

Controlling Collective Tasks With An ALN

C. Ronald Kube, Hong Zhang and Xiaohuan Wang

Department of Computing Science

University of Alberta

Edmonton, Alberta CANADA T6G 2J9

kube@cs.ualberta.ca zhang@cs.ualberta.ca xiaohuan@cs.ualberta.ca

April 29, 1993

Abstract

In this paper, we explore the idea of using an Adaptive Logic Network (ALN) [2] for behaviour arbitration. Our approach is to define the collective task, to be performed by multiple robots, as a group behaviour. The group behaviour is a set of behaviours, each of which specifies a single step in the collective task. An environmental cue can be used to control the transition between behaviours, thus allowing the progress of the collective task to self-govern its execution. We simplify the behaviour arbitration in the collective task by training an ALN implemented using simple combinational logic. We provide a description of our Collective Robotic Intelligence Project (CRIP) including our simulation results and our multi-robot system on which these results will be deployed.

1 Introduction

Can collective tasks be accomplished using group behaviours? Interest in accomplishing tasks by using multiple robots has resulted in systems designed using cooperative behaviour [11, 8, 1, 7, 17]. Our previous work [10] demonstrated that computationally simple control mechanisms allowed multiple autonomous robots to perform simple tasks without centralized control or use of explicit communication. In this paper, we present a new method of behaviour arbitration, which we have characterized as a pattern classification problem used to train an Adaptive Logic Network. We also explore the idea of using *environmental cues* for controlling task achieving group behaviour.

Collective tasks are a result of a group of task achieving robots all with a common purpose. For example, a group of robots designed to keep a table top free from objects, will collectively locate and push to an edge any object placed on the table. When the object is heavier than a single robot can move, the cooperative effort of the

group is required. Similarly, consider the task of a group of fire fighting robots. A fire that quickly spreads is easier to bring under control if a system of multiple robots can physically distribute itself over the area. Collective tasks of these forms are suitable to the multiple robot approach.

Research projects are now beginning to investigate the cooperative behaviour of multiple robot systems necessary for collective tasks. Such tasks include simple retrieval tasks, flocking, and cooperative pushing [1, 11, 3, 9].

Our own work has examined the problem of controlling multiple autonomous robots. Taking an Animat approach, we proposed some simple mechanisms which invoked group behaviour based on observations made from the study of social insects. The mechanisms were then tested in simulation. Some of these mechanisms have been tested on a group of five homogeneous mobile robots [10]. Specifically, we demonstrated that a common task and a simple form of cooperation, implemented as robot-avoidance, could be used to control a group of multiple robots as they executed a simple collective task.

Combining and merging the outputs of behaviour modules is a daunting task for any more than a few behaviours. A fixed priority scheme between behaviours works well when the number of behaviours is small or their priority obvious. However, as the number of behaviours increases, it becomes less clear how behaviour arbitration should be resolved. The approach we are currently exploring is to formulate behaviour arbitration as a pattern classification problem on which an ALN can be trained.

The remainder of this paper is organized as follows. In section 2 we discuss collective tasks which are suitable for multiple robot systems. In section 3 we examine how a group behaviour can be constructed from a set of individual task achieving behaviours, and how environmental cues can be used to activate each behaviour. In section 4 we describe the fundamentals of ALN and present an approach we are investigating to simplify the design of behaviour arbitration techniques. In section 5 we describe

briefly our Collective Robotic Intelligence Project (CRIP) and both the simulation results and implementation of our multiple mobile robot system. Finally, section 6 summarizes our work to date.

2 Collective Tasks

Nature supplies us with numerous examples of collective task achieving behaviour. Ants, bees and termites all achieve tasks collectively with, seemingly simple, insect intelligence. Several approaches to collective tasks inspired by observations of social insects have been proposed [6, 14, 15, 1]. We have also been examining examples of collective task achieving behaviour by social insects [9] in the hope that we may gain some insight into the mechanisms that govern these fascinating examples of collective behaviour provided by Nature.

