Collective Robotics: From Social Insects to Robots

C. Ronald Kube and Hong Zhang
kube@cs.ualberta.ca zhang@cs.ualberta.ca

University of Alberta
Department of Computing Science
Edmonton, Alberta, Canada T6G 2H1

October 1, 1993

Abstract

Achieving tasks with a multiple robot system will require a control system that is both simple and scalable as the number of robots increases. Collective behavior as demonstrated by social insects is a form of decentralized control that may prove useful in controlling multiple robots. Nature's several examples of collective behavior have motivated our approach to controlling a multiple robot system using a *group behavior*. Our mechanisms, used to invoke the group behavior, allow the system of robots to perform tasks without centralized control or explicit communication. We have constructed a system of five mobile robots capable of achieving simple collective tasks to verify the results obtained in simulation. The results suggest that decentralized control without explicit communication can be used in performing cooperative tasks requiring a collective behavior.

1 Introduction

Can useful tasks be accomplished by a homogeneous team of mobile robots without communication using decentralized control? The hypothesis implicit in this question is that such a *synergistic* robot system—one whose capabilities are greater than the sum of its individuals—can be created. Recent interest in task-achieving systems of multiple robots has led to several approaches in the design of their controllers. Among them, the Animat approach [46], which models whole, albeit simple, animal-like systems, offers a computational model in which perception and motor control may be studied. Using this approach, and motivated by several examples of cooperative behavior in social insects, we conjecture that decentralized control techniques can be used with multiple robots to achieve tasks in a cooperative fashion. In this paper, we describe our approach to collective robotics, the Collective Robotic Intelligence Project (CRIP), in which social insects are first studied, interesting examples of cooperative behavior are then simulated, and finally real robots are constructed on which we run our experiments.

The fact that man has yet to invent a highly autonomous robot capable of functioning in a changing environment has led researchers to propose the organization of several simpler robots into collections of task-achieving populations [9, 13, 33, 12, 25]. It has been conjectured that systems of multiple robots should prove more efficient and more fault-tolerant due to their number, more cost-effective due to their individual simplicity, and more flexible in their working configurations due to their redundancy, than a single robot [38]. These conjectures are shared by researchers in Distributed Robotic Systems (DRS) [30], especially those involved with Distributed Intelligence—often referred to as *Swarm Intelligence*—and Multi-robot robotic systems [5]. Research in DRS concentrates on three areas: the construction of physical systems, the use of communication to form cooperative systems, and the creation of task-accomplishing algorithms. Of the three areas of research, creating a system capable of displaying intelligent behavior from unintelligent units is dependent on progress being made in algorithms that produce "swarm intelligence" [5].

Before starting on the intellectual challenges connected with designing a team of multiple robots, let us first consider eligible tasks. Collective tasks for such teams are either noncooperative or cooperative. Noncooperative tasks gain efficiency in execution due to the parallel divide-and-conquer approach, but can be accomplished by a single robot given enough time. For example, a lawn mowed by a team of robots can also be mowed by one robot in a longer period. Other such tasks include sorting [13], searching [18], map making [32], material handling [15], and harvesting [19, 2]. On the other hand, *cooperative* tasks cannot be accomplished by a single robot and require the cooperative behavior of several machines working together. Such jobs include material-transport [34],

box-pushing [11, 4], tandem movement [27], and formation marching [41]. With the advances of micromachine technology promising to deliver multiple-robot applications as diverse as aircraft engine maintenance, microsurgery, and waste disposal [39], comes a growing need to develop control strategies suitable for a collective or cooperative behavior.

In our research, a bottom-up approach to controller design is taken, where tasks are accomplished using many homogeneous robots which function collectively in groups. In this regard we share many similar goals with Mataric [25]. We design feasible versus optimal solutions, with emphasis on ease of design. We design algorithms with an emphasis on locally sensed information, which allows for a decentralized control solution without the use of explicit communication.

Although it seems intuitive that communication between robots would allow greater cooperation, researchers have begun to investigate cooperative behavior without communication between robots [1, 34, 15]. The advantage of such a noncommunicating system lies in its ability to scale upwards without incurring a communication bottleneck as more robots are added. As found in Nature, tasks are solved with feasible versus optimal solutions, with a resulting reduction in the complexity of both the system's controller and the computation mechanisms of the individual robots. The results, presented by these researchers, based on simulation, suggest several strategies exist which produce task-achieving cooperative behavior.

Our own investigation of cooperative behavior, via a *box-pushing* task, has resulted in a decentralized system of five mobile robots capable of simple collective tasks without use of explicit communication [21, 22]. Our strategy involves *group behaviors*, and simple mechanisms to invoke them. Since explicit communication is not possible among our first system of reflexive robots, a form of positive feedback was necessary to ensure each robot was making progress in the task. The lack of any internal memory in the robots has led to problems with *stagnation* and *cyclic behavior*, both to be explained in the sequel, and motivated the development of our second multiple robot system.

In this paper, our research on collective behavior is described. Social insects provide the inspiration for our approach to collective robotics, and in Section 2 we discuss why their study is a useful starting point in the design of decentralized control strategies. In Section 3, based on these observations, we describe how group behaviors are used to control multiple robots and the mechanisms used to trigger them. In Section 4 we present our simulation results, obtained in our multiple robot simulator *SimbotCity*, and some initial results obtained with a neural net controller used for behavior arbitration. Verifying our simulation results by constructing real robots is a main requirement in our research and in Section 5 we discuss our first system of five mobile robots controlled using simple combinational logic, which eventually led to the design of a second system of 10 programmable robots. Finally, in Section 6 we discuss the problem of designing tasks suitable for collective robotics, and the direction of our future work.

2 Social Insects

One of Science's most challenging questions is how the behavior of large systems is generated from its individual components. Examples of task achieving societies abound in Nature. Social insects such as bees, ants and termites live in societies and exhibit collective behaviors in maintaining their societies [43]. Can the study of social insects motivate the design of decentralized controllers for robots? Several researchers [40, 13, 33, 10] have proposed models based on the study of social insects to control groups of interacting robots. By allowing Nature to guide us, by example, valuable lessons in population dynamics and its control may aid in the development of task specific collective robotic systems.

2.1 Sensing

Without a master architect to orchestrate the actions of individual ants, what initiates these behavioral programs? The answer may lie in the ant's sensing abilities. Behavioral biologists examine the sensor physiology of a species as the first step in understanding its behavior [43]. Behavior in social insects is thought to be like a stored program whose execution is a result of specific sensory stimuli. Moser [26] writes:

> Insects function like tiny robots programmed to do specific jobs. Their nervous systems act like biological computers; they are activated, as if by punch cards, when their receptors are stimulated. The external receptors respond to pressure, sound, light, heat, and chemicals.

The study of social insects has concentrated on four main species: ants, termites, bees and wasps. Of these, most is known about ants and bees. Considered the foremost social insect, ants are the most abundant with a population of roughly 10^{15}, or 1.5 million ants for every person on the planet.

Honeybees are the most studied insect species with a large repertoire of sensing capabilities. With its almost omnidirectional view, the honeybee sees fuzzy images of objects, but with a high sensitivity to broken patterns,

glimmering light, and sudden movement. Ants possess vision ranging from complete blindness in some species to bee-like acuity in others. Both bees and ants are able to estimate sunlight's plane of polarization providing directional information that allows them to navigate using the Sun, even on overcast days, due to their ability to see ultraviolet light.

Hearing of groundborne sound by bees and ants is accomplished through their feet. The sense of smell in both bees and ants is comparable to that of humans. The sense of taste is less sensitive, with coarser selection, than that of man. Bees and ants have an excellent sense of balance allowing them to orient to gravity at a constant angle.

