 + 4);
    // Turn right to get off line
    Off (OUT_C);
    OnFor (OUT_A, 30);
    }
   }
}
  catch
  {
    AddToDatalog (Timer (3));
    start avoid;
  }
AddToDatalog (Timer (3));
start wander;
}
```

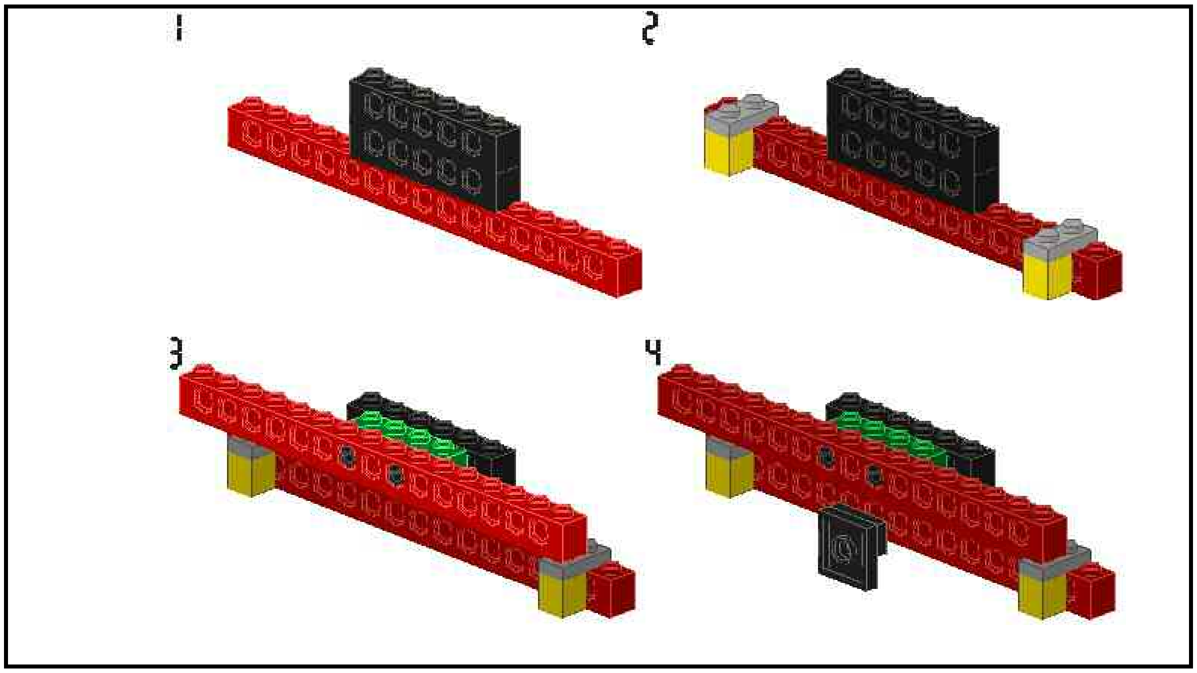
APPENDIX F – LEGO Robot Assembly Instructions

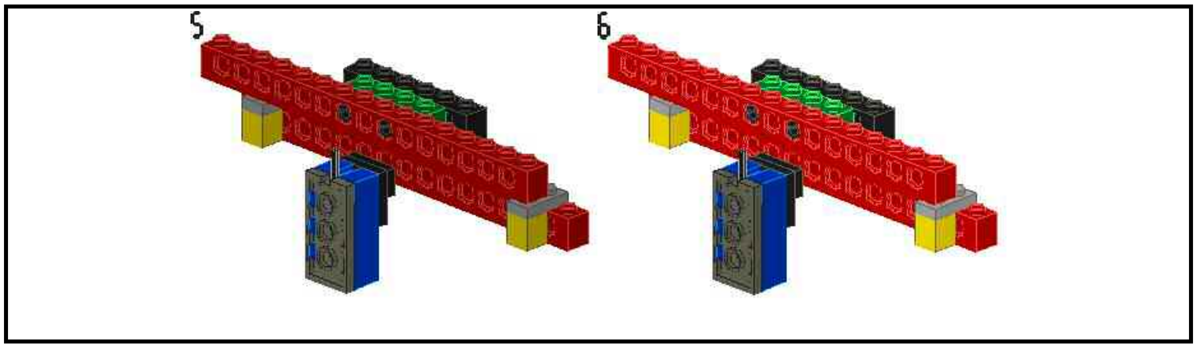
In the following pages, the assembly instructions are provided, pictorially, for each individual subsystem of the robot. After the assembly of each subsystem, instructions are shown for combining the subsystems into a completed robot. These figure were generated using MLCAD v.3.0 (© 2002 by Michael Lachmann), a computer-aided design package specifically designed for use with LEGO constructions [120].

CASTER WHEEL ASSEMBLY

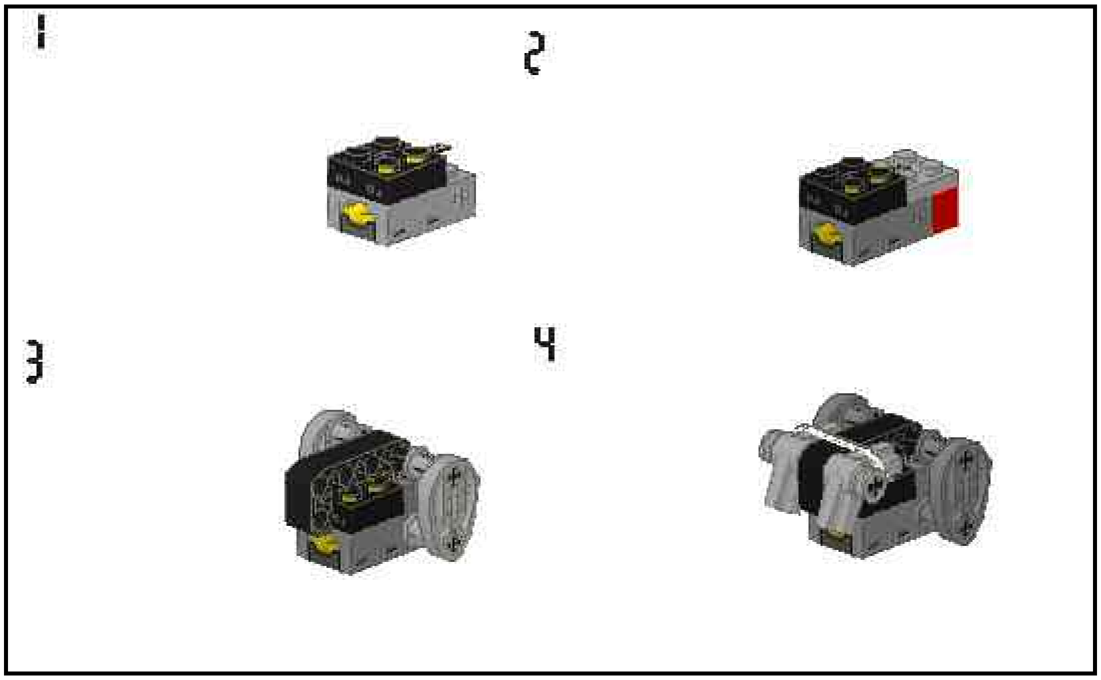


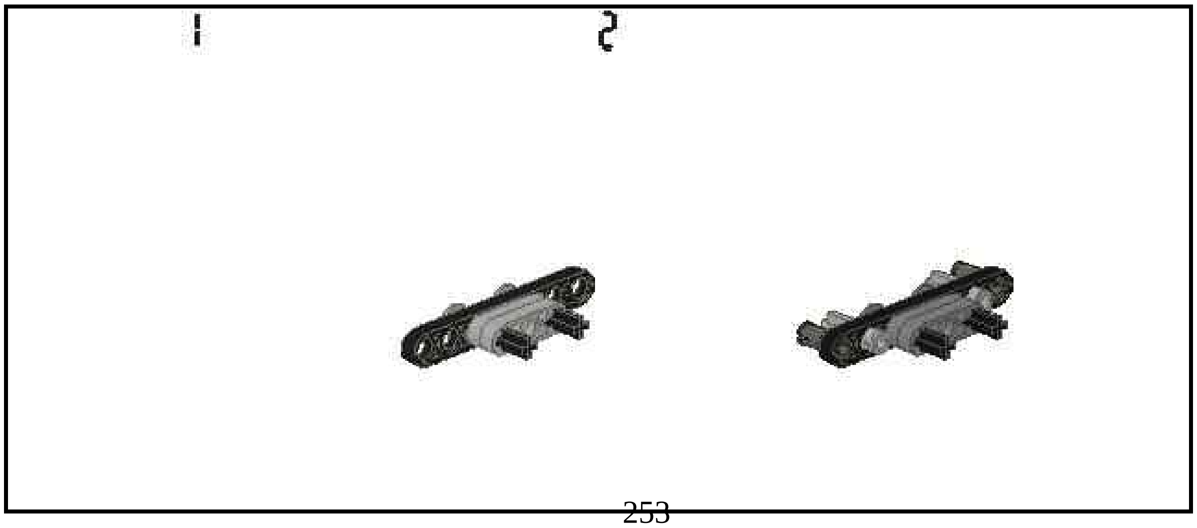
FRONT FRAME ASSEMBLY

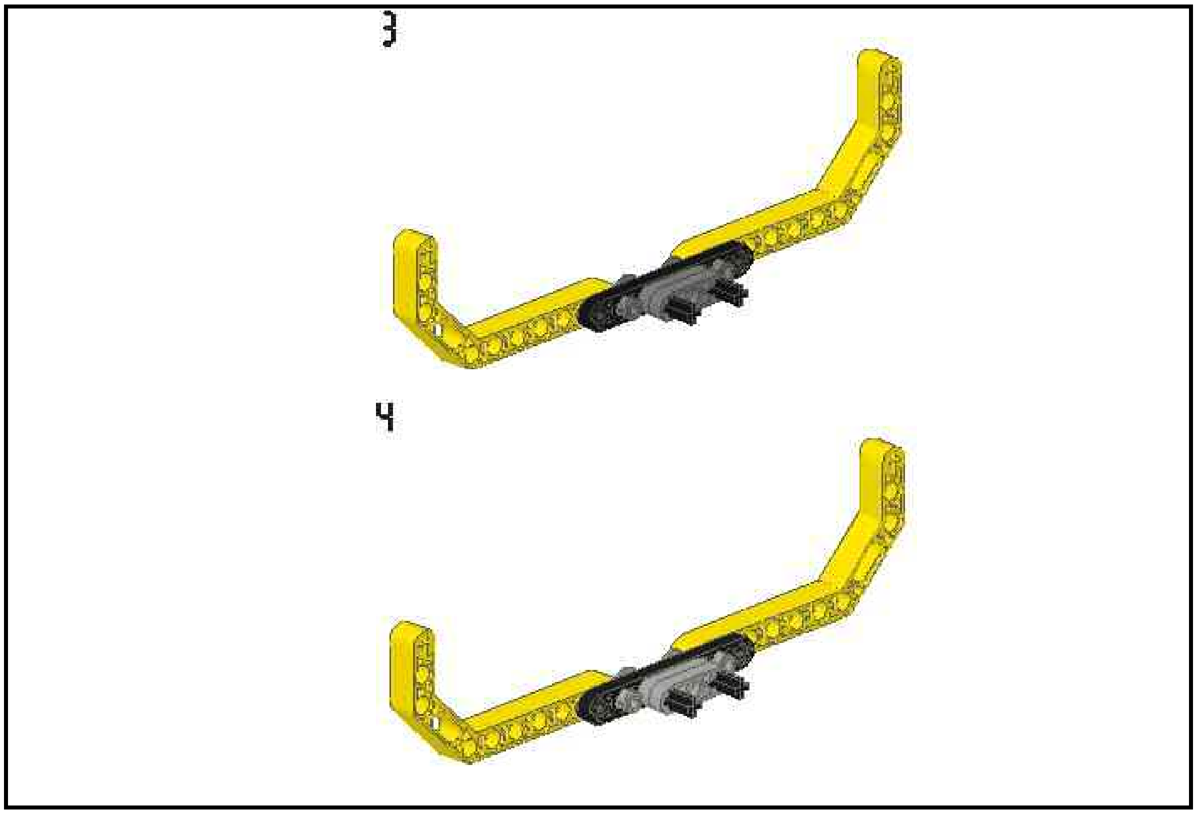


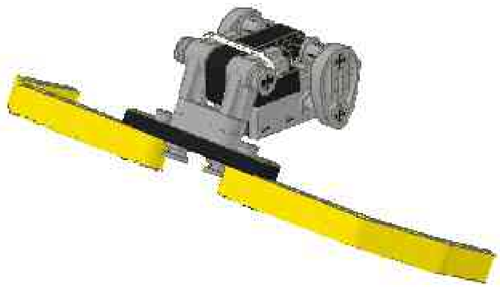


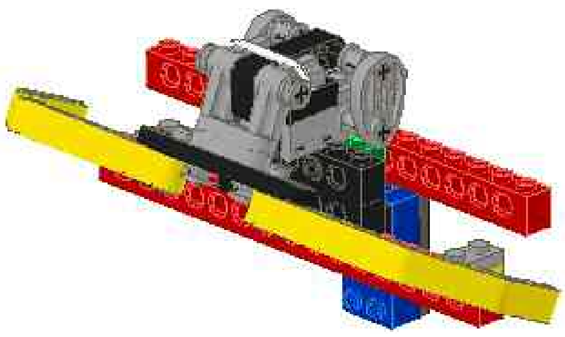
FRONT BUMPER ASSEMBLY

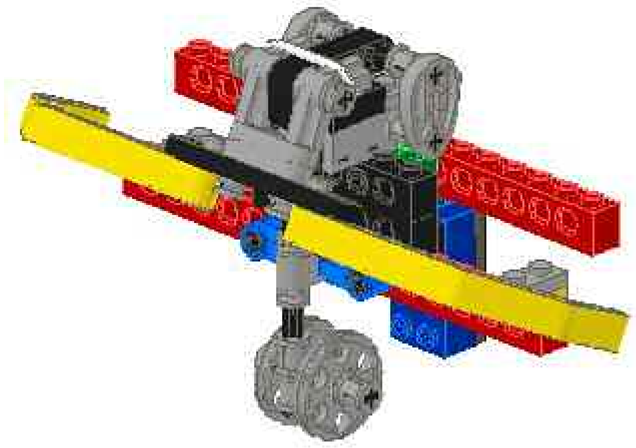




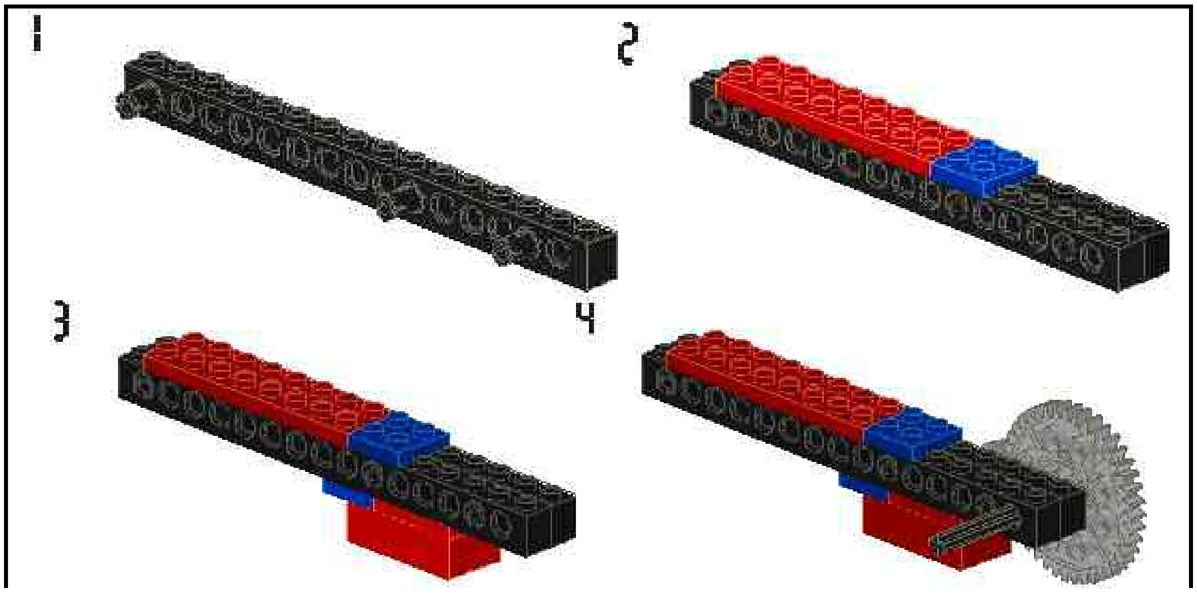

3
4

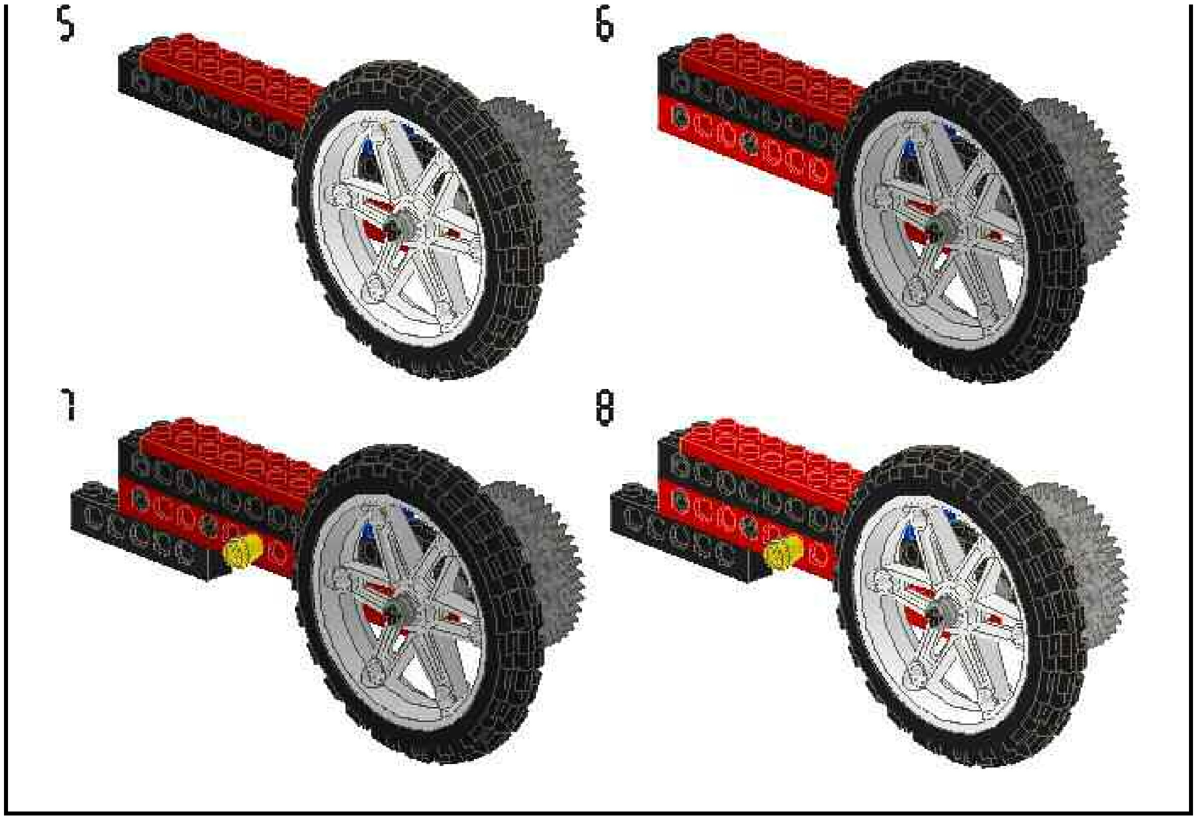




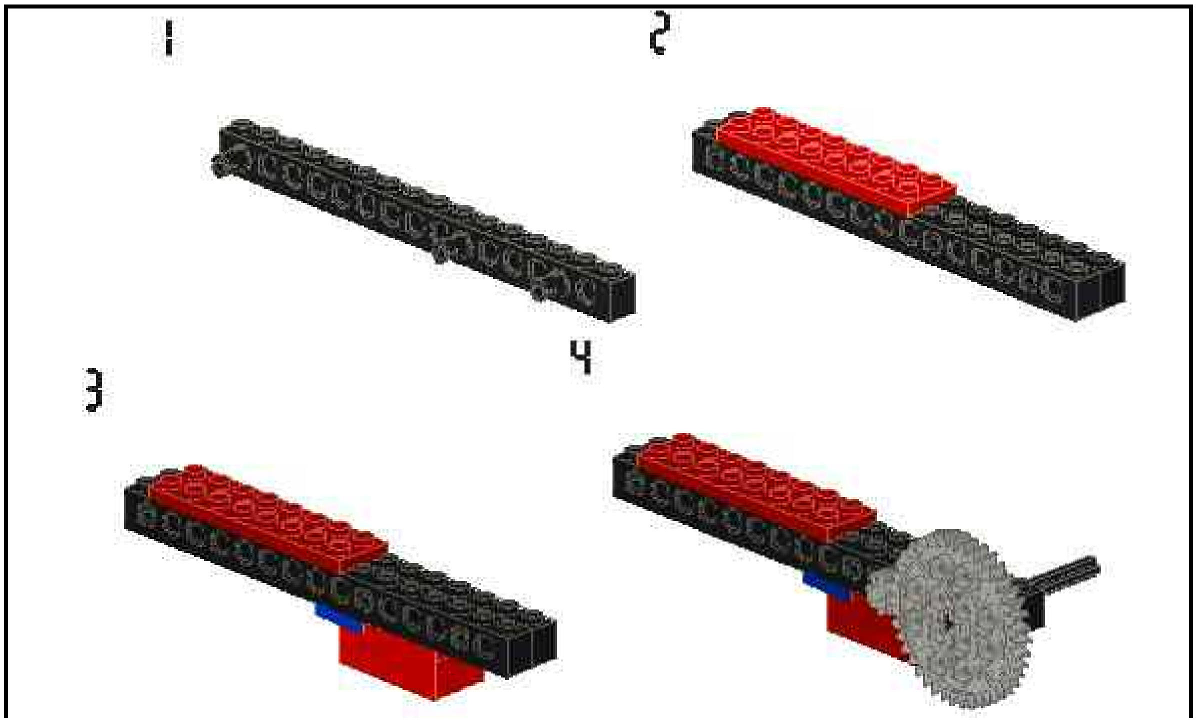


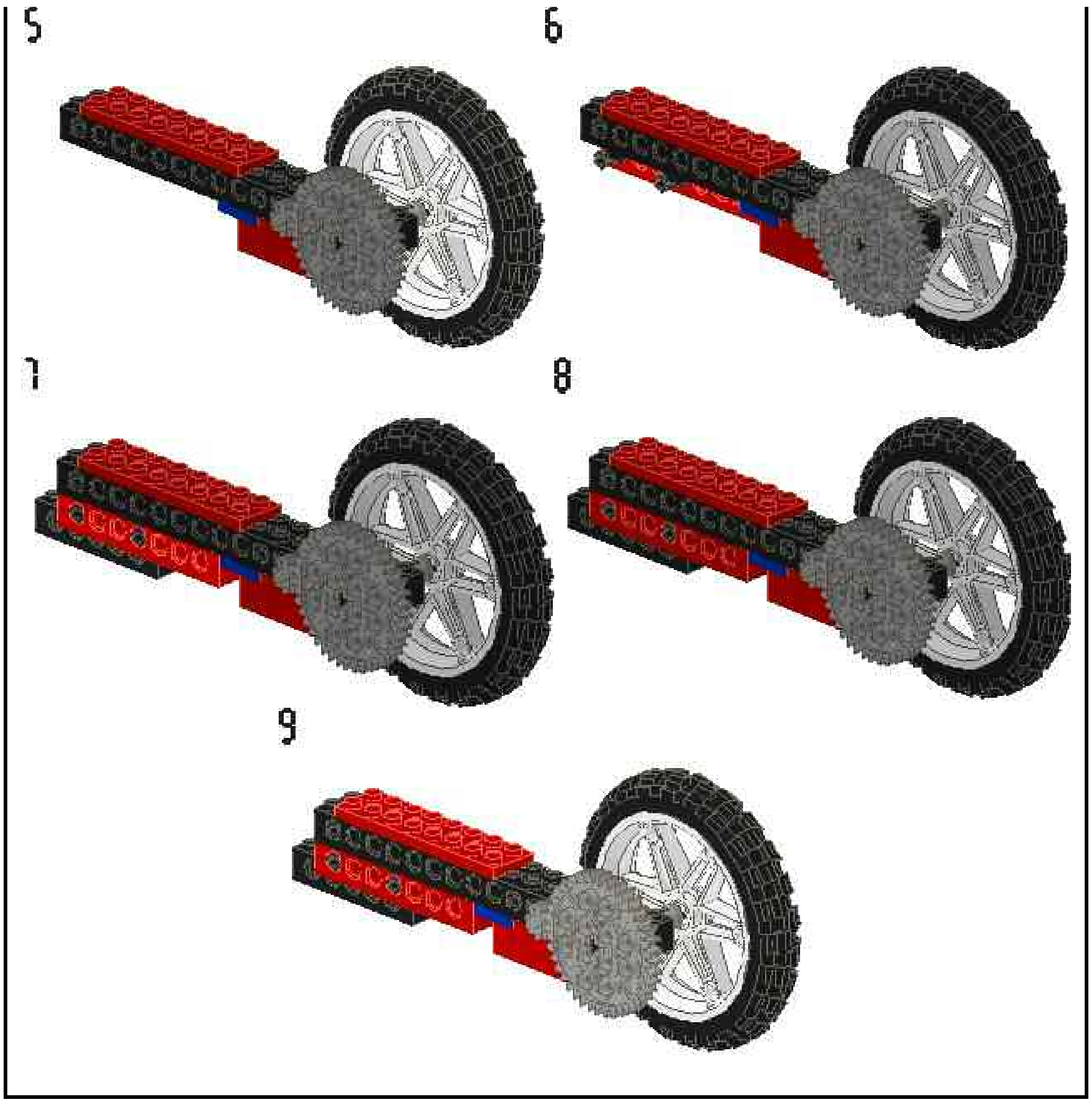
LEFT SIDE ASSEMBLY



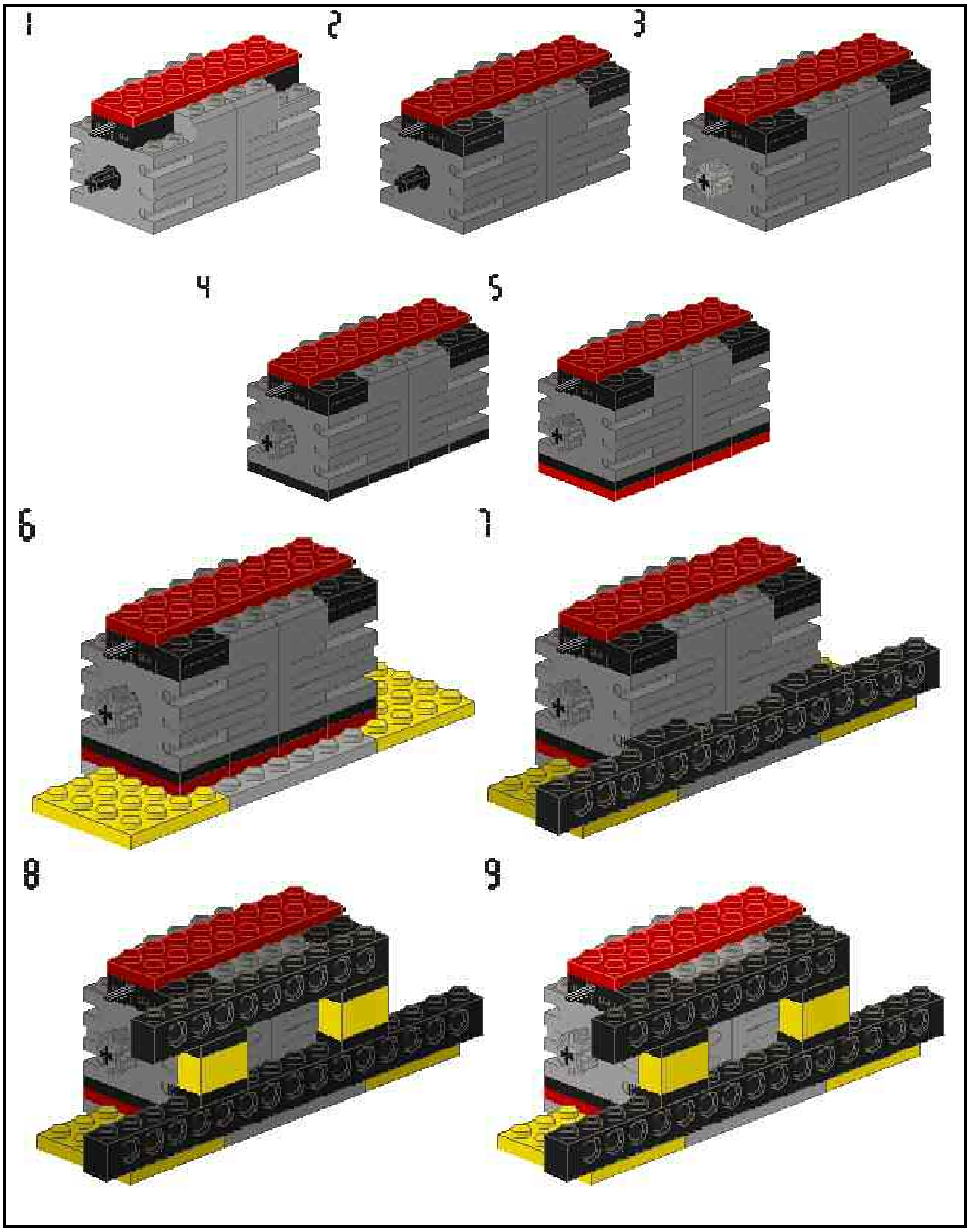


RIGHT SIDE ASSEMBLY

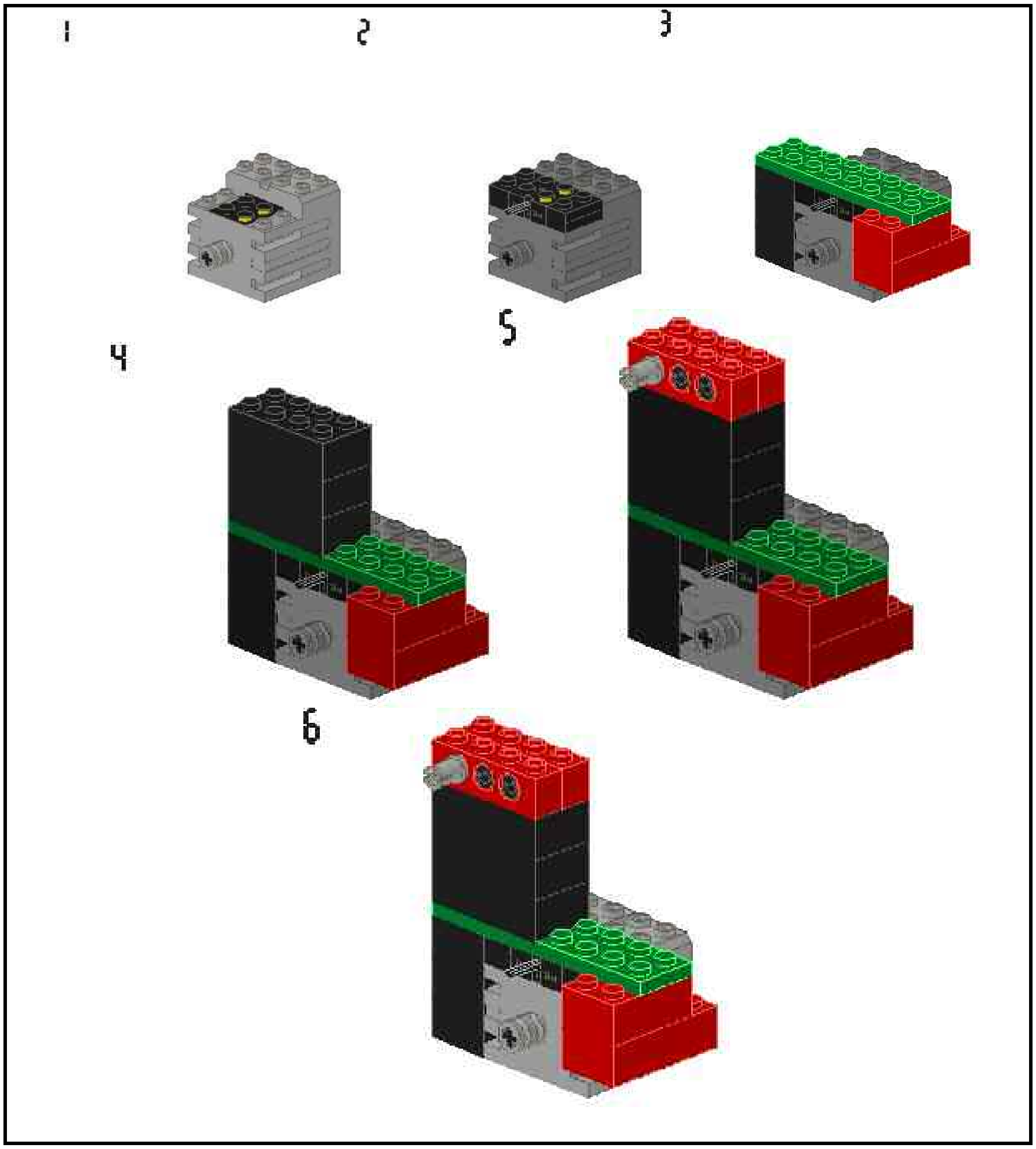



5
6
7
8
9

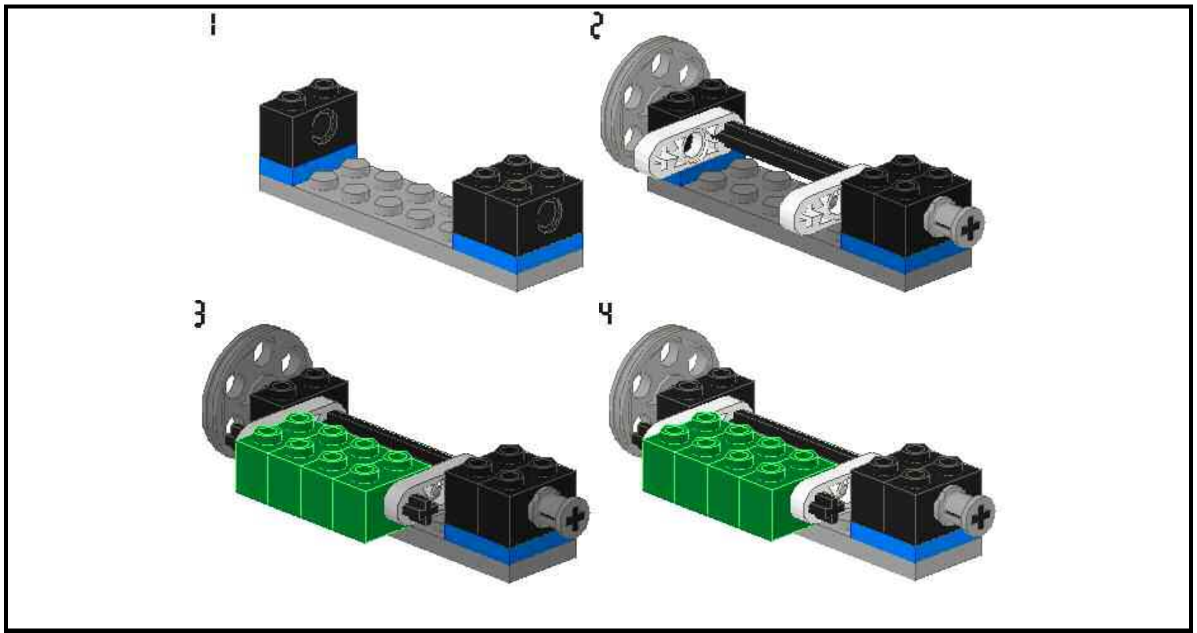
DRIVE MOTOR ASSEMBLY

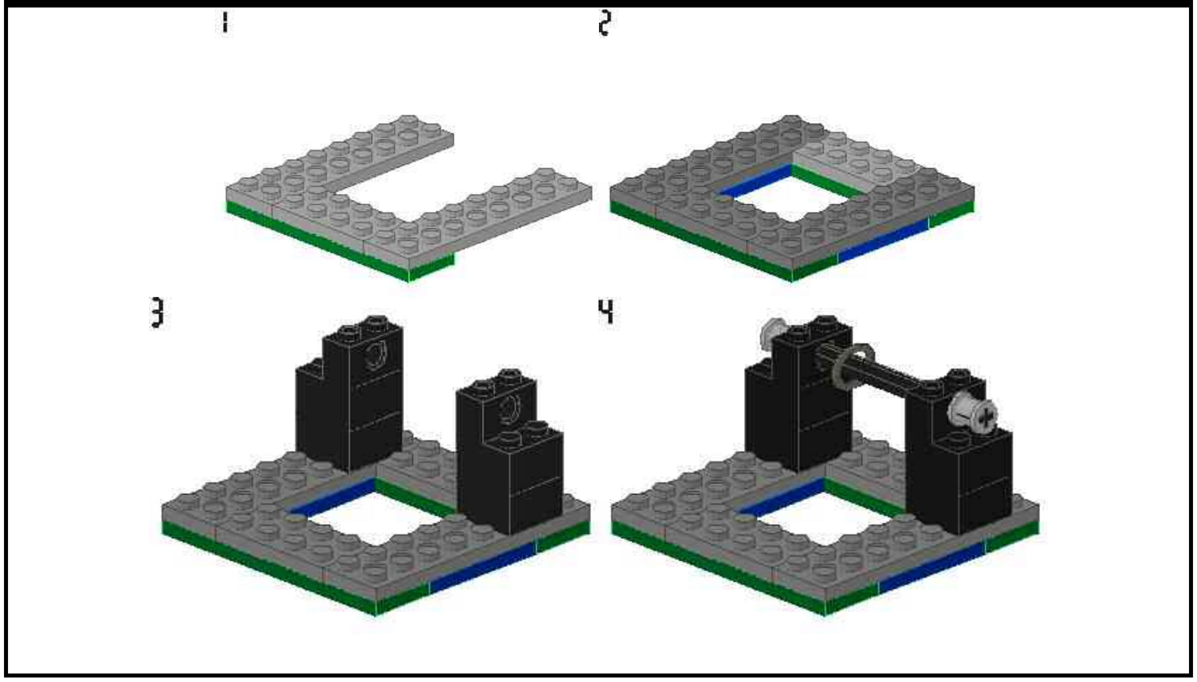


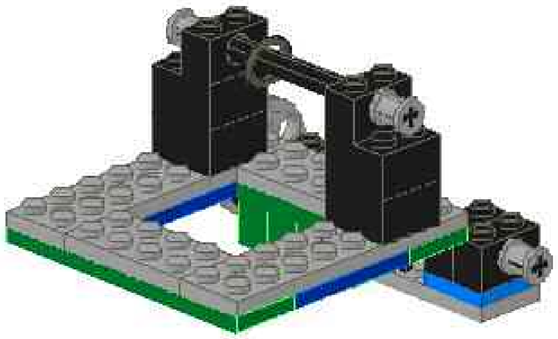
MOTOR 3 ASSEMBLY