We have considered several specific examples of collective behaviour. Wilson and Holldobler described nest construction by weaver ants [16]. Nest walls are constructed by folding leaves, a task requiring the cooperative effort of several ants. After having spread over the leaf's surface they randomly pull any graspable edge. The tip is often more easily turned than an edge, and the initial success causes other ants to aid the effort abandoning their own. It is not clear which stimulus causes the ant to switch from an individual to group effort, although it is hypothesized that a visual cue is involved.

A second example uses a stimulus in the environment to invoke a collective behaviour. These *environmental cues* simultaneously invoke the same behaviour in the group of insects within range of the stimulus. For example, the amount of time spent by a bee on food collecting is governed by light intensity [13]. Ants also begin their daily foraging for food upon the detection of the light of dawn [12]. In these cases, cues in the environment serve to activate the common collective behaviour.

The field of behavioural biology is still searching for a theory that explains the collective behaviour of social insects. However, the many well researched observations can serve as examples for controlling multiple autonomous mobile robots.

Solving tasks with the use of multiple robot systems has advantages researchers are just beginning to explore. For example, tasks with an inherent parallel nature, such as search, can be accomplished in a shorter time frame using multiple searching robots. The multiple robot approach also serves to increase the redundancy and distribute the risk of single failure; an important feature of any system working in hazardous environments.

Distributing a task over multiple robots is not a simple matter of dividing the time necessary for task completion by the number of robots. Issues involving cooperation and

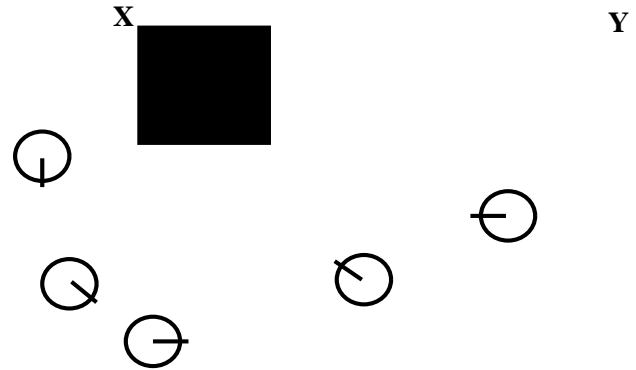


Figure 1: The robots (circles) must locate and collectively push the box from position X to position Y.

task progression must also be addressed when designing a collective task suitable for execution by a multiple robot system.

Cooperation comes into play when the collective tasks involve several robots working together towards some common goal. In these situations some mechanism must exist which both allows for the cooperation to take place and to regulate the progress of the cooperative task. In the next section we discuss one possible approach using group behaviours.

3 Group Behaviour

Collective tasks to be performed by multiple robots can be defined using group behaviours. A group behaviour is a set of behaviours, each specifying a single step in the collective task. An *environmental cue* is used to control the transition between behaviours. This allows the progress of the collective task to self-govern its execution. We simplify the behaviour arbitration of each individual step in the collective task by training an Adaptive Logic Network (ALN)[2] implemented using simple combinational logic.

Group behaviour can be defined as a set of behaviours each of which are activated by a specific sensor pattern. For example, consider the simple task of pushing a box depicted in Figure 1. The objective is to move the box from its initial position at X to a position designated as Y. The task can be specified with the following four behaviours.

- *FIND* A default behaviour that keeps the robot in motion.
- *SLOW* A speed reduction behaviour.
- *GOAL* A goal seeking behaviour.
- *AVOID* A collision avoidance behaviour.

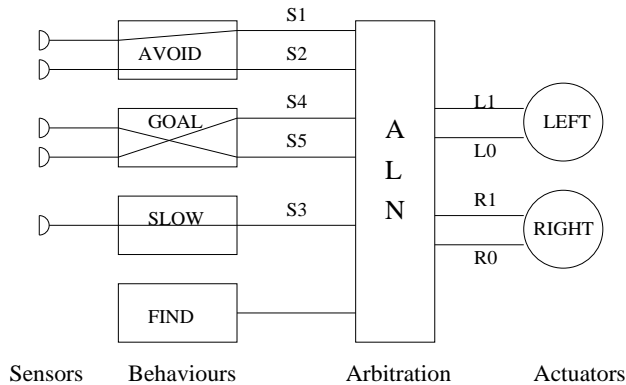


Figure 2: The model for the *push-box* group behaviour. Behaviour modules process sensor data to produce actuator commands. The commands are then arbitrated by an ALN trained to recognize a pattern on its input.