Bees are capable of sensing temperature changes of one quarter of a degree, allowing them to maintain a constant temperature during honey production. A bee's ability to sense odor using its antennae allows it to estimate the difference between two sources [24]. It is with this array of sensing systems that bees and ants display their fascinating repertoire of behaviors.

These stored behavioral programs can be invoked by researchers using appropriate stimuli. In ants, corpse removal is a collective behavior invoked by chemical odor. Workers dispose of dead ants by carrying them from the nest to a refuse pile. Wilson *et al.* [44] were able to invoke the same behavior in ants by treating bits of paper with acetone extracts of dead corpses. In fact, by daubing small amounts of acetone extract on live ants, they too were carried away by nestmates and dumped on the refuse pile! Thus, stimulus sensing serves to trigger stored patterns of behavior and ultimately forms the basis for the behavior-based approach to robot control.

Without a complex computational mechanism how can insects' sensory mechanisms solve the myriad problems presented by the peculiarities of their environment? Wehner suggests "matched filters," which are spatially placed receptors specialized to some feature in the environment, as Nature's simple solution [42]. An example cited is the flat world of the Saharan salt pans inhabited by desert ants. This environment, dominated visually by the horizon, is particularly well suited to the structure of the ant's optics, in which a rather large number of horizon-looking photoreceptors are found. Wehner explains it is these "band-like zones" of receptors that provide the high degree of visual acuity. "These 'visual streaks' are perfectly aligned with the horizon, irrespective of the load an ant may carry" [42].

Animals living in open environments are all found to have visual streaks. Among them, crabs were found to have visual systems even more elaborate with receptor spacing varying at right angles to the visual streak [48]. This variation in receptor spacing results in the stimulation of a constant number of receptors. This allows the crab to detect objects that appear near the horizon of a constant absolute size without regard to its distance away. Because of this spacing, and the eye's stabilization against pitch and roll axis displacements, retinal images of objects larger than the crab appear above the eye's horizontal, and those objects smaller appear below [48]. This would allow for a flee behavior to easily determine predators based on size alone. Thus, these carefully evolved sensory systems tuned to features unique to the animals environment produce robust behavior without complex processing.

Given the simple stimulus-response mechanisms involved in behavior activation, it is a wonder that tasks are achievable by these insect societies. When the society is viewed as a whole, a behavioral complexity emerges that seems to be more than just a composite of individual behaviors.

2.2 The Social Machine

An insect colony is often referred to as a *superorganism* due to the resemblance between the many social phenomena it displays and the physiological properties of organs and tissues. These behavioral attributes of the superorganism are an emergent property resulting from the interaction of the colony's many members each displaying their own simple repertoire [43]. Deneubourg and Goss [14] raise the question of whether the colony's behavioral complexity lies within the individual members or between them? Being able to deduce collective activity from individual behavior is one of the main problems faced by behavioral biologists; since as Pasteels *et al.* pointed out "collective behavior is not simply the sum of each participant's behavior, as others emerge at the society level" [29].

As the superorganism's individuals are brought into focus, one is startled to find the display of antagonistic actions involved in a collective activity. Wilson provides an example in the process of moving a nest [43]:

> As workers stream outward carrying eggs, larvae, and pupae in their mandibles, other workers are busy carrying them back again. Still other workers run back and forth carrying nothing.

Honeybees exhibit the same disarray in the construction of comb cells. Workers, in search of pieces of wax for cell construction, will usually tear down walls that their nestmates are in the process of building [23]. This seemingly chaotic activity usually results in a well constructed nest and is an example of Nature's feasible versus optimal solution approach. So how is it possible then, for the colony to display such a purposeful collective behavior? The answer may lie in the positive feedback mechanism responsible for collective decision-making.

One can not help but ponder, when viewing a two meter high termites' nest, the intelligence behind its construction, especially in light of the fact that, as Sudd [37] describes, "each of the grains of soil of which the nest is built has been carried separately and placed by a termite perhaps half a centimeter long." The mechanism involved in this task-achieving collective behavior is *allelomimesis*, or positive feedback, which Deneubourg and Goss roughly translate as "do what my neighbour is doing"[14]; coupled with a set of common simple rules and invoked by sensing a stimulus, this decentralized system generates a colony-level response characteristic of the behavioral attributes often ascribed to a superorganism.

How does the behavior of such a system arise? Sudd suggests that cooperative behavior is a result of the application of a three phase approach of disorder, search, and order. The effect of each is judged by positive feedback communicated through the work itself. Thus, order "arises through the trial of many possibilities" [37]. An example is nest construction by Weaver ants [45, 37]. Nest walls are constructed from folded green leaves held together by sticky larval silk. In order to fold a leaf, ants begin by spreading over the leaf's surface and randomly pulling at any graspable edge. Some edges are more easily turned, while unsuccessful efforts are quickly abandoned, causing a search for a new edge. The success of a turning edge reinforces the continuance of the effort. The result is an ordered and collective effort of pulling on successfully turned edges, with a folded leaf as the final outcome.

The effect of positive feedback and the simple rules shared by each individual can result in the performance of the system exceeding the sum of its parts. An example is the prey-transport task. Franks conducted experiments in group retrieval of prey by army ants. The evidence gathered suggested that workers in a retrieval group were "able to assess their own performance and their potential contribution to a group effort" [16]. Franks cites a simple algorithm used in this prey-transport task [17]:

> If there is a prey item in the trail moving below the standard retrieval speed, and you are not carrying an item, then help out; otherwise continue.

Franks attributes the "superefficiency" of the group to its ability to overcome rotational forces in the object being transported; forces too large for the individual to successfully balance on its own [16].

Simple shared rules also seem to play a part in a honeybee colony's collective ability to select the most profitable nectar sources. Seeley *et al.* [31] suggest if foragers of a honeybee colony all share the same rules for food gathering, and adjust their response threshold between recruitment and abandonment accordingly, a collective response will result. The response threshold adjustment results in the bees varying their foraging behaviors, such as the strength of their waggle dance—used to recruit other bees—and how often they visit or decide to return to the nectar source. How the bee is able to compare nectar sources for profitability—determined by such variables as sugar concentration, distance from the hive, difficulty in acquisition and amount of nectar—is not known, but it is presumed their nervous system is somehow calibrated to differentiate between low and high sources [31].

As biologists uncover some of Nature's solutions to the sensing complexities faced by social insects, we stand to gain much by the study of this natural example of decentralized control. In the next section we present several mechanisms, motivated by social insects, used to control a team of robots.

3 Collective Behavior

Constructing tools from a collection of individuals is not a novel endeavor for man. A chain is a collection of links, a rake a collection of tines, and a broom a collection of bristles. Sweeping the sidewalk would certainly be difficult with a single or even a few bristles. Thus there must exist tasks that are easier to accomplish using a collection of robots, rather than just one. Of course the difficulty increases when the individuals are somewhat autonomous, and there lies the challenge. How do we create an intelligent task-achieving collective behavior from a group of simple robots? By studying Nature's many examples of task-achieving collective behavior we hope to uncover some of her more useful mechanisms.

3.1 Controlling Robots Using Group Behavior

A group behavior is the task-achieving activity for which the multiple robot system is designed, and it consists of a common set of rules for accomplishing the task. The group behavior is simply the activity all the robots are engaged in, and some tasks may consist of two or more group behaviors, like synchronized steps in a dance performed by a group of dancers. A simple example of a collective task is emptying a room of heavy furniture which consists of two group behaviors: lifting-furniture and moving-furniture.

From the examples seen in social insects, several mechanisms exist which may be used to invoke the group behavior. Mechanisms are ways to control the system of robots and may consist of shared goals, as in the common task mechanism, useful behaviors, such as *following* that keep robots together in a floor-washing group behavior, or cues in the environment, which may serve to either invoke a single group behavior or cause the transition between two

group behaviors. These mechanisms by no means represent a comprehensive set, but rather serve to illustrate our initial exploration into task-achieving collective behavior.