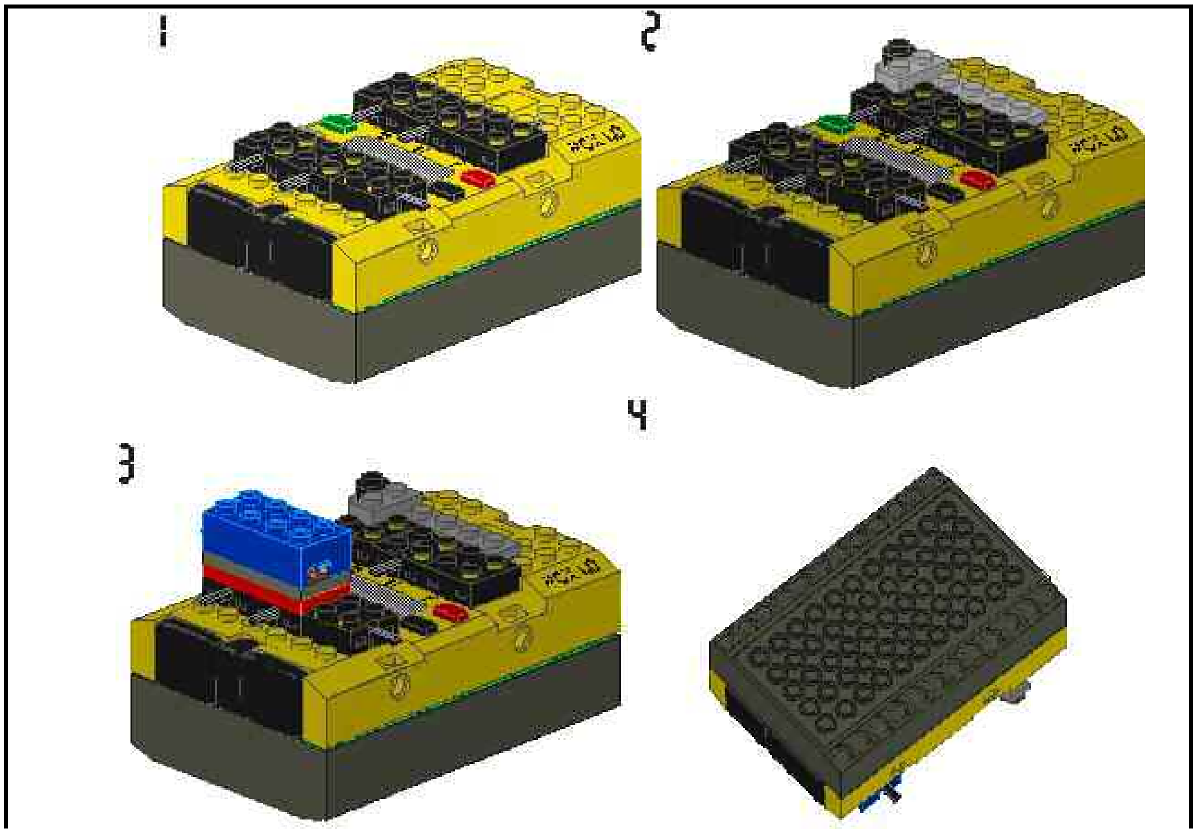
PEN HOLDER/DRIVE SYSTEM ASSEMBLY



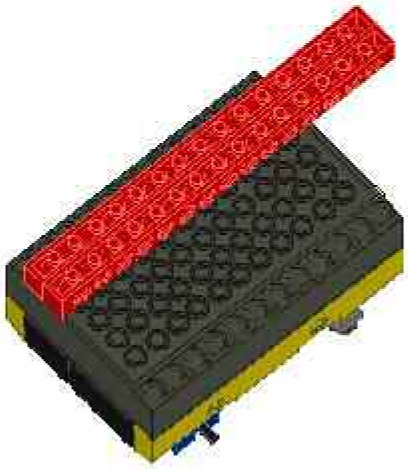




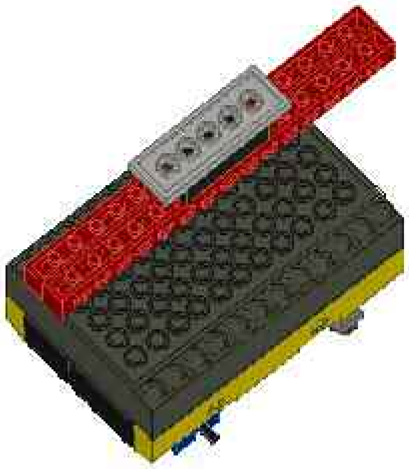
RCX ASSEMBLY



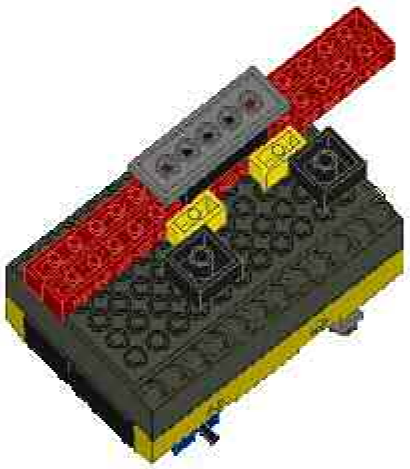
5



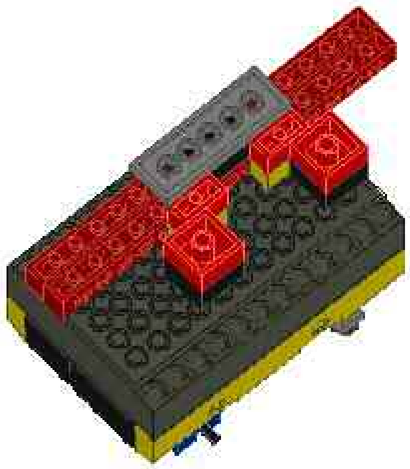
6



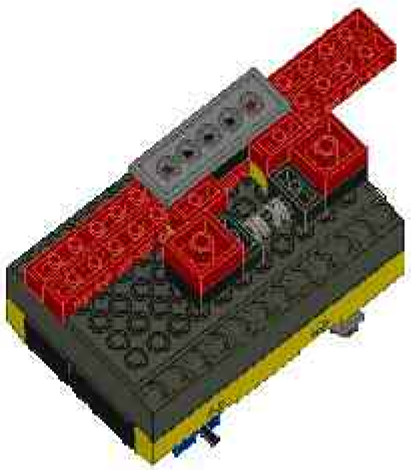
7



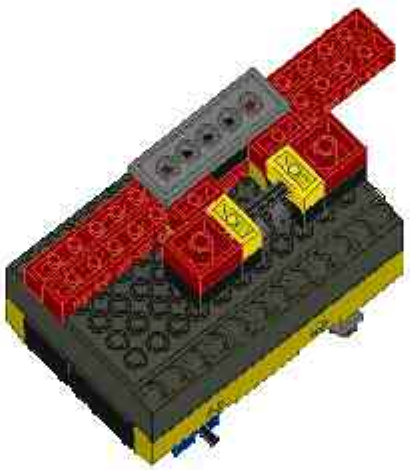
8

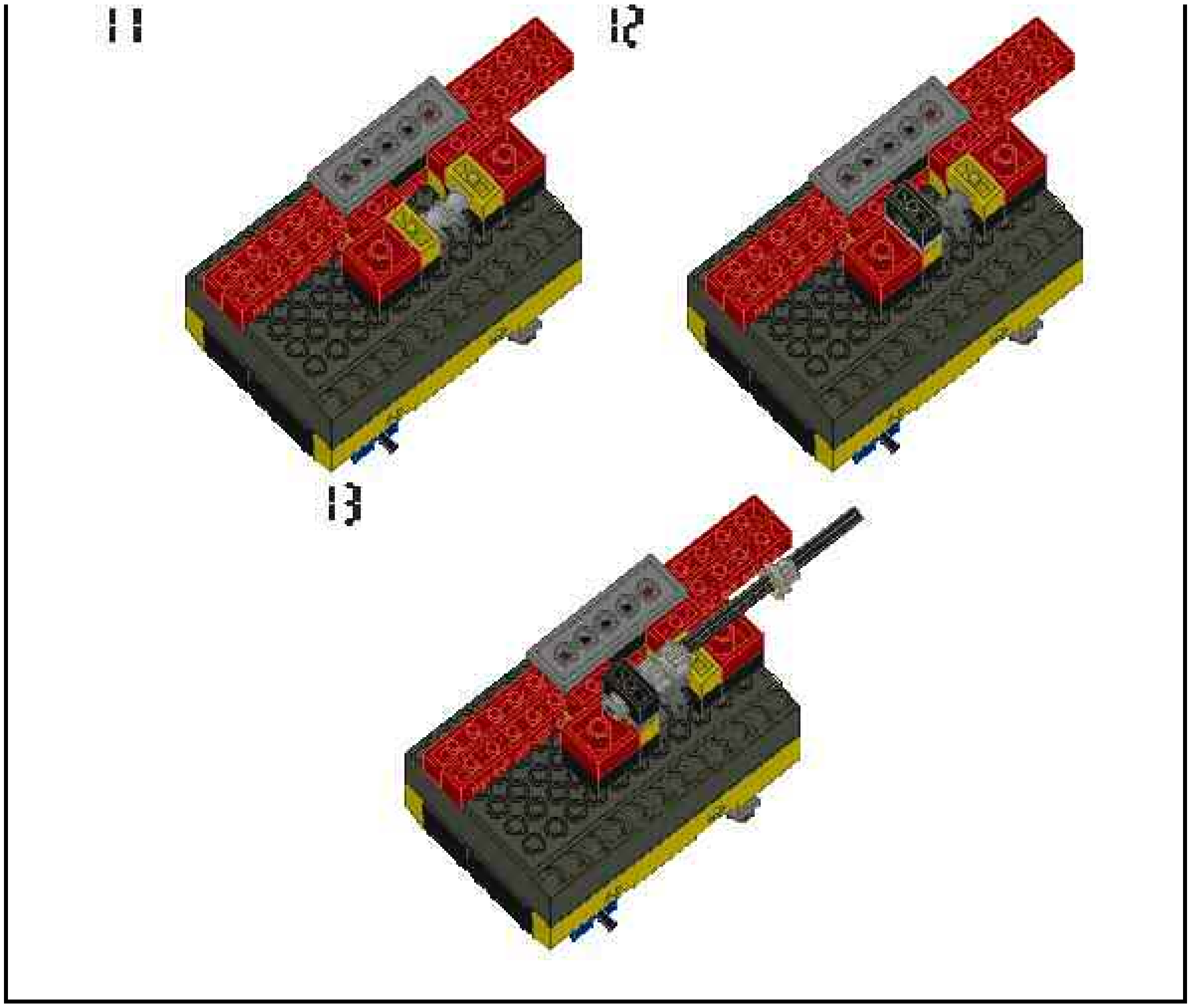


9

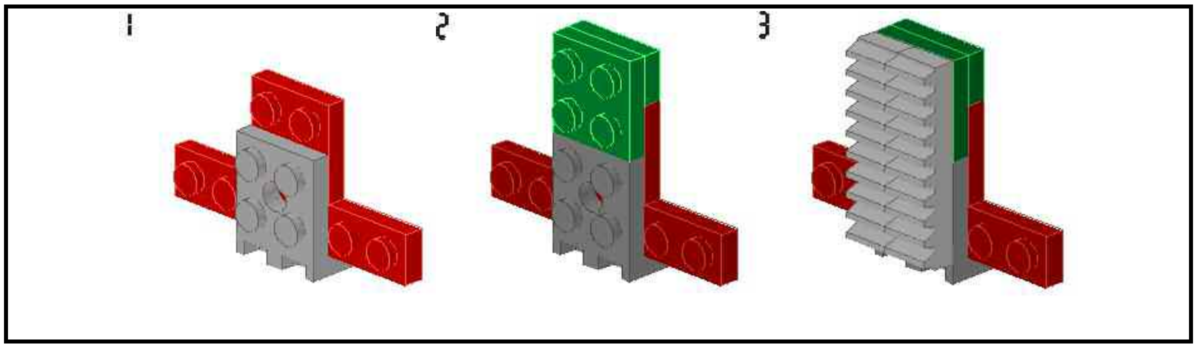


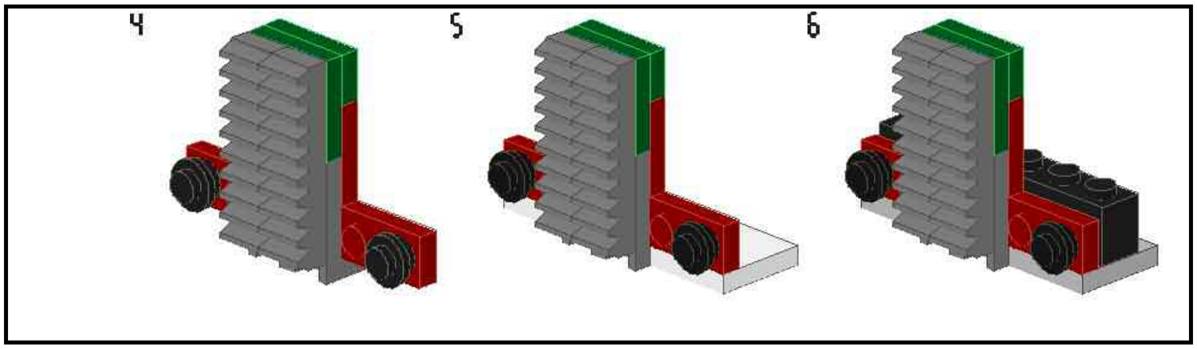
10



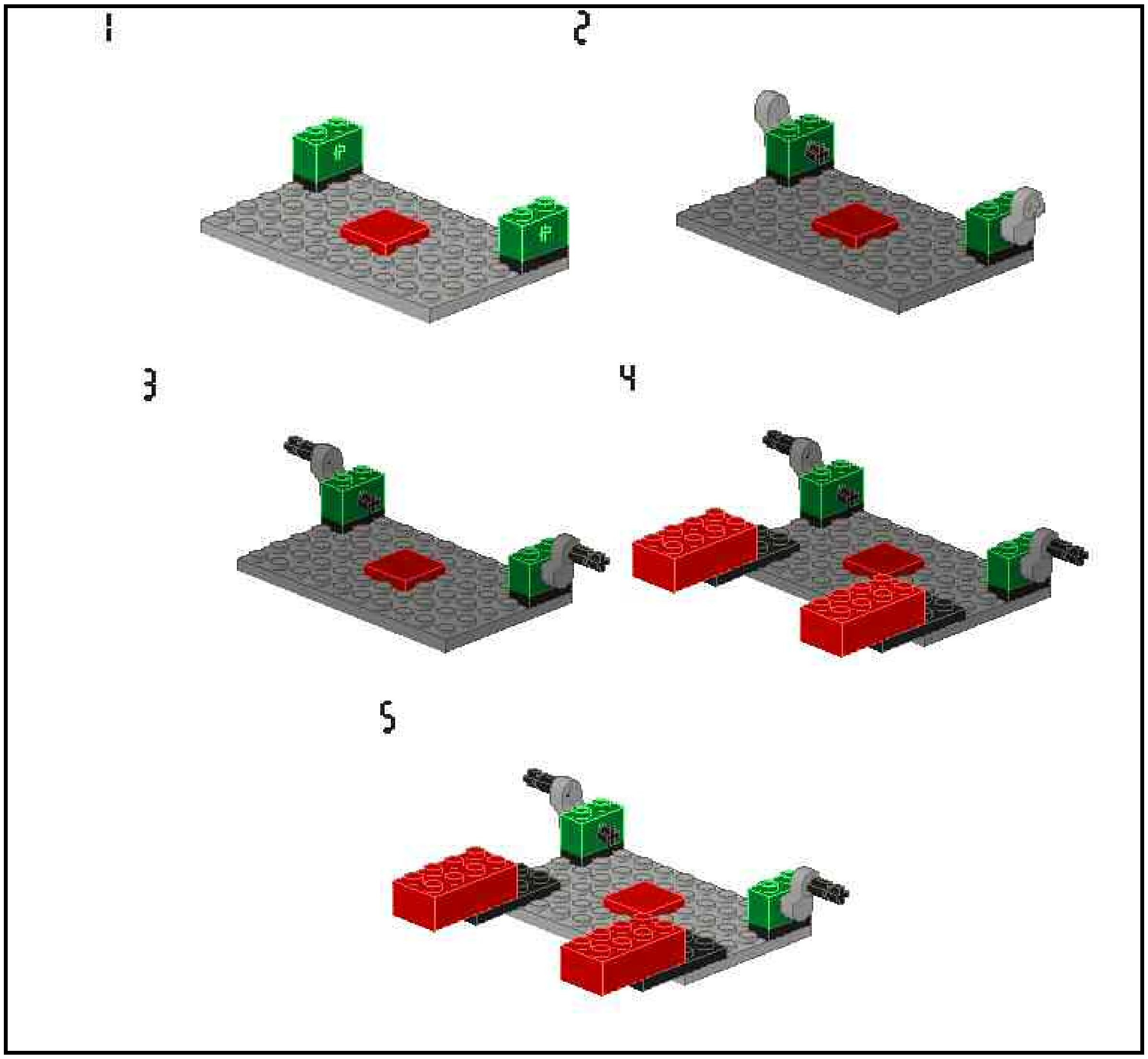


MAGNET APPARATUS ASSEMBLY

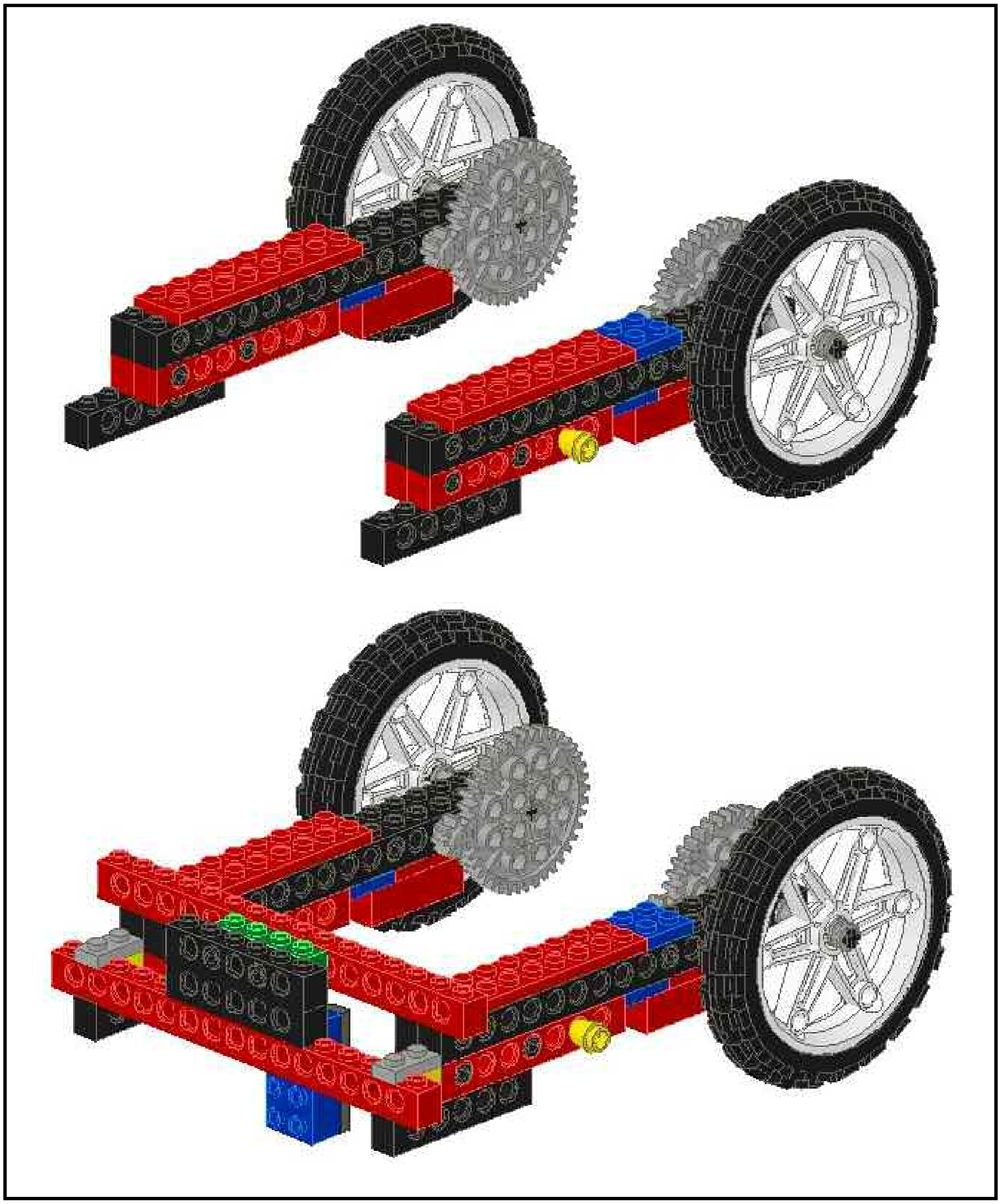


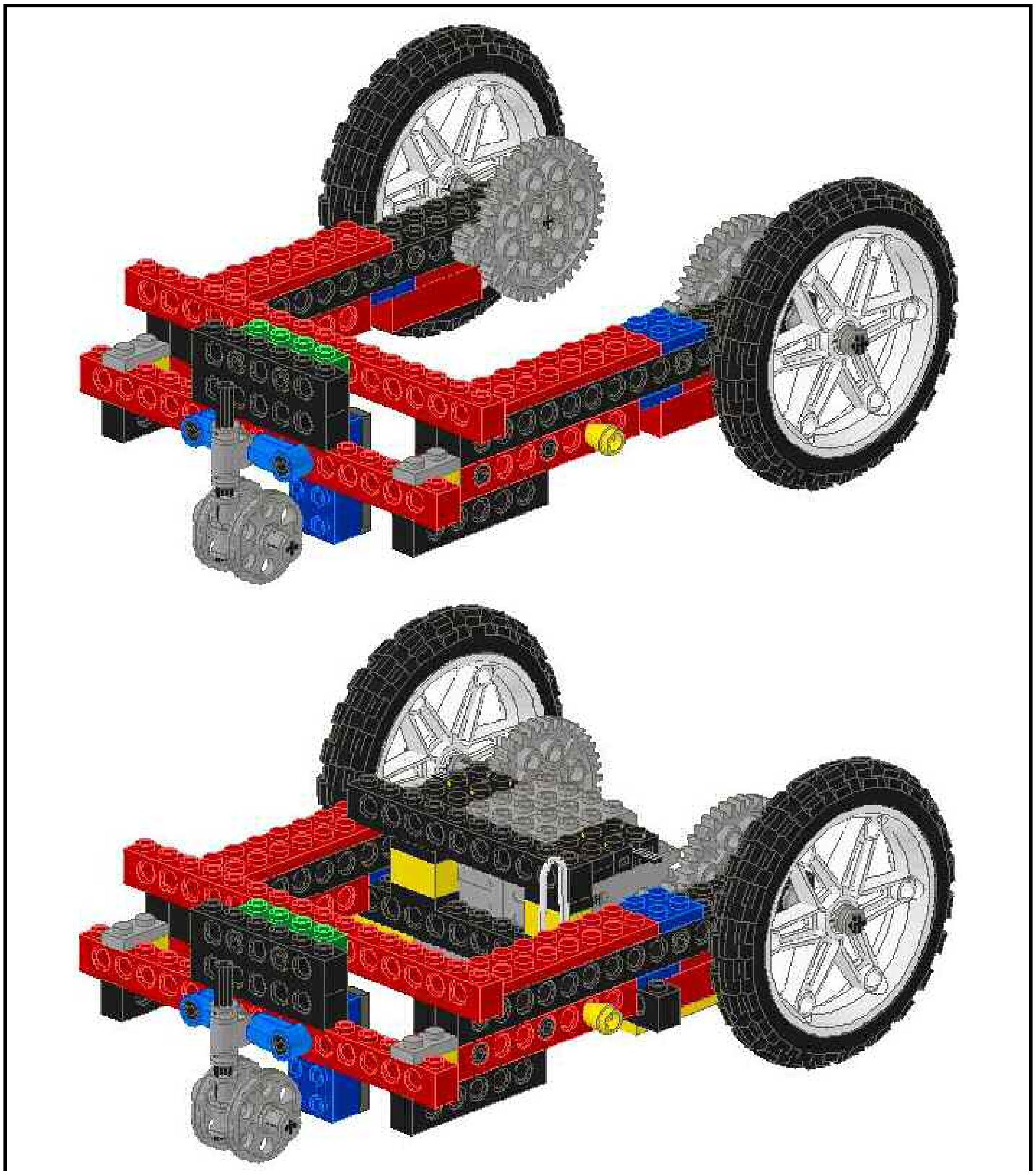


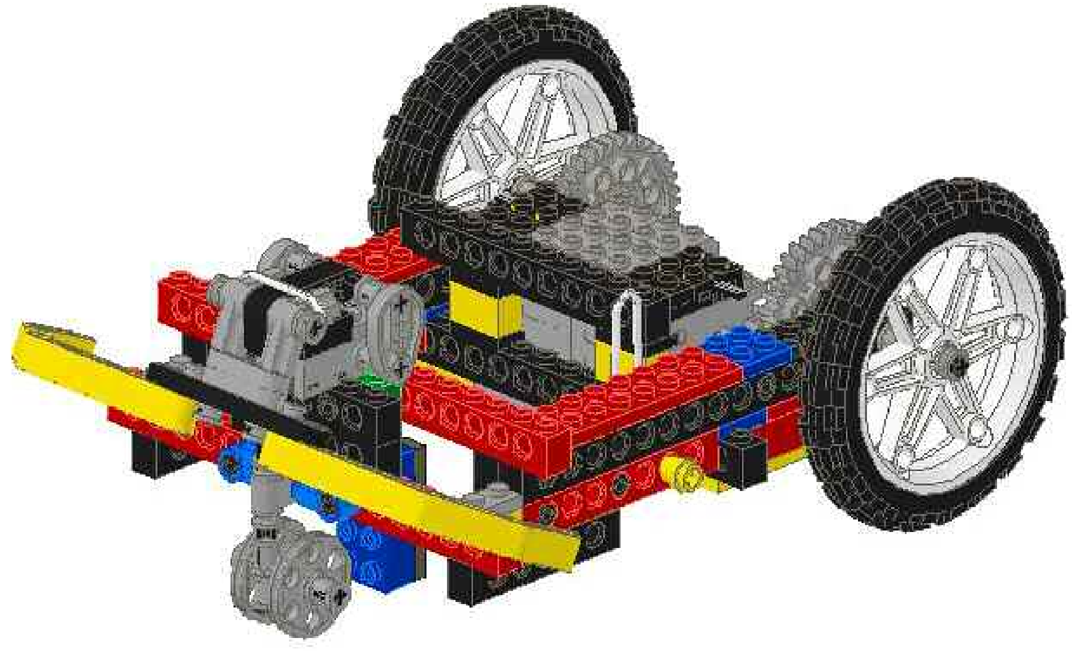
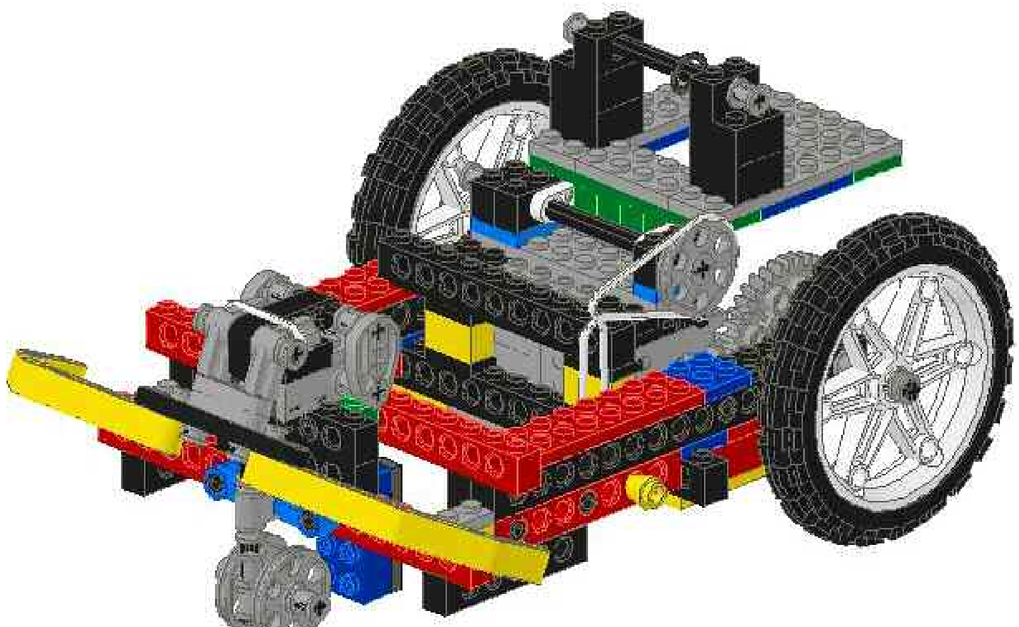
WASHER PLATE ASSEMBLY

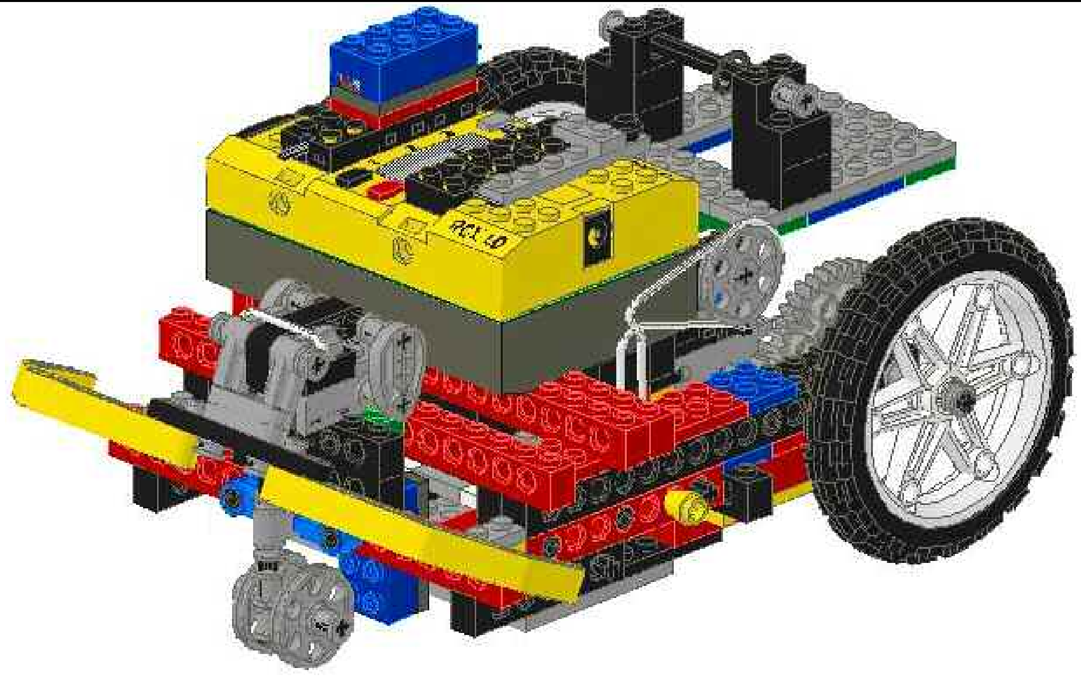


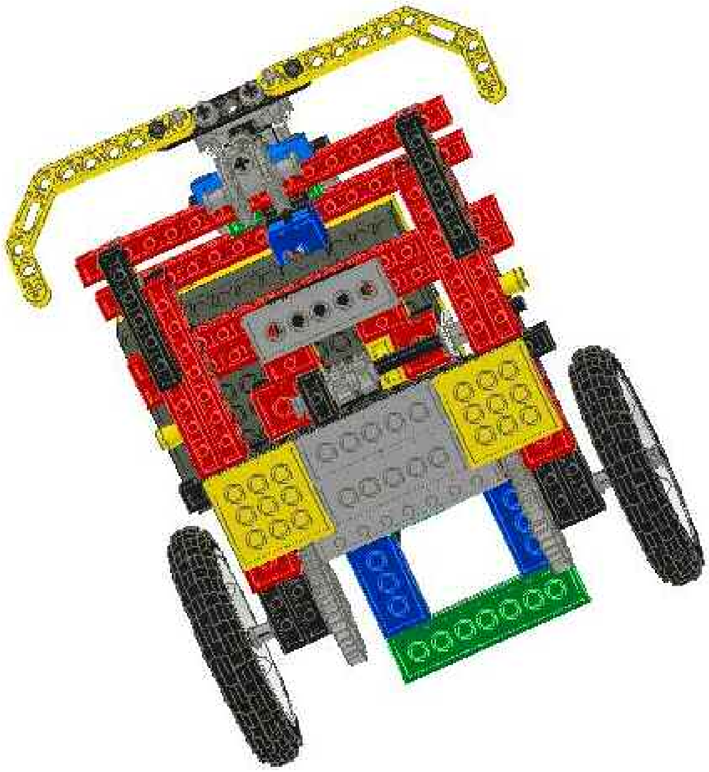
ASSEMBLY OF SUBSYSTEMS

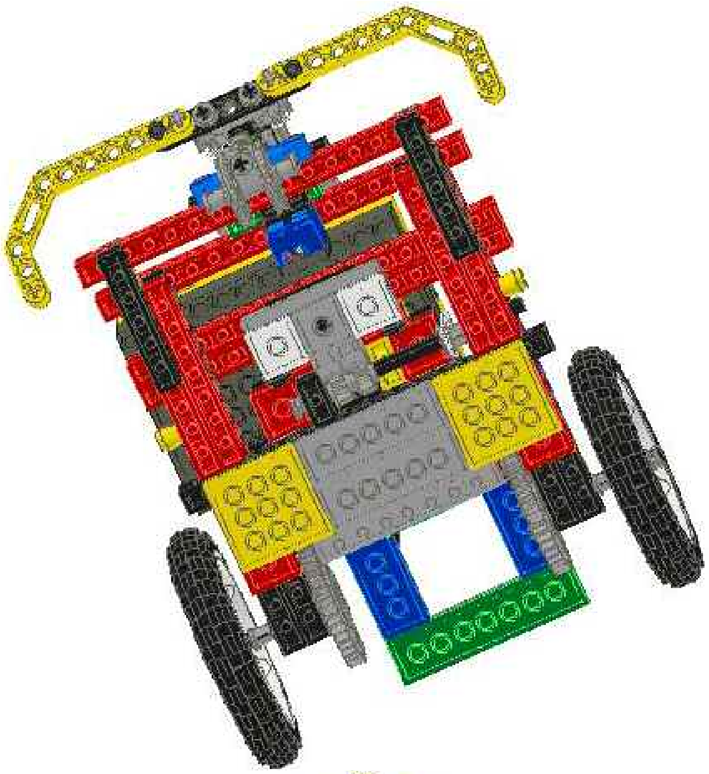


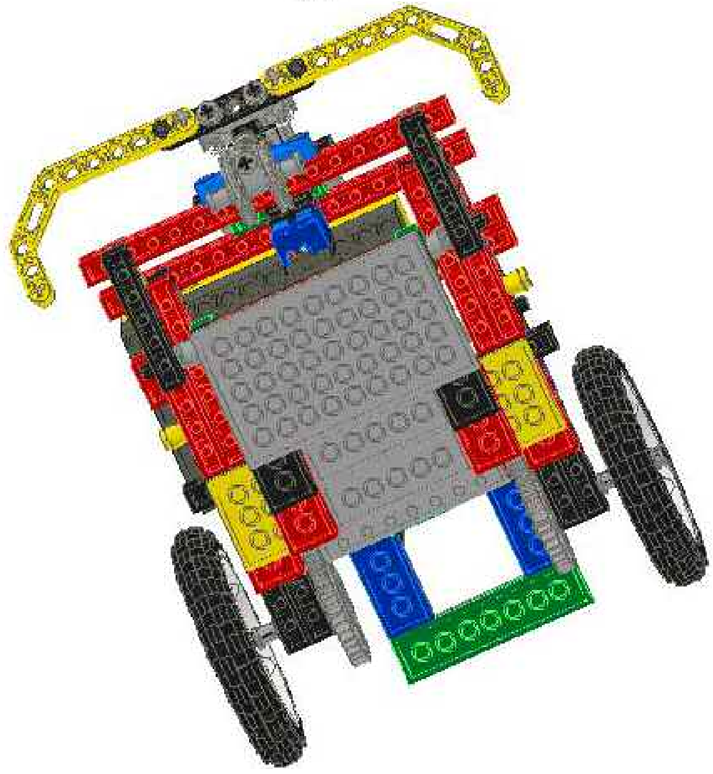


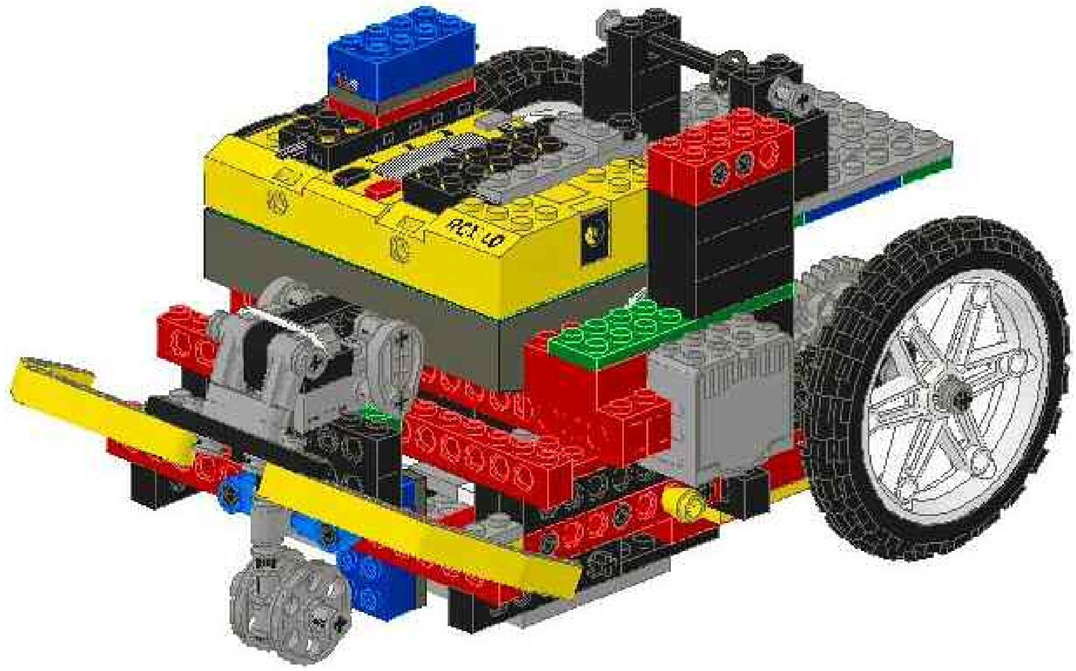



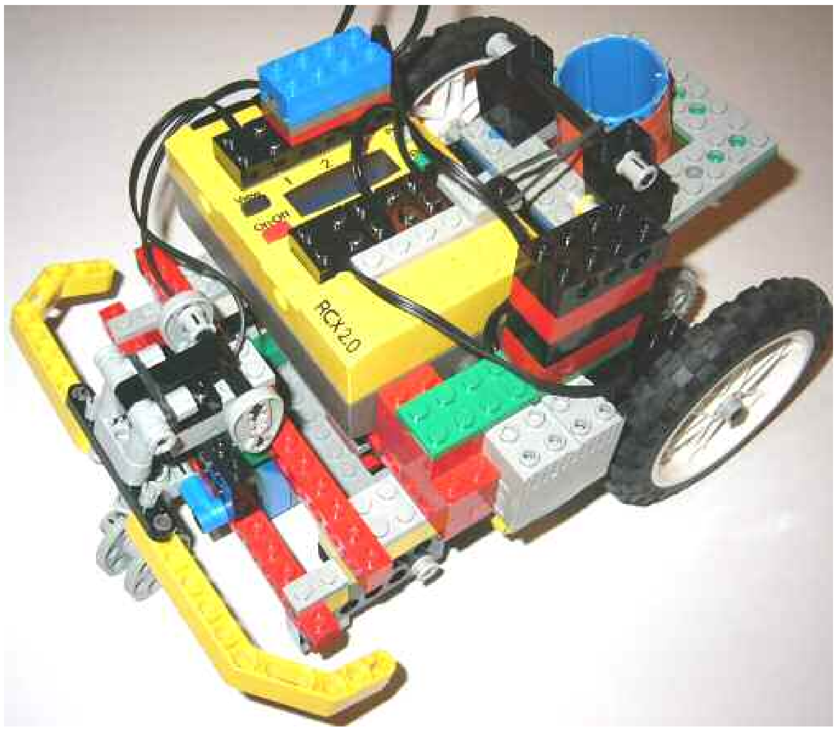









RCX 2.0


RCX 2.0
On-Off
1
2

REFERENCES

[1] Cao, Y., Fukunaga, A., and Kahng, A., "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, pp. 1-23, 1997.

[2] Arai, T., Pagello, E., and Parker, L. E., "Guest Editorial: Advances in multirobot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 655-661, 2002.

[3] Fukada, T., Nakaggawa, S., Kawauchi, Y., and Buss, M., "Structure decision method for self-organizing robots based on cell structure--CEBOT," presented at IEEE International Conference on Robotics and Automation, Los Alamitos, CA, pp. 695-700, 1989.