The model of this group behaviour is depicted in Figure 2. Behaviour modules process sensor data to produce actuator commands. This processing may simply consist of crossed or straight connections, as in Braitenberg's vehicles [4], sensor thresholding or the altering of a sensor's view angle. For example, in the robots we constructed (described in the sequel) for the push-box task, two infrared sensors provided left and right obstacle detection for the AVOID behaviour. The behaviour thresholds the input and uses a simple straight connection to its output. The result is activation of the right sensor produces a right motor output command causing the robot to move left. The GOAL behaviour is constructed in a similar manner using crossed connections and sensors designed to locate a brightly lit box. Individual behaviours may also process the sensor data differently depending on the state of the robot. An example would be a FOLLOW behaviour, designed to locate and follow another robot, that initially considers sensor data with a field-of-view of 180° while searching, only to narrow the field-of-view to 45° during following.

Behaviour modules with their actuator commands compete for control of actuator resources by each issuing, possibly conflicting, commands. For example, the GOAL behaviour may wish to direct the robot to the left while the AVOID behaviour, having detected an obstacle on the left, may wish to steer the robot right. In a fixed priority system behaviour arbitration is straight forward with the highest priority active behaviour taking control of the robot.

In the push-box model, actuators are left and right wheel motors each with two control bits allowing for stop, half-speed and full-speed commands. Table 1 is a partial truth table for the left and right wheel motors. The five sensor outputs (processed by the behaviour modules) la-

Dec	S_1	S_2	S_3	S_4	S_5	L_1	L_0	R_1	R_0
0	0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0	0	0
2	0	0	0	1	0	0	0	1	0
...			
20	1	0	1	0	0	0	1	0	0
...			
31	1	1	1	1	1	0	0	0	1

Table 1: Truth table for sensor outputs and motor commands.

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

Table 2: Motor commands; where $M = \{L,R\}$

belled S_1 to S_5 in Table 1 correspond to the four motor control bits labelled L_0, L_1 for the left motor and R_0, R_1 for the right motor, resulting in a 32 entry truth table that maps sensor outputs to motor commands (shown in table 2). The labels S_1 to S_5 correspond to *avoid-left*, *avoid-right*, *slow*, *goal-left* and *goal-right* sensor outputs. For example, the entry in row 20 of table 1 has $S_1 = \text{avoid-left ON}$; resulting in a *move-right* motor command which is implemented by turning the *left* motor on.

As the complexity of collective tasks increase, so does the problem of arbitration of behaviours each of the multiple robot possess. We propose to solve this problem by characterizing the behaviour output pattern as a mapping problem between sensor outputs and actuator commands, which an ALN can be trained to recognize. We then manually guide the robots through the task to be performed, training the ALN in the process. The result is an ALN that serves as a behaviour arbitration mechanism. In the next section we describe ALNs and our simulation approach.

4 Simulation Approach

Our robot population simulator, *SimbotCity*, described in [9] allows us to model a robot as a collection of sensor systems and actuator resources as well as a set of behaviour modules that map sensor inputs to actuator outputs. A user interface shown in Figure 3 allows the simulation to