The first mechanism is a *common goal* shared by all the robots in the system. Such a single purpose system is controlled by having only one activity to choose from. Leaf folding by Weaver ants could be considered such an activity, as could the previously described corpse-removal behavior. In the sequel we examine how a common goal, in the form of box-pushing, can be used to control a group of robots.

Collective tasks requiring groups of robots in close proximity may make use of a *follow behavior* to accomplish the group behavior. Several examples of a follow behavior can be found among ants. Odor trails, tactile sensing used in tandem running, and visual stimulus used in rapid running are examples of group movement, which results from the activation of a follow behavior. A follow behavior may be used to maintain a formation and could prove useful in a system designed for distributed environmental sensing. These mobile sensors, traveling in herds, could spatially cover a search area while gathering data. For tasks which involve some dynamically changing physical parameter such as the size of an area covered by a liquid spill, or the advancement of a rapidly spreading fire front, quicker response by the system is possible when robots are kept together.

For tasks which require a dynamic stimulus to invoke the group behavior or which are accomplished using a sequence of two or more group behaviors, an *environmental cue* is used to control the transition between behaviors. This mechanism is found in many examples from the collective activity of ants. Food collecting behaviors are governed by the visual cue of dawn and dusk. Termites use cues to control the transition between vertical construction of columns and their bending toward one another in the formation of an archway.

The construction of archways by termites begins with the random movement of pellets which eventually results—by a seemingly random occurrence—in the placement of a second pellet on top [43]. Apparently, termites have a preference for this structure and continue to place pellets on top constructing a column in the process. The next step, in the construction of an archway, requires a second column nearby. At some point in the column's construction termites, working on separate columns, begin to bend the column towards each other thereby forming an archway. The environmental cue that causes the transition to the *bending* step in the task is unknown, although it is hypothesized by Wilson to be olfactory in nature [43]. In this manner cues serve to either initiate the group behavior or serve in a regulatory manner guiding transitions between group behaviors.

There is some evidence to suggest that some species of ants alter their behavior using *group detection*. Worker ants were found to excavate soil and attend larvae at a higher rate while in large groups. In fact, the stimulus responsible for this altered behavior was found to be carbon dioxide [20]. Wilson also found that worker ants kept in solitude did not respond to the natural alarm substances of their species. However, when placed among hundreds of their coworkers they were found to respond normally to the same alarm substance. This could prove to be useful as a way of invoking a group behavior once a collection of robots had formed. Consider the task of leveling the ground by a number of small bulldozer robots. The effect of a large blade, by a number of bulldozers with small blades, is not realized until the group has configured itself in an appropriate formation traveling in the same direction. Thus the formation of robots itself invokes the group behavior.

Collective behavior by social insects is an area of research rich in examples of mechanisms suitable for implementation in collective robotics. Simulation is our next step in the investigation of some of these mechanisms, and it ultimately results in their implementation in real robots.

4 Simulation

Inspired by the examples of cooperative behavior found in social insects, we wish to simulate the common cooperative task of box-pushing. The objective is to locate and push a large box too heavy to be moved by a single robot (see Figure 6). As such, it will require the cooperative effort of at least two robots both pushing on the same side to move the box. Like the leaf folding task, all the robots involved will be interested in one common goal: box-pushing. To simulate such a task we have created a simulation environment, called *SimbotCity*, in which to model a small population of robots. The robots are modeled as a set of sensors, actuators and behaviors, combined in a control architecture that uses either a subsumption [7] fixed priority or an Adaptive Logic Network [3] behavior arbitration mechanism. The resulting herd of box-pushing robots are capable of accomplishing their task, and the problems of stagnation and cyclic behavior are partially solved by examining the social insects and their positive feedback mechanisms.

4.1 SimbotCity: A Robot Population Simulator

Accomplishing tasks using a decentralized system of autonomous robots without explicit communication requires each robot's control algorithms to make use of local information only. Acquired by the robot's onboard sensors,

this information mediated by behavior must be sufficient to ensure that the entire system of robots converges towards the desired goal. A task to be realized by such a system defines the sensing requirements. For example, our box-pushing task requires the ability to sense the box, in order to locate it, sense other objects including other robots in order to avoid collisions, and sense task progression in order to assess performance. Can these decentralized control mechanisms accomplish cooperative tasks?

To successfully function as a group our system will need some form of cooperation. Cooperation might simply equate to noninterference as suggested in [47] without explicit communication, or may involve a more elaborate explicit form of communication. Can cooperative tasks be accomplished without explicit communication?

Allocation, also referred to as *density dependence* by Brooks [8], is the problem of how many robots to use in a collective task. Since the box-pushing task can be accomplished with two or more robots, what is the optimal number of robots to be employed for the task, given a performance measure?

Our initial exploration of these questions resulted in the creation of our robot population simulator *SimbotCity*, in which we have simulated the box-pushing task. Robots are modeled as a collection of sensors, actuators and behaviors presently combined using one of two arbitration mechanisms. A configuration file specifies the number of robots along with their initial positions. A simulation may be run continuously or single stepped while each robot's sensor readings are displayed. Performance is measured as a function of simulated time steps versus distance the box has moved. The robot's model is based on our current capability to construct its physical counterpart.

4.1.1 The Robot Model

A population consists of a group of robots with each robot composed of a sensor model, an actuator model, and a behavior model. Models can be further subdivided into model *types*. For example, sensor types may consist of *infrared* for near obstacle detection, *light* for brightness calculations or *sonar* for long range distance measurements. The box-pushing task makes use of three sensors: a goal sensor, an obstacle sensor, a robot sensor, and two actuators: left and right wheel motors (see Figure 1). There are five behaviors: a *goal* behavior directing the robot toward the box, an *avoid* behavior to handle collisions, a *follow* behavior allowing one robot to follow another, a *slow* behavior which adjusts motor speed preventing rear-end collisions, and a *find* behavior used in exploration.

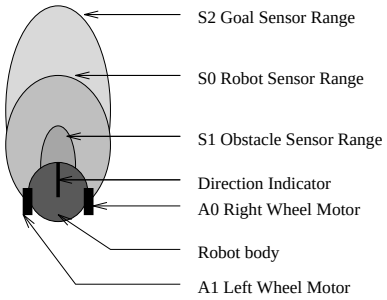

Figure 1: The box-pushing robot model. Each robot is equipped with a goal sensor, an obstacle sensor, a robot sensor, and two actuators: a left and right wheel motor.

Sensors Implemented as an abstract data type (ADT), the sensor model has six attributes: sensor number, type, sensor direction, view angle width, an input value for active sensors, and an output value. Processing is based on sensor type with macros providing access to the sensor's data structure. Currently there are five sensor types corresponding to the physical sensors available in our lab. These types are: sonar, acoustic, infrared, light and switch. Since sensor types may only represent physical sensors, hardware implementation is easier and ensures the simulated robots may be built using the same sensing techniques.

Actuators The actuator model is also implemented as an ADT with five attributes: actuator number, type, position on the robot, input value, and on/off switch. Accessing and processing is also based on type, with four types of actuators available: motor, hand, plow, and solenoid. For the box-pushing robots only the motor type is used, with one each for the left and right wheel motors. Steering the robot is achieved by turning one motor on at a time. For example, to turn right the left motor is turned on with the right motor turned off. This switching takes place for each simulation time step. Actuator and robot dynamics are not modeled.

Behaviors A behavior maps sensor inputs to actuator outputs to define a stimulus-response relationship. For example, a Braitenberg [6] vehicle that seeks light may be created by cross connecting the left-light sensor with the right wheel motor illustrated in Figure 2.