[4] Beni, G., "The concept of cellular robot," presented at Third IEEE Symposium on Intelligent Control, Arlington, VA, pp. 57-61, 1988.

[5] Beni, G., "Swarm Intelligence," presented at Seventh Annual Meeting of the Robotics Society of Japan, Tokyo, pp. 425-428, 1989.

[6] Hackwood, S. and Beni, G., "Self-organizing sensors by deterministic annealing," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1177-1183, 1992.

[7] Sugawara, K. and Sano, M., "Cooperative acceleration of task performance: Foraging behavior of interacting multi-robot systems," *Physica D*, vol. 100, pp. 343-354, 1997.

[8] Deneubourg, J.-L., Theraulaz, G., and Beckers, R., "Swarm-made architectures," presented at European Conference on Artificial Life, pp. 123-133, 1992.

[9] Brooks, R. A., "Intelligence without reason," presented at International Joint Conference on Artificial Intelligence, pp. 569-595, 1991.

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

[11] Werger, B., "Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams," *Artificial Intelligence*, vol. 110, pp. 293-320, 1999.

[12] Beckers, R., Holland, O. E., and Deneubourg, J.-L., "From local actions to global tasks: Stigmergy and collective robotics," presented at 4th International Workshop on the Synthesis and Simulation of Living Systems: Artificial Life IV, vol. 4, pp. 181-189, 1994.

[13] Cassinis, R., Bianco, G., Cavagnini, A., and Ransenigo, P., "Strategies for navigation of robot swarms to be used in landmines detection," presented at Third European Workshop on Advanced Mobile Robots, pp. 211-218, 1999.