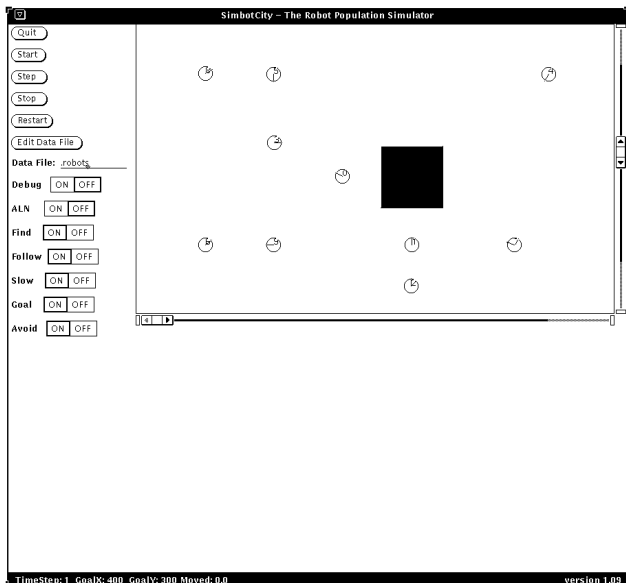


Figure 3: Initial robot configuration. Robots (the circles) must locate and collectively push the black box.

be run continuously or in single step mode. Robots are represented by numbered circles with a pointer indicating their direction of travel. Each robot is modelled with two obstacle avoidance sensors, two goal sensors and one collision sensor used to slow the robot's motion. The actuators are two wheel motors which control both the direction and speed of motion. A configuration file specifies the initial number, location and heading of each robot. Individual behaviours can be switched on or off during the simulation by the user. As well, behaviour arbitration using either ALN or a fixed priority scheme can be selected. A debug button allows the user to view sensor patterns of each robot individually while the simulation is in progress.

Behaviour arbitration is the process of deciding how individual behaviours within the robot interact. Within our robots behaviour arbitration is handled with a modified fixed priority *subsumption* architecture [5]. This approach worked well when a small number of behaviours are used to implement the robot's control architecture. As previously mentioned, as we began to add additional behaviours to accomplish more difficult tasks it became increasingly difficult to decide how the behaviours should be arbitrated. While viewing the execution of a collective task, as an observer, it was easy to decide what the group should do at any given moment, but hard to specify the behaviour arbitration required.

Some previous experience with ALNs suggested that it should be possible to characterize the behaviour arbitration problem, as a pattern classification problem on which the ALNs can be trained.

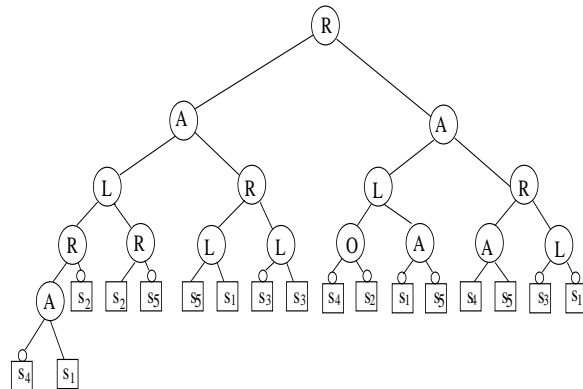


Figure 4: A Trained ALN with 17 Leaves

4.1 Adaptive Logic Networks

ALNs are a type of neural network constructed using binary trees. Each node in the tree is initially assigned a boolean function from the set { AND, OR, LEFT or RIGHT }. At the base of the tree are the leaves to which the input vector is presented and the root of the tree represents one output bit. A tree begins with a random node assignment and is trained, resulting in an assignment of the correct boolean function to non-leaf nodes using a simple training algorithm explained in [2]. Once trained, the ALNs classify new input vectors into one of the classes established during training.

ALNs offer speed and implementation advantages important in our approach. Both can be attributed to the boolean nature of each node in an ALN binary tree. For example, a node whose assigned boolean function is AND is both quick to evaluate if one input is zero and easy to implement in VLSI circuit technology. In the next section we briefly discuss some of the results from the use of ALNs in our Collective Robotic Intelligence Project (CRIP).

4.2 ALN Based Controller

A typical ALN tree after training is shown in Figure 4 for the push a box example. This is a 17-leaf tree whose inputs are the sensor readings (processed by behaviour modules) and whose output is one bit of the motor command. Circles and squares represent the logical operators and input leaves, respectively. Letters A, O, L, and R, represent logic AND, OR, LEFT, and RIGHT functions. The small circle on top of a leaf represents the fact that the compliment of the sensor reading is used. In this example, there are five sensor readings involved and they are labelled S_1 , S_2 , S_3 , S_4 and S_5 , respectively.