Sensors provide input to a behavior which then processes the data to provide output commands to actuators. During a simulation time step, each behavior reads its connected sensors and calculates an appropriate response, with the resulting command sent to a behavior arbitration module. Behavior processing may include thresholding where only signals of a certain strength are acted upon.

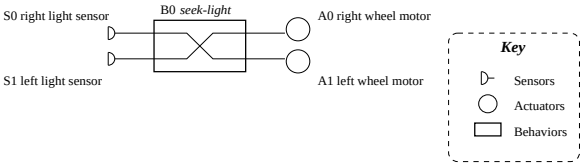

Figure 2: The seek-light stimulus-response behavior in a simple Braitenberg vehicle is implemented by cross-connecting the opposite side sensor and wheel-motor actuator pairs.

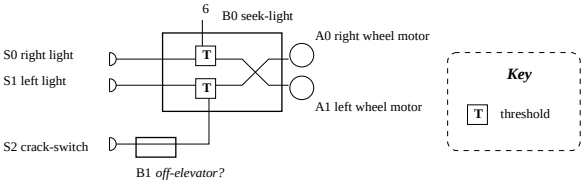

Figure 3: Behavior processing may include simple thresholding, which may be of fixed value (i.e. ≤ 6) or set by another behavior.

Thus our Braitenberg light-seeking vehicle may have a preference for very bright lights. Thresholds may be externally set by other behaviors or fixed as illustrated in Figure 3.

Memory mechanisms may also be incorporated into a behavior. For example, a progress monitoring behavior could be created which counts to some predetermined amount of time when it then becomes active. However, if progress is being made in the task, then a positive feedback stimulus constantly resets the time counter never allowing the behavior to become active as illustrated in Figure 4. This addition of a memory device creates what Wilson has termed a "Virtual stimulus-response" and allows behaviors to incorporate "intention memory" [46].

4.1.2 Behavior Arbitration

In a bottom up approach to designing robot control systems, a problem arises as how to best arbitrate conflicting actuator commands. One approach, the subsumption networks [7], uses a fixed priority assignment between behaviors. The resulting actuator commands are simply those belonging to the highest priority behavior. This requires the designer to consider all behaviors in the control system and decide on how to assign priority. As the number of behaviors increases, so does the burden on the designer to make a priori decisions about behavior arbitration.

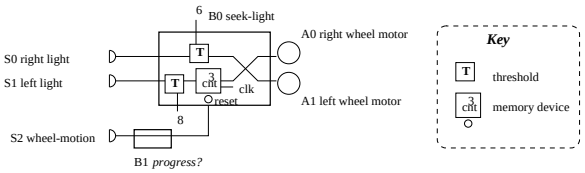

Figure 4: A progress "virtual stimulus-response" behavior monitors wheel-motion and constantly resets an interval counter.

An alternate approach to the behavior arbitration problem which we have been exploring [22], is to formulate the problem as a pattern classification problem on which an Adaptive Logic Network (ALN) [3] can be trained. ALNs are neural networks formed in binary tree configurations with nodes eventually assigned either the Boolean function AND or OR. The base of the tree receive its input comprised of behavior actuator commands, with the root of the tree forming a single output bit. A supervised training procedure takes a subset of all possible behavior outputs and classifies each of them into one of many actuator commands. If the behavior output subset is truly representative of all possible outputs and a functional relationship between behavior outputs and actuator commands exist, then the ALN will learn the relationship and correctly classify yet unseen behavior outputs to the desired actuator commands. The power of the technique only becomes evident as the ALN's input space (behavior outputs) becomes large. In the sequel we will examine both approaches to behavior arbitration using the box-pushing collective task.

4.2 Box-Pushing

The objective in the box-pushing task is to locate and move a large box using a group of robots. The task is designed such that moving the box requires the net force of at least two robots both pushing on the same side. We have experimented with two approaches in simulation, implementing the first in hardware. The first was a subsumption style behavior-based controller with a fixed priority behavior arbitration. This controller was then further simplified and implemented with five mobile robots discussed in the sequel. We then came back to our simulation environment and revisited the problem using reflexive behaviors and an ALN for behavior arbitration. This second approach allowed us to train the controller with a supervised training algorithm. It was not our intent to do a strict comparison between the two controllers, and although the ALN proved to be less efficient in terms of accomplishing the task, it was much simpler to design.

4.2.1 Subsumption Networks

To accomplish the task each simulated robot was given three sensors, two actuators, and five behaviors illustrated in Figure 5. The lowest priority and default behavior is *find*, which requires no sensor inputs and causes the robot

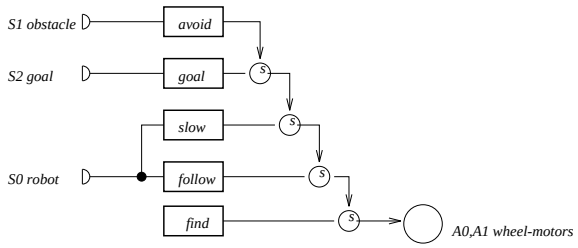

Figure 5: The box-pushing robot's behavior architecture. A behavior's actuator commands may be suppressed (the circles with an 'S') and replaced by those of a higher priority.

to move in a large arc. Its output may be suppressed (the circles with an "S") and replaced with actuator commands from *follow* which causes robots to form groups by following other robots. While following, if a robot gets too close to another, the *slow* behavior is activated and reduces the robot's speed while active. The *goal* behavior, activated by a goal sensor, directs the robot towards the goal only to be suppressed by the highest priority *avoid* behavior if obstacle collision is imminent. Figure 6 shows the box being moved after several steps into the simulation.

The design of the box-pushing controller begins by specifying the task's sensing requirements. Collision free movement will require an obstacle sensor; to follow other robots requires a robot sensor; and locating the box will require a box or goal sensor. Next, a default behavior is chosen. In this case, a *find* behavior that moves the robot forward on a gradual arc produces movement. Starting from an initial configuration this single behavior controller creates motion which often results in collisions. A *follow* behavior is added which uses a robot sensor to direct the robot toward its nearest sensed neighbor. Once follow becomes active herds begin to form and are maintained by adapting the follow behavior with a *behavior preference*.

Behavior preferences adapt a behavior by filtering sensor input to suit the behavior's state. In the case of the follow behavior, the sensor input is filtered by considering a smaller field of view—similar to narrowing focus of attention in visual tasks—while engaged in following. This eliminates distractions from passing robots moving in opposite directions. This mechanism can also allow a collision-avoidance behavior to pass through a narrow doorway by having the doorway behavior adjust avoid's behavior preference.

Without velocity control, robots moving the same speed would not form groups, as distant robots could never catch up. For this reason, a simple two speed system is implemented by having robots traveling in herds move at the slower speed. Thus a *slow* behavior is added which reduces the robot's velocity whenever neighboring robots

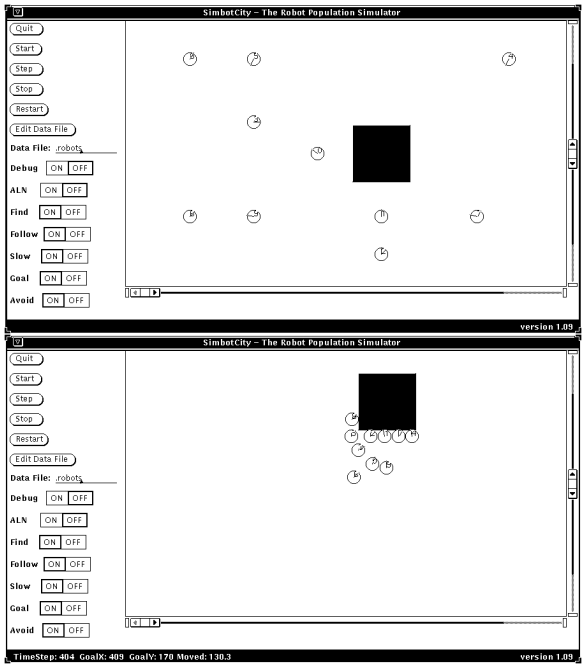

Figure 6: The initial configuration of the cooperative box-pushing task (top) and after 404 simulation steps in which the box has been moved 130 units upwards. The robots (circles) must locate and push the large box, which is too heavy to be moved by a single robot; therefore requiring the cooperative effort of at least 2 robots pushing on the same side.

are detected.