[14] Gaussier, P. and Zrehen, S., "Avoiding the world model trap: An acting robot does not need to be so smart!," *Robotics and Computer-Integrated Manufacturing*, vol. 11, pp. 279-286, 1994.

[15] Genovese, V., Odetti, L., Magni, R., and Dario, P., "Self organizing behavior and swarm intelligence in a cellular robotic system," presented at IEEE International Symposium on Intelligent Control, pp. 243-248, 1992.

[16] Goss, S. and Deneubourg, J.-L., "Harvesting by a group of robots," presented at First European Conference on Artificial Life: Toward a Practice of Autonomous Systems, pp. 195-204, 1992.

[17] Mataric, M. J., "Minimizing complexity in controlling a mobile robot population," presented at IEEE International Conference on Robotics and Automation, Nice, France, pp. 830-835, 1992.

[18] Brooks, R. A., "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139-159, 1991.

[19] Howe, J., "A flexible and innovative platform for autonomous mobile robots," MIT, Masters Thesis Proposal January 2003.

[20] Arai, T., Ogata, H., and Suzuki, T., "Collision avoidance among multiple robots using virtual impedance," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 479-485, 1989.

[21] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D., "A taxonomy for swarm robots," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, pp. 441-447, 1993.

[22] Ichikawa, S., Hara, F., and Hosokai, H., "Cooperative route-searching behavior of multi-robot system using hello-call communication," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, pp. 1149-1156, 1993.