The training of the ALNs is in general supervised and conducted through a higher level process which instructs the ALNs if they are providing the correct responses in a

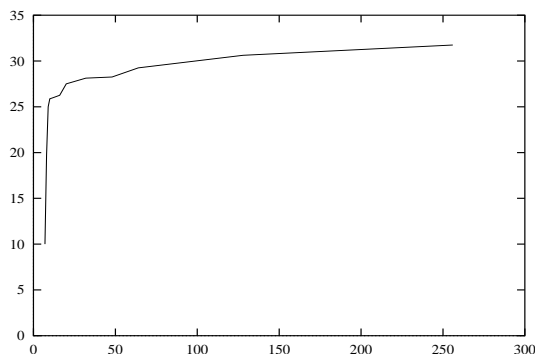


Figure 5: Relationship between ALN Leaf Number and Its Performance

given situation. This can be done, for example, through a human instructor. Although the detailed relationship between the sensor inputs and actuator response is not known a priori, the human instructor can nonetheless determine if a robot is behaving properly in a given circumstance. This information can be conveyed to the robots involved in the task and ALNs can be modified accordingly to improve the performance incrementally. In our case, in order to illustrate the applicability of the ALNs to collective behavior learning, we used as the instructor the robot simulator that has been preprogrammed properly to train another simulator which starts off with an ALN-based controller in a randomly connected state.

For the push-a-box task, Figure 5 illustrates the relationship between the number of leaves chosen for each of the four trees (x-axis) and the number of correctly computed outputs by the four trees (y-axis). Since there are five input bits, there are a total of 32 possible outputs to be learned by the four ALNs. As can be seen, the performance of the ALN increases roughly exponentially with the number of leaves. When the number of leaves becomes sufficiently large, a perfect performance is possible. For each leaf number considered, several experiments were run and the average of the numbers of the correctly computed outputs was used to produce the figure. This relationship is important in the design of an ALN-based controller, in order to choose an appropriate leaf number for a given task, when the correct outputs are available only in the form of high-level observations.

Figure 6 shows how the ALN's quality affects the performance of a task, using the push-a-box task as the example. In this figure, the x-axis represents the competence of an ALN-based controller expressed in terms of the percentage of correctly learned outputs, and y-axis represents the amount of simulation steps it takes to move the box 100 units. It can be seen that a minimum of near 80% compe-

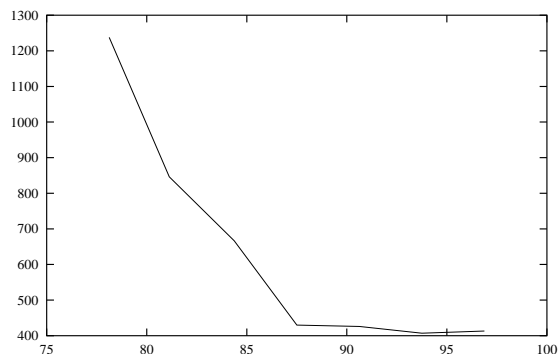


Figure 6: Execution Time as a Function of ALN Competence

tence level is required to accomplish the task at all, and the execution time approaches a lower bound when the competence of an ALN controller approaches 100%. Similar to Figure 5, the performance of the ALN controller for each competence level is studied by averaging the execution times of several different ALN controllers that have the same competence level.

5 Experimental Robots

Our previous work has been looking for suitable control mechanisms with which to control multiple robot systems without using a centralized supervisory approach [9, 10]. The collective task of pushing a box (see Figure 7) was accomplished using one common group behaviour. Implementation resulted in the construction of 5 simple robots controlled by combinational logic.

More difficult tasks will require several group behaviours with the added burden of behaviour arbitration. Motivated by the results of the previous section, we will extend the collective task to formation marching requiring a group behaviour to keep the robots together. Such a group behaviour will be useful for the previously mentioned fire fighting or search collective tasks.