To prevent collisions an *avoid* behavior is added which becomes active and stays active as long as the obstacle sensor has detected an obstacle. Collisions are avoided by turning a fixed number of degrees in the opposite direction at each simulation time step. The range of the obstacle sensor is 1.5 times the robot's diameter. Keeping task behaviors active, only as long as their sensory stimulus is present, can solve some problems concerning the spatial distribution of robots in a collective task. For example, the box-pushing task requires the robots to spatially distribute themselves along the sides of the box. Finding an open spot on the box can be accomplished by a robot whose persistent goal behavior is only momentarily dissuaded by its noninterfering avoid behavior. As soon as an open spot appears, avoid switches off allowing the goal behavior to move the robot in and fill the vacant spot. To locate the box, robots are equipped with a goal sensor capable of detecting the box at a distance of six times the robot's diameter. Similar in design to the avoid behavior, the *goal* behavior turns the robot a fixed number of degrees toward the goal while active.

The task is accomplished once several robots have located the box and collectively pushed it off the edge of their world. Robots unfortunate enough to be caught pushing on the opposite side of a herd are quickly pushed backward. Task progression is implied in the forward motion of a robot and a robot moving backward immediately turns away from the box to assume a new position that allows forward motion.

Two problems occur in which the box-pushing task does not progress. The first involves *stagnation* in which a number of robots equally distribute themselves around the box. In this situation the forces around the box cancel each other. The solution is to introduce a behavior to monitor positive feedback which may be determined, in this case, by a constant forward motion. As long as a forward motion is achieved every n time steps the behavior remains inactive. However, once n time steps has passed since the last forward motion was detected the behavior becomes active with a random motion to break the stagnation. The second problem involves cyclic behavior—really another form of stagnation but with motion—and can be illustrated by a robot moving in a cyclic pattern. Although movement is occurring, no progress is being made toward the goal. A method to detect cyclic behavior remains an interesting challenge.

4.2.2 Adaptive Logic Networks

Adaptive Logic Networks (ALN) are a kind of neural network designed to synthesize functions using a binary tree of logical AND and OR operations, where complementation is allowed only at the input level (see [3]). They are particularly well suited for Boolean input vectors, in contrast to the more familiar backpropagation networks, whose inputs are continuous variables, and whose outputs are derived by applying a sigmoidal function to a sum of weighted inputs. The adaptive, or learning process, adjusts a node's logical function (equivalent to the weight adjustment in backpropagation) based on a training set of input vectors that are representative of the input space. Assuming generalization occurs (i.e. the training converges) the network will correctly classify yet unseen input vectors. Since trained ALNs are binary trees of AND and OR functions, they are easily implemented in Programmable Array Logic (PAL).

As previously mentioned, behavior arbitration is the process of deciding which behaviors have control of the actuators resources at any given moment. When the number of behaviors is small, as in our box-pushing controller, deciding on a behavior's relative priority is easy. However, as the number of behaviors increases their relationship to each other during task execution is less clear, and we begin having difficulty deciding how to assign priority. On the other hand, as an observer of a simulated collective task, it is possible to decide what the team of robots should do at any point, but difficult to specify how the behaviors should be arbitrated. Previous experience with ALNs suggested it may be possible to characterize behavior arbitration as a pattern classification problem which an ALN was suitable for.

The architecture illustrated in Figure 7 was used to train four ALNs, one for each motor control output bit. The actuators are the same left and right wheel motors each with two control bits with their corresponding four motor commands shown in Table 1. Training the ALN in a supervised manner which confirms correct responses to a given input pattern of actuator commands results in a tree shown in Figure 8. Nodes are represented as circles with the letters A, O, L, and R standing for the logical functions AND, OR, LEFT and RIGHT. Upon implementation, LEFT and RIGHT nodes are replaced with connections to the left or right subtree. Input leaves are represented as squares and receive actuator commands. Compliments of an actuator command are represented by a small circle on top of the leaf. Since there are five input bits (S_1 to S_5), the four ALNs must learn 32 possible outputs. Although the input space in this example is small, and easily handled by table lookup, we are interested in testing the feasibility of the approach.

Training the ALN is accomplished using a supervisory controller illustrated in Figure 9. The supervisory controller sends motor commands to the simulated robots in the same format as the ALN controller's motor commands. For example, suppose we are trying to create a

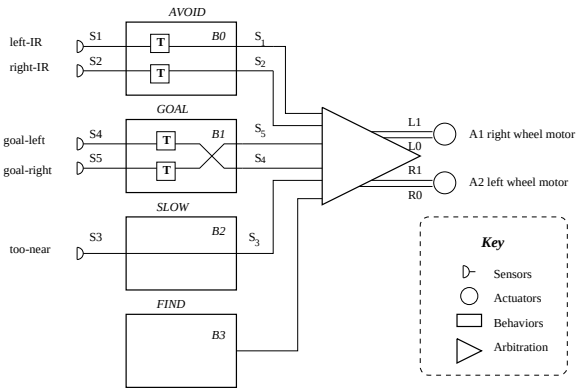

Figure 7: Architecture used to train four ALNs. The labels S_1 to S_5 correspond to avoid-left, avoid-right, slow, goal-left and goal-right motor commands.

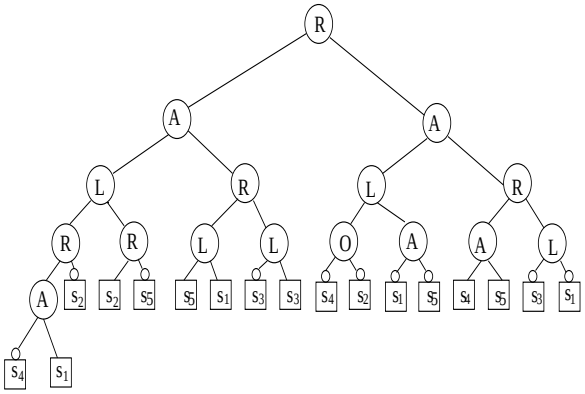

Figure 8: A trained Adaptive Logic network with 17 input leaves.

M_1	M_0	Motor
0	0	Stop
0	1	Half speed
1	0	Full speed
1	1	Half speed

Table 1: Motor commands for left and right wheel motors; where $M_i = \{L_i, R_i\}$

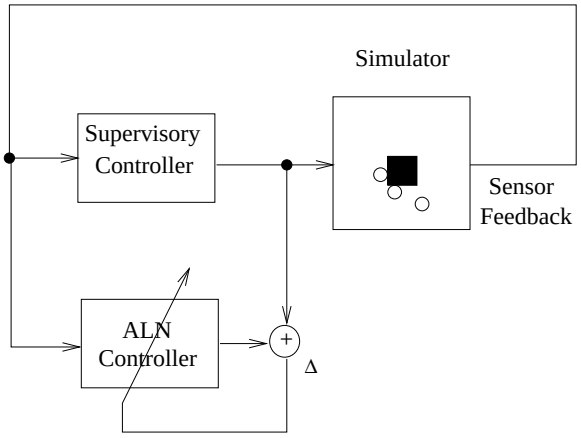

Figure 9: An illustration of how the ALN controller is trained. A supervisory controller commands the simulated robots whose sensor feedback (mediated by behavior) is sent to the ALN being trained. Adaptation takes place as the difference between the desired output of the supervisory controller and the ALN's actual output. Training stops once a prespecified competence is reached.

decentralized controller to be used in a formation marching task. The problem of how each robot should move is trivial for a centralized controller which simply issues the same motor command to each robot. The supervisory controller is implemented as a centralized controller instructing the simulated robots to move, while their sensor feedback, mediated by behavior, is sent to the decentralized ALN controller being trained. The ALN uses the difference between its motor commands and that of the supervisory controller to adapt its response. Once trained, the decentralized ALN controller is tested independently (see Figure 10) and then transferred to a microprocessor controller. The simulated robots are then tested using real robots with the ALN controller implemented using PALs.