[23] Singh, S. P. N., Thayer, S. M., and Thayer, W. P., "A foundation for kilorobotic exploration," presented at 2002 Congress on Evolutionary Computation, vol. 2, pp. 1033-1038, 2002.

[24] Russell, A., "Mobile robot guidance using a short-lived heat trail," *Robotica*, vol. 11, pp. 427-431, 1993.

[25] Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chretien, L., "The dynamics of collective sorting: robot-like ants and ant-like robots," presented at First European Conference on Artificial Life: Toward a Practice of Autonomous Systems, pp. 123-133, 1992.

[26] Melhuish, C., Holland, O. E., and Hoddell, S., "Collective sorting and segregation in robots with minimal sensing," presented at Fifth International Conference on Simulation of Adaptive Behavior: From Animals to Animats V, Zurich, Switzerland, 1998.

[27] Mataric, M. J., "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 323-336, 1997.

[28] Beni, G. and Wang, J., "Theoretical problems for the realization of distributed robotic systems," presented at IEEE International Conference on Robotics and Automation, Sacramento, CA, pp. 1914-1920, 1991.

[29] Ferrari, M., Ferrari, G., and Hempel, R., *Building Robots with LEGO MINDSTORMS*. Rockland, MA: Syngress, 2002.

[30] Kotay, Keith, "Robo-Rats Locomotion Page," 2001, http://www.cs.dartmouth.edu/~robotlab/robotlab/courses/cs54-2001s/locomotion.html

[31] Borenstein, J., Everett, H. R., and Feng, L., "Where am I? Sensors and Methods for Mobile Robot Positioning," University of Michigan 1996.