6 Summary

In this paper we have presented our approach towards the control of a group of multiple robots and their progress towards executing a collective task. We have explored the use of Adaptive Logic Networks as a behaviour arbitration mechanism by characterizing the robot's behaviour output patterns as a classification problem. The simulation results have demonstrated the soundness of the approach; we will then implement the results in combinational logic

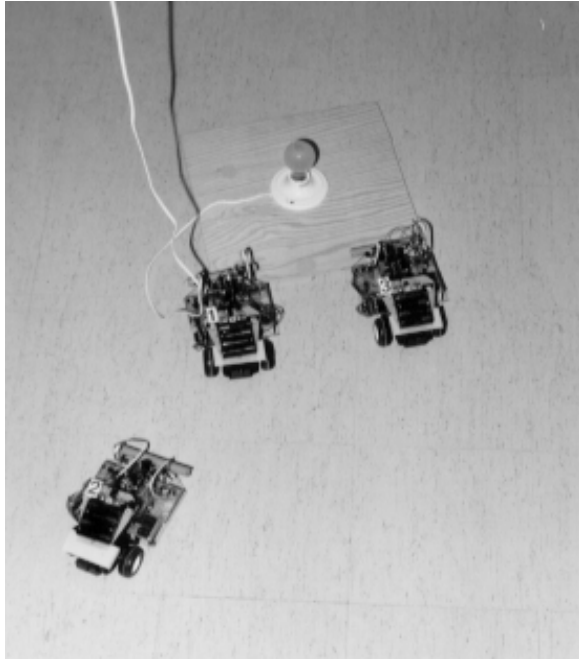


Figure 7: Robots, controlled with simple combinational logic, shown collectively pushing a brightly lit object forward. The object is too heavy for one robot to move, and requires the collective effort of at least two robots.

on our system of five physical robots and continue our exploration of cooperative robotic behaviour.

We consider ALNs to be naturally suited for the task of controlling behaviour-based robots in which both the sensor inputs and actuator outputs either are binary or can be encoded with a few bits. The learning ability of ALNs allows the robots to gain performance incrementally and reduces the difficulty of the controller design process. Although conventional neural networks can be used as well, they are more suited to process continuously variables. In addition, ALNs have been shown to be safer than conventional neural networks due to its ability to interpolate predictably. Our future includes training of the ALNs to perform collective tasks for which no known controllers exist such as marching in formation. To achieve a reasonable training time, an appropriate user interface and training algorithms will be designed.

References

- [1] Ronald C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.
- [2] 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.
- [3] 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.
- [4] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1984.
- [5] 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.
- [6] 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.
- [7] Toshio Fukuda, Tsuyoshi Ueyama, and Fumihito Arai. Control strategy for a network of cellular robots. In *1991 IEEE International Conference on Robotics and Automation*, pages 1616–1621, 1991.
- [8] Maki K. Habib, Hajime Asama, Yoshiki Ishida, Akihiro Matsumoto, and Isao Endo. Simulation environment for an autonomous and decentralized multi-agent robotic system. In *1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1550–1557, 1992.
- [9] C. Ronald Kube. Collective robotic intelligence: A control theory for robot populations. Master's thesis, University of Alberta, 1992.
- [10] C. Ronald Kube and Hong Zhang. Collective robotic intelligence. In *Second International Conference on Simulation of Adaptive Behavior*, pages 460–468, December 7-11 1992.
- [11] Maja J. Mataric. Minimizing complexity in controlling a mobile robot population. In *1992 IEEE International Conference on Robotics and Automation*, pages 830–835, 1992.
- [12] T.C. Schneirla. The army ants. In *Report of The Smithsonian Institution for 1955*, pages 379–406, 1955.
- [13] B. Schricker. Die orientierung der honigbiene in der dammerung zugleich ein beitrag zur frage der ocellenfunktion bei bienen. *Zeitschrift fur Vergleichende Physiologie*, 49:420–458, 1965.
- [14] Luc Steels. Cooperation between distributed agents through self-organisation. In Yves Demazeau and Jean-Pierre Muller, editors, *Decentralized A.I.*, pages 175–196, Amsterdam, 1990. North-Holland.
- [15] 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.

- [16] E.O. Wilson and B. Holldobler. *The Ants*. The Belkap Press of Harvard University Press, 1990.
- [17] 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.