For the box-pushing task, Figure 11 illustrates the relationship between the number of robots used and the task's completion time, measured as the number of simulation time steps to move the box 100 units of distance. An ALN controller which achieved 100 percent in the training phase was chosen for the test. Robots were placed in random positions on the left side of the box. As can be seen from the graph, as the number of robots increases, the task execution time decreases until too many robots are used. In this case, "too many robots" is related to the number of robots that fit on a box side, but how does one determine this number for any given task? Table 2 lists the data to produce the graph as well as the suboptimal ALN controllers. The last column on the right shows the average number of simulation time steps taken to move the box 100 units of distance, and only takes into account

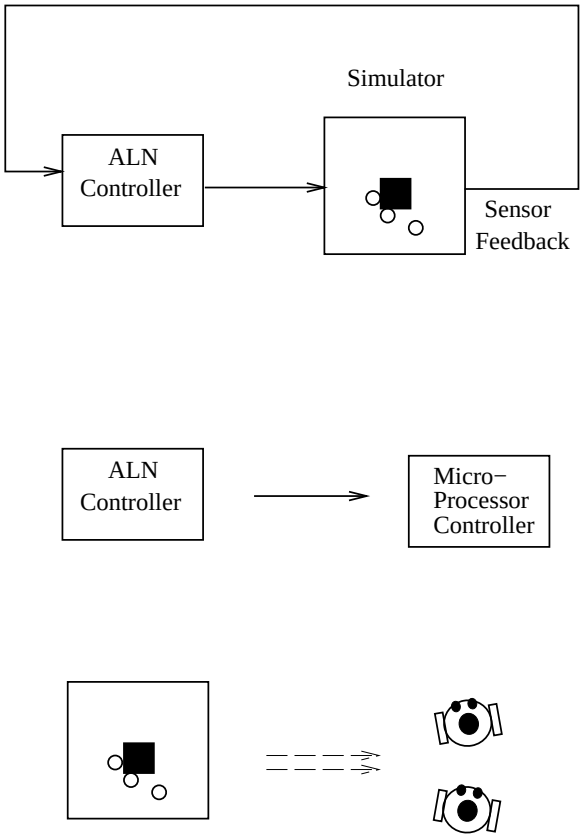

Figure 10: Once trained the ALN controller is tested using the simulator (top illustration) and then implemented using a microprocessor (middle). The simulated robots are then constructed and controlled using the ALN controller (bottom).

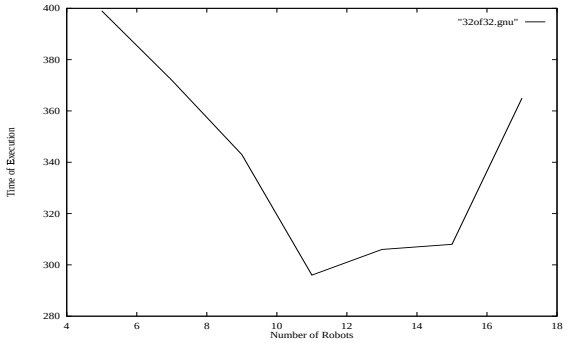

Figure 11: The effects of population size on the box-pushing task.

ALN Quality	*Number of Robots Participating in Task*							
	5	7	9	11	13	15	17	19
32	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0
30	2	1	0	0	0	0	0	0
29	6	4	6	6	4	5	5	6
28	3	2	4	3	2	3	3	5
27	10	8	8	9	7	6	9	9
26	15	5	11	7	8	7	10	6
25	15	11	8	15	8	7	10	6
24	15	8	9	10	6	7	7	9
23	15	15	13	15	11	14	14	14
22	15	14	14	15	11	9	15	15

Table 3: Chances of failure in 15 trials.

the tests which completed. Table 3 lists the number of failures in 15 trials for ALN controllers with 100 percent quality (i.e. 32/32 correct classifications) to 68.7 percent quality (i.e. 22/32 correct classifications).

In the next section we discuss our hardware implementation of the box-pushing task by a system of five mobile robots controlled in a simple reflexive manner using combinational logic.

5 Real Robots

Based on our simulation experiments of the box-pushing task, we now know three things about the mechanisms involved that should prove feasible in its implementation. First, to control the system of multiple robots in a cooperative task, without using any communication between the robots, we employ two simple rules that govern the interaction between their behaviors:

- avoid interfering with another robot;

ALN Quality	*Number of Robots Participating in Task*								Average
	5	7	9	11	13	15	17	19	
32	399	372	343	296	306	308	365	342	341
31	399	396	401	305	297	307	449	323	360
30	406	378	397	316	337	342	409	331	365
29	445	385	395	348	323	349	394	375	377
28	456	394	400	314	325	395	402	347	379
27	548	364	413	427	343	352	411	383	405
26	N/A	576	494	520	504	471	777	515	551
25	N/A	369	407	N/A	501	339	355	368	390
24	N/A	369	438	336	339	339	291	358	353
23	N/A	N/A	586	N/A	462	662	333	639	536
22	N/A	621	403	N/A	684	544	N/A	N/A	563
Average	442	437	409	415	420	370	485	371	

Table 2: Results of simulation on the effects of population size and the box-pushing task. Note: N/A refers to cases in which all tests failed to finish.

- work toward a common task while observing the first rule.

This provides a decentralized control strategy for the system on the whole. Second, when designing individual behaviors for the collective task, the behavior need only be active as long as the stimulus in the environment—for which the behavior's sensors were chosen—is present, as can be seen from the previous example, in Section 4.2.1, on spatial distribution. Stuart's study of nest wall repairs by termites [35, 36] showed that wall repair behavior ceased once the stimulus that caused the behavior (i.e. a hole in the nest wall) was removed. Third, a mechanism to monitor task progression is needed to ensure stagnation does not occur. In simulation, progress was defined as forward motion and a backward motion sensor was used to activate the avoid behavior.

Having modeled our simulated robots with the intention of later constructing them has simplified our implementation. Each robot has a left and right wheel motor providing a differential drive mechanism. Obstacles are sensed using a left and right infrared sensor that provides a logic low signal when infrared energy is reflected back from an obstacle. To locate the brightly lit box, left and right photocells are used with adjustable thresholds.

Behaviors are implemented in hardware using simple combinational logic. By adjusting the photocell threshold, to switch on under ambient lighting conditions, a default *find* behavior is created as uneven lighting conditions cause the robots to wander their environment. By confining the robots to a small area, both the *follow* and *slow* behaviors—used to form groups in wide open spaces—were unnecessary, and simplified the experimental setup. The *avoid* behavior is created by thresholding the infrared sensors, and connecting the left and right outputs directly to the left and right wheel motors. The *goal* behavior is fashioned in a similar manner by thresholding the left and right photocells and cross-connecting the motor outputs. The completed control architecture is illustrated in Figure 12. Behavior arbitration is a simple fixed priority between behaviors, with avoid having the highest and find the lowest priority, and is implemented in combinational logic illustrated in Figure 13.