[32] ActivMedia Robotics, "Pioneer 3 General Purpose Robot," 2003, http://www.activrobots.com/ROBOTS/p2dx.html#descrip

[33] ActivMedia Robotics, "Pioneer 3 All-Terrain Robot," 2003, http://www.activrobots.com/ROBOTS/p2at.html#descrip

[34] Nelson, W. L., "Continuous steering-function control of robot carts," *IEEE Transactions on Industrial Electronics*, vol. 36, pp. 330-337, 1989.

[35] Mason, M., Pai, D., Rus, D., Howell, J., and Taylor, L. R., "Experiments with desktop mobile manipulators," *International Symposium on Experimental Robotics*, 1999.

[36] iRobot Corporation, "B21r Mobile Robot," 2003, http://www.irobot.com/rwi/p06.asp

[37] R.A., J., "Autonomous Navigation of a Martian Rover in Very Rough Terrain," presented at International Symposium on Experimental Robotics, Sydney, pp. 225-234, 1999.

[38] Yamaguchi, H., "A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations," presented at IEEE Conference on Decision and Control, Las Vegas, Nevada, vol. 41, pp. 2984-2991, 2002.

[39] Zhang, B., Chen, X., Liu, G., and Cai, Q., "Agent architecture: a survey on RoboCup-99 simulator teams," presented at Third World Congress on Intelligent Control and Automation, pp. 194-198, 2000.

[40] Yamaguchi, H., "A cooperative hunting behavior by mobile-robot troops," *International Journal of Robotics Research*, vol. 18, pp. 931-940, 1999.

[41] Werger, B. and Mataric, M. J., "Robotic "food" chains: Externalization of state and program for minimal-agent foraging," presented at Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats, vol. 4, pp. 626-633, 1996.