A system of five box-pushing robots were constructed based on the architecture illustrated in Figures 12 and 13. The system was tested using a variety of initial configurations first in simulation and then compared with the actual robots. Video recordings were made for later review. The robots located the brightly lit box, converging upon and pushing it in a number of directions depending on the number of robots on each side (see Figure 14). The progress sensor was implemented as a micro-switch which activated the avoid behavior when the robot was pushed backward and moved the robot away from the side. The avoid behavior kept robots from colliding most of the time, with collisions occurring when sensors missed detection due to their limited field of view.

The system demonstrated that a cooperative task is possible using a simple common task and noninterference control mechanism; however, it also pointed out the importance of progress monitoring behaviors to prevent problems with stagnation and cyclic behavior. The system demonstrates the feasibility of cooperative tasks without explicit communication in a decentralized system, a point we are currently exploring with our new system of 10 robots.

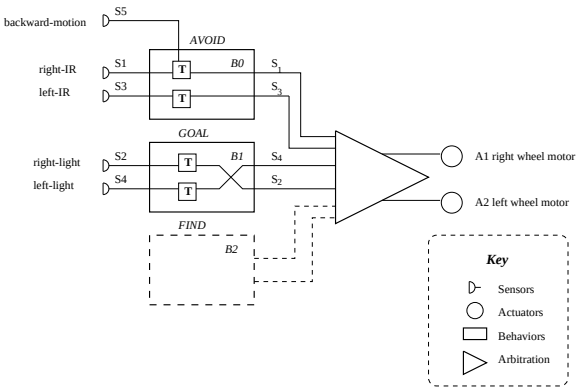

Figure 12: The box-pushing robot's control architecture. Behavior arbitration is handled using a fixed priority subsumption network.

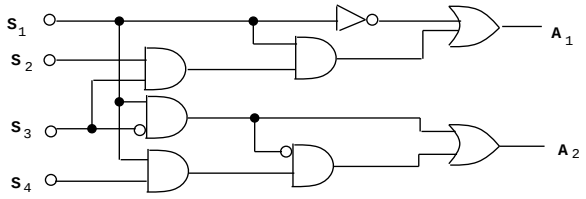

Figure 13: The box-pushing robot's arbitration circuit using simple combinational logic.

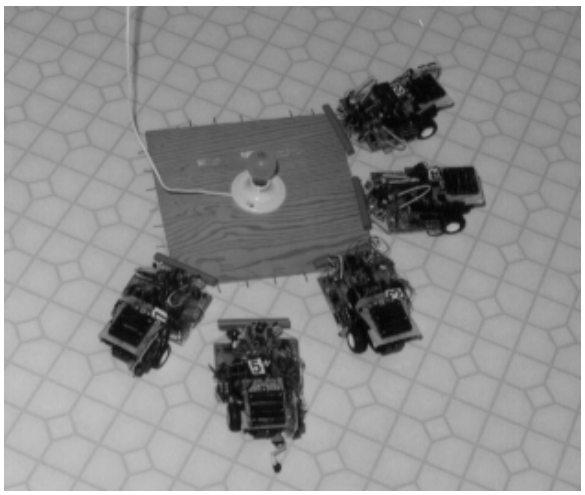

Figure 14: Five box-pushing robots moving a brightly lit box in a cooperative manner.

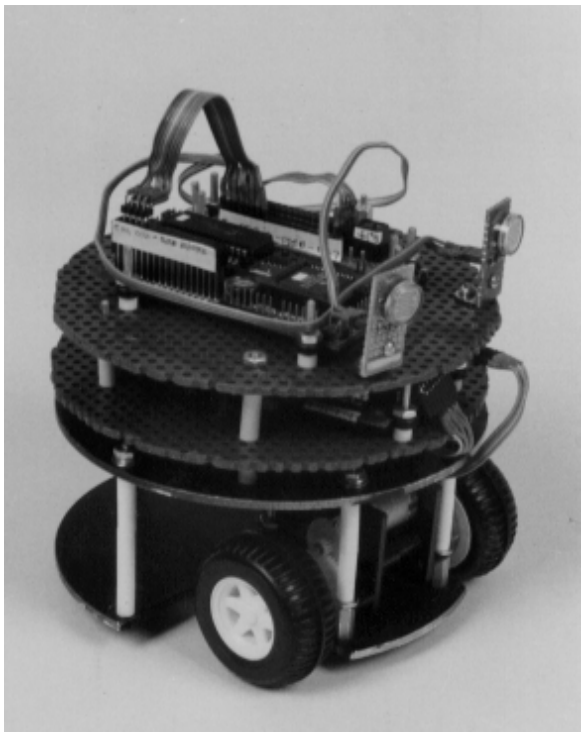

Figure 15: New prototype robot constructed with plug-in modules.

The control logic for our system of five robots is hardwired making changes tedious to implement. We are therefore currently in the process of constructing a new system of robots built with plug-in modules (shown in Figure 15). This new series of 10 robots is controlled using a multitasking processor. A robot can literally be assembled from its individual components in five minutes without use of tools making its configuration of sensors easily changeable. Our plans for the new system are to investigate formation marching and to extend the box-pushing task to a transport task.

6 Discussion

Designing "intelligent" autonomous robots that accomplish useful tasks is a challenging and still elusive goal of scientific research. Yet its pursuit has led to several new and unconventional approaches. Among them, achieving tasks through the use of a system of multiple robots has an appeal that captivates the imagination because of its analogous relationship with the task-achieving populations of social insects. The main hypothesis of the approach lies in the hope that such a population of machines will achieve a higher level of competence due to an emergent property of the system making it more than just the sum of its parts.

It is also this hope that largely fuels the current research efforts in micromachine technology with promises of applications unseen due to limitations in current technology.

Nature's decentralized approach to achieving collective tasks results in feasible versus optimal solutions. In fact, admitting some randomness occurs at the individual level is felt by some researchers as part of the society's functioning [29]. Oster and Wilson [28] have suggested that social insects can well afford behavioral variance. This variance, they claim, could increase the probability that the collective activity will eventually be performed, with their collective reliability more than compensating for the individual inefficiency. The usefulness of these conjectures in controlling robot populations will only become evident as larger systems of multiple robots are simulated and built.

Although in principle communication among robots should improve their ability to cooperate, as a system grows in number, noncommunicating systems should scale more easily. This essentially amounts to a tradeoff between local and global sensing strategies, but may just result in a degradation in response time to external stimulus.

The key to controlling teams of robots using group behaviors lies in the mechanisms with which they are invoked. By combining the mechanisms, collective tasks may be created and accomplished by robots with more than one group behavior. Like executing the steps of a program on a synchronized distributed computer system, group behaviors form a nonconnected link between robots coordinating their activity in a cooperative manner.

In our animat approach to building intelligent systems, the study of social insects plays an important role in guiding our selection of control strategies for our multiple robot systems. Although the strategies proposed in this paper are not a comprehensive set, they do represent the approach we have taken and are intended as examples of our initial exploration into collective robotics. What is still missing in our approach is a mathematics on which to base our models, of both robots and the tasks they are designed to accomplish. Lacking this formal theory, we have taken the approach of analyzing specific tasks, couched in terms of their sensory requirements, in the hope that the more salient features will generalize across specific task domains.

The discovery, by social biologists, of the various stimulus-cues in collective tasks motivates our mechanisms in controlling tasks by multiple robots. A common theory that adequately explains the cooperative behavior of social insects is still missing in the field of behavioral biology. However, the many well researched examples of collective task-achieving behavior do provide a starting point from which to build systems in collective robotics. Whether these systems will show to be scalable to the point of accomplishing useful tasks remains yet to be proven; however, Nature has already provided an existence proof in social insects demonstrating its feasibility.

References

[1] Arkin, R.C. (1992). Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351-364.

[2] Arkin, R.C., Balch, T. & Nitz, E. (1993). Communication of behavioral state in multi-agent retrieval tasks. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Vol. 3, 588-594.

[3] W. Armstrong, A. Dwelly, J. Liang, D. Lin, and S. Reynolds. Learning and generalization in adaptive logic networks, in artificial neural networks. In *Proceedings of the 1991 International Conference on Artificial Neural Networks*, pages 1173-1176, 1991.

[4] Hajime Asama, Maki K. Habib, Isao Endo, Koichi Ozaki, Akihiro Matsumoto, and Yoshiki Ishida. Functional distribution among multiple mobile robots in an autonomous and decentralized robot system. In *1991 IEEE International Conference on Robotics and Automation*, pages 1921-1926, 1991.

[5] Gerardo Beni and Jing Wang. Theoretical problems for the realization of distributed robotic systems. In *1991 IEEE International Conference on Robotics and Automation*, pages 1914-1920, 1991.

[6] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, 1984.

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

[8] Rodney A. Brooks. Challenges for complete creature architectures. In *First International Conference on Simulation of Adaptive Behavior*, pages 434-443. MIT Press, 1990.

[9] Rodney A. Brooks and Anita M. Flynn. Fast, cheap and out of control. AI Memo 1182, MIT, 1989.

[10] Rodney A. Brooks, Pattie Maes, Maja J. Mataric, and Grinell More. Lunar base construction robots. In *IEEE International Workshop on Intelligent Robots and Systems IROS '90*, pages 389-392, 1990.

[11] Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *IEEE International Workshop on Intelligent Robots and Systems IROS '90*, pages 67-72, 1990.

[12] P. Dario, F. Ribechini, V. Genovese, and G. Sandini. Instinctive behaviors and personalities in societies of cellular robots. In *1991 IEEE International Conference on Robotics and Automation*, pages 1927-1932, 1991.

[13] J-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting robot-like ants and ant-like robots. In *First International Conference on Simulation of Adaptive Behavior*, pages 356-363. MIT Press, 1990.

[14] J-L. Deneubourg and S. Goss. Collective patterns and decision making. *Ethology Ecology & Evolution*, 1:295-311, 1989.

[15] Keith L. Doty and Ronald E. Van Aken. Swarm robot materials handling paradigm for a manufacturing workcell. In *1993 IEEE International Conference on Robotics and Automation*, Vol. 1, pages 778-782, 1993.

[16] Nigel R. Franks. Teams in social insects: Group retrieval of prey by army ants. *Behavioral Ecology and Sociobiology*, 18:425-429, 1986.

[17] Nigel R. Franks. Army ants: A collective intelligence. *American Scientist*, 77:139-145, 1989.

[18] V. Genovese, P. Dario, R. Magni, and L. Odetti. Self organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of pollutants. In *1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1575-1582, 1992.

[19] S. Goss and J-L. Deneubourg. Harvesting by a group of robots. In *Proceedings of the First European Conference on Artificial Life*, pages 1-10, 1991.

[20] W. Hangartner. Carbon dioxide, a releaser for digging behavior in solenopis geminata. *Psyche*, 76:58-67, 1969.

[21] C. Ronald Kube and Hong Zhang. Collective robotic intelligence. In J. A. Meyer, H. Roitblat, and S. Wilson, editors, *Second International Conference on Simulation of Adaptive Behavior*, pages 460-468. MIT Press, 1992.

[22] C. Ronald Kube, Hong Zhang and Xiaohuan Wang. Controlling collective tasks with an ALN. In *1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 289-293, 1993.

[23] M. Lindauer. Ein beitrag zur frage der arbeitsteilung im bienenstaat. *Zeitschrift für Vergleichende Physiologie*, 34(4):299-345, 1952.

[24] M. Lindauer and H. Martin. Über die orientierung der biene im duftfeld. *Naturwissenschaften*, 50(15):509-514, 1963.

[25] Maja J. Mataric. Minimizing complexity in controlling a mobile robot population. In *1992 IEEE International Conference on Robotics and Automation*, pages 830-835, 1992.

[26] J. C. Moser. Pheromones of social insects. In D. Wood, R. Silverstein, and M. Nakajima, editors, *Control of Insect Behavior by Natural Products*, pages 161-178. Academic Press, 1970.

[27] Fabrice R. Noreils. Multi-robot coordination for battlefield strategies. In *1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1777-1784, 1992.

[28] G. F. Oster and E. O. Wilson. *Caste and Ecology in Social Insects*, Princeton University Press, 1978.

[29] J. M. Pasteels, J. Deneubourg, and S. Goss. Self-organization mechanisms in ant societies: Trail recruitment to newly discovered food sources. In J. M. Pasteels and J. Deneubourg, editors, *From Individual to Collective Behavior in Social Insects*, pages 155-175, 1987.

[30] RIKEN. *International Symposium on Distributed Autonomous Robotic Systems*, Saitama, Japan, September 1992.

[31] Thomas D. Seeley, Scott Camazine, and James Sneyd. Collective decision-making in honey bees: How colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28:277-290, 1991.

[32] Karansher Singh and Kikuo Fujimura. Map making by cooperating mobile robots. In *1993 IEEE International Conference on Robotics and Automation*, Vol. 2, pages 254-259, 1993.

[33] Luc Steels. Cooperation between distributed agents through self-organisation. In Yves Demazeau and Jean-Pierre Muller, editors, *Decentralized A. I.*, pages 175-196, Amsterdam, North-Holland, 1990.

[34] Daniel J. Stilwell and John S. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *1993 IEEE International Conference on Robotics and Automation*, Vol. 1, pages 766-771, 1993.

[35] A. M. Stuart. Alarm, defense, and construction behavior relationships in termites (*isoptera*). *Science*, 156:1123-1125, 1967.

[36] A. M. Stuart. Social behavior and communication. *Biology of Termites*, 1:193-232, 1969.

[37] John Sudd. How insects work in groups. *Discovery*, pages 15-19, June 1963.

[38] Tapio Taipale and Hirai Shigeoki. A comparative study of multi-robot systems. *Bulletin of the Electrotechnical Laboratory*, 56(8):892-922, 1992.

[39] Yoshitaka Tatsue. Worldwide research and development of micromachines. Technical Report, Micromachine Center, 1993.

[40] G. Theraulaz, S. Goss, J. Gervet, and J-L. Deneubourg. Task differentiation in polistes wasp colonies: A model for self-organizing groups of robots. In *First International Conference on Simulation of Adaptive Behavior*, pages 346-355. MIT Press, 1990.

[41] P. K. C. Wang. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177-195, 1991.

[42] Rüdiger Wehner. Matched filters - neural models of the external world. *Journal of Comparative Physiology A*, 161:511-531, 1987.

[43] E. O. Wilson. *The Insect Societies*, The Belkap Press of Harvard University Press, 1971.

[44] E. O. Wilson, N. I. Durlach, and L. M. Roth. Chemical releasers of necrophoric behavior in ants. *Psyche*, 65(4):108-114, 1958.

[45] E. O. Wilson and B. Hölldobler. *The Ants*, The Belkap Press of Harvard University Press, 1990.

[46] Stewart W. Wilson. The animat path to AI. In *First International Conference on Simulation of Adaptive Behavior*, pages 15-21. MIT Press, 1990.

[47] S. Yuta and S. Premvuti. Consideration on cooperation of multiple autonomous mobile robots. In *IEEE International Workshop on Intelligent Robots and Systems*, pages 545-549, 1991.

[48] J. Zeil, G. Nalbach, and H. O. Nalbach. Eyes, eye stalks and the visual world of semi-terrestial crabs. *Journal of Comparative Physiology A*, 159:801-811, 1986.