[42] Wang, Z. D., Nakano, E., and Matsukawa, T., "Cooperating multiple behavior-based robots for object manipulation," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1524-1531, 1994.

[43] Sugawara, K. and Watanabe, T., "Swarming robots - Foraging behavior of simple multi-robot system," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, pp. 2702-2707, 2002.

[44] Hayes, A. T. and Dormiani-Tabatabaei, P., "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots," presented at IEEE International Conference on Robotics and Automation, Washington, D.C., pp. 3900-3905, 2002.

[45] Fredslund, J. and Mataric, M. J., "Robot formations using only local sensing and control," presented at IEEE International Symposium on Computational Intelligence in Robotics and Automation, Alberta, Canada, pp. 308-313, 2001.

[46] Balch, T. and Arkin, R. C., "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926-939, 1998.

[47] Winfield, A. F. T. and Holland, O. E., "The application of wireless local area network technology to the control of mobile robots," *Microprocessors and Microsystems*, vol. 23, pp. 597-607, 2000.

[48] Krotkov, E. and Blitch, J., "The Defense Advanced Research Projects Agency (DARPA) Tactical Mobile Robotics Program," *International Journal of Robotics Research*, vol. 18, pp. 769-776, 1999.

[49] Yamaguchi, H., "Adaptive formation control for distributed autonomous mobile-robot groups," presented at IEEE International Conference on Robotics and Automation, Washington, D.C., pp. 2300-2305, 1997.

[50] Sudd, J. H., "The transport of prey by ants," *Behaviour*, vol. 25, pp. 234-271, 1965.

[51] Kube, R. C. and Zhang, H., "Stagnation recovery behaviours for collective robotics," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1893-1890, 1994.

[52] Bay, J. S., "Design of the "Army-Ant" cooperative lifting robot," in *IEEE Robotics & Automation Magazine*, March 1995, pp. 36-43.

[53] Stilwell, D. J. and Bay, J. S., "Toward the development of a material transport system using swarms of ant-like robots," presented at IEEE International Conference on Robotics and Automation, Atlanta, GA, 1993.

[54] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E., "RoboCup: The Robot World Cup Initiative," presented at First International Conference On Autonomous Agents, Marina del Ray, 1997.

[55] Hugel, V., Bonnin, P., and Blazevic, P., "Reactive and adaptive control architecture designed for the Sony legged robots league in RoboCup," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1032-1037, 2000.

[56] Werger, B., "Principles of minimal control for comprehensive team behaviour," presented at IEEE International Conference on Robotics and Automation, vol. 4, pp. 3504-3509, 1998.

[57] Pagello, E., D'Angelo, A., Montesello, F., Garelli, F., and Ferrari, C., "Cooperative behaviors in multi-robot systems through implicit communication," *Robotics and Autonomous Systems*, vol. 29, pp. 65-77, 1999.

[58] Drogoul, A. and Ferber, J., "From Tom Thumb to the Dockers: Some experiments with foraging robots," presented at Second International Conference on Simulation of Adaptive Behavior, Honolulu, HI, pp. 451-459, 1992.

[59] Stephens, D. W. and Krebs, J. R., *Foraging Theory*. Princeton, NJ: Princeton University Press, 1986.

[60] Altenburg, K., "Adaptive resource allocation for a multiple mobile robot system using communication," North Dakota State University, Technical Report NDSU-CSOR-TR-9404, 1994.

[61] Mataric, M. J., "Designing and understanding adaptive group behavior," *Adaptive Behavior*, vol. 4, pp. 51-80, 1995.

[62] Holldobler, B. and Wilson, E. O., *The Ants*. Cambridge, Mass.: Harvard University Press, 1990.

[63] Shen, W.-M., Chuong, C.-M., and Will, P., "Simulating self-organization for multi-robot systems," presented at 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, pp. 2776-2781, 2002.

[64] Prescott, T. J. and Ibbotson, C., "A Robot Trace Maker: Modeling the Fossil Evidence of Early Invertebrate Behavior," *Artificial Life*, vol. 3, pp. 289-306, 1997.

[65] Deveza, R., Thiel, D., Russell, A., and Mackay-Sim, A., "Odor sensing for robot guidance," *International Journal of Robotics Research*, vol. 13, pp. 232-239, 1994.

[66] Russell, A., "Laying and sensing odor markings as a strategy for assisting mobile robot navigation tasks," in *IEEE Robotics & Automation Magazine,* 1995, pp. 3-9.

[67] Grasse, P. P., "La reconstruction du nid et les coordinations inter-individuelle chez bellicoitermes natalenis et cubitermes sp. la theorie de la stigmergie: Essai d'interpretation des termites constructeurs," *Insectes Sociaux*, vol. 6, 1959.

[68] Bonabeau, E., Dorigo, M., and Theraulaz, G., *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press, 1999.

[69] Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Aron, S., and Camazine, S., "Self-organization in social insects," *Trends in Ecology and Evolution*, vol. 12, pp. 188-193, 1997.

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

[71] Deneubourg, J.-L., Pasteels, J. M., and Verhaeghe, J. C., "Probabilistic behaviour in ants: A strategy of errors?," *Journal of Theoretical Biology*, vol. 105, pp. 259-271, 1983.

[72] Pasteels, J. M. and Deneubourg, J.-L., "From Individual to Collective Behavior in Social Insects," in *Experientia Supplementum*, vol. 54. Boston: Birkhauser Verlag, 1987, pp. 433.

[73] Martinoli, A., Ijspeert, A. J., and Mondada, F., "Understanding collective aggregation mechanisms: From probablistic modelling to experiments with real robots," *Robotics and Autonomous Systems*, vol. 29, pp. 51-63, 1999.

[74] Pamecha, A., Chiang, C. J., Stein, D., and Chirikjian, G. S., "Design and implementation of metamorphic robots," presented at ASME Design Engineering Technical Conference and Computers and Engineering Conference, New York, NY, pp. 1-10, 1996.

[75] Chirikjian, G. S., "Kinematics of a metamorphic robotic system," presented at IEEE International Conference on Robotics and Automation, Los Alamitos, CA, pp. 449-455, 1994.

[76] Hayes, A. T., Martinoli, A., and Goodman, R. M., "Swarm robotic odor localization," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, Hawaii, pp. 1073-1078, 2001.

[77] Genovese, V., Dario, P., Magni, R., and Odetti, L., "Self organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of pollutants," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, NC, pp. 1575-1582, 1992.

[78] Rybski, P. E., Papanikolopoulos, N. P., Stoeter, S. A., Krantz, D. G., and Yesin, K. B., "Enlisting Rangers and Scouts for reconnaissance and surveillance," in *IEEE Robotics & Automation Magazine*, December 2000, pp. 14-23.

[79] Camazine, S., "Self-organizing pattern formation on the combs of honey bee colonies," *Behavioral Ecology and Sociobiology*, vol. 28, pp. 61-76, 1991.

[80] Camazine, S. and Sneyd, J., "A model of collective nectar source selection by honey bees: self-organization through simple rules," *Journal of Theoretical Biology*, vol. 149, pp. 547-571, 1991.

[81] Edelstein-Keshet, L., Watmough, J., and Ermentrout, G., "Trail following in ants: individual properties determine population behavior," *Behavioral Ecology and Sociobiology*, vol. 36, pp. 119-133, 1995.

[82] Beckers, R., Deneubourg, J.-L., and Goss, S., "Trail laying behaviour during food recruitment in the ant *Lasius niger* (L.)," *Insectes Sociaux*, vol. 39, pp. 59-72, 1992.

[83] Pasteels, J. M., Deneubourg, J.-L., and Goss, S., "Self-organization mechanisms in ant societies (I): Trail recruitment to newly discovered food sources," *Experientia Supplementum*, vol. 54, pp. 155-175, 1987.

[84] Resnick, M., *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. London: MIT Press, 1994.

[85] Nicolis, S. C. and Deneubourg, J.-L., "Emerging patterns and food recruitment in ants: an analytical study," *Journal of Theoretical Biology*, vol. 198, pp. 575-592, 1999.

[86] Bonabeau, E. and Cogne, F., "Oscillation-enhanced adaptability in the vicinity of a bifurcation: the example of foraging in ants," presented at Fourth International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 537-542, 1996.

[87] Jun, J.-H., Lee, D.-W., and Sim, K.-B., "Realization of cooperative strategies and swarm behavior in distributed autonomous robotic systems using artificial immune system," presented at IEEE International Conference on Systems, Man, and Cybernetics, vol. 6, pp. 614-619, 1999.

[88] Taheri, S. A. and Calva, G., "Imitating the human immune system capabilities for multi-agent federation formation," presented at 2001 IEEE International Symposium on Intelligent Control, Mexico City, Mexico, pp. 25-30, 2001.

[89] Mitsumoto, N., Fukada, T., Shimojima, K., and Ogawa, A., "Micro autonomous robotic system and biologically inspired immune system swarm strategy as a multi agent robotic system," presented at IEEE International Conference on Robotics and Automation, pp. 2187-2192, 1995.

[90] Goldbeter, A., *Biochemical Oscillations and Cellular Rhythms*. Cambridge: Cambridge University Press, 1996.

[91] Pratt, S. C., "Recruiment and other communication behavior in the Ponerine ant *Ectatomma ruidum*," *Ethology*, vol. 81, pp. 313-331, 1989.

[92] Bernstein, R. A., "Foraging strategies of ants in response to variable food density," *Ecology*, vol. 56, pp. 213-219, 1975.

[93] Hahn, M. and Maschwitz, U., "Foraging strategies and recruitment behaviour in the European harvester ant *Messor rufitarsis* (F.)," *Oecologia*, vol. 68, pp. 45-51, 1985.

[94] Oster, G. F. and Wilson, E. O., *Caste and Ecology in the Social Insects*. Princeton, New Jersey: Princeton University Press, 1978.

[95] Deneubourg, J.-L., Goss, S., Franks, N., and Pasteels, J. M., "The blind leading the blind: Modeling chemically mediated army ant raid patterns," *Journal of Insect Behavior*, vol. 2, pp. 719-725, 1989.

[96] Deneubourg, J.-L., Aron, S., Goss, S., and Pasteels, J. M., "The self-organizing exploratory pattern of the Argentine ant," *Journal of Insect Behavior*, vol. 3, pp. 159-168, 1990.

[97] Edelstein-Keshet, L., "Simple models for trail-following behavior; Trunk trails versus individual foragers," *Journal of Mathematical Biology*, vol. 32, pp. 303-328, 1994.

[98] Burton, J. L. and Franks, N., "The foraging ecology of the army ant *Eciton rapax*: an ergonomic enigma?," *Ecological Entomology*, vol. 10, pp. 131-141, 1985.

[99] Beckers, R., Deneubourg, J.-L., and Goss, S., "Modulation of trail laying in the ant *Lasius niger* (Hymenoptera: Formicidae) and its role in the collective selection of a food source," *Journal of Insect Behavior*, vol. 6, pp. 751-759, 1993.

[100] Traniello, J. F. A., "Social organization and foraging success in *Lasius neoniger* (Hymenoptera: Formicidae): behavioral and ecological aspects of recruitment communication," *Oecologia*, vol. 59, pp. 94-100, 1983.

[101] Holldobler, B. and Wilson, E. O., "The multiple recruitment systems of the African weaver ant *Oecophylla longinoda* (Latreille) (Hymenoptera: Formicidae)," *Behavioral Ecology and Sociobiology*, vol. 3, 1978.

[102] Howse, P. E., *Termites: a study in social behavior*. London: Hutchinson and Co., 1970.

[103] Detrain, C. and Deneubourg, J.-L., "Scavenging by *Pheidole pallidula*: a key for understanding decision-making systems in ants," *Animal Behaviour*, vol. 53, 1997.

[104] Judd, T. M. and Sherman, P. W., "Naked mole-rats recruit colony mates to food sources," *Animal Behaviour*, vol. 52, pp. 957-969, 1996.

[105] Donnett, J. and Smithers, T., "LEGO vehicles: A technology for studying intelligent systems," presented at First International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 540-549, 1991.

[106] Bruckstein, A. M., "Why the ant trails look so straight and nice," *The Mathematical Intelligencer*, vol. 15, pp. 59-62, 1993.

[107] Edelen, M., "Commercial Robotics Platforms."

[108] Ermentrout, G. and Edelstein-Keshet, L., "Cellular automata approaches to biological modeling," *Journal of Theoretical Biology*, vol. 160, pp. 97-133, 1993.

[109] Watmough, J. and Edelstein-Keshet, L., "Modelling the formation of trail networks by foraging ants," *Journal of Theoretical Biology*, vol. 176, pp. 357-371, 1995.

[110] MIT Media Laboratory, "StarLogo on the Web," 2002,

[111] Proudfoot, Kekoa, "RCX Internals," 1999, http://graphics.stanford.edu/~kekoa/rcx/

[112] Resnick, M., Martin, F., Sargent, R., and Silverman, B., "Programmable Bricks: Toys to think with," *IBM Systems Journal*, vol. 35, pp. 445-452, 1996.

[113] Binder, J., "Bricx Command Center 3.3," 2003, http://hometown.aol.com/johnbinder/bricxcc.htm

[114] Baum, David, "Not Quite C," 2003, http://www.baumfamily.org/nqc/

[115] Katz, D.A., "Chemistry in the Toy Store, 6th Edition," 2002, http://www.chymist.com/Toystore%20part3.pdf

[116] Martin, F. G., "The Art of LEGO Design," *The Robotics Practitioner: The Journal for Robot Builders*, vol. 1, pp. 1-20, 1995.

[117] Cunniff, P. F., Dally, J. W., Herrmann, J. W., Schmidt, L. C., and Zhang, G., *Product Engineering and Manufacturing*. Knoxville, TN: College House Enterprises, LLC, 1998.

[118] Pitsco LEGO Educational Division, "LEGO Replacement Parts," 2003, http://www.pldstore.com/pld/catalog.cfm

[119] Gasperi, M., "LEGO Light Sensor," 2000, http://www.plazaearth.com/usr/gasperi/light.htm

[120] Michael Lachmann, "Mike's Lego CAD," 2002, http://www.lm-software.com/mlcad/