

Organizational Characteristics for Multi-Agent Robotic Systems

G. Dudek[§], M. Jenkin[†], E. Milios[‡] and D. Wilkes[‡]

[§]Research Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada

[†]Department of Computer Science, York University, North York, Ontario, Canada

[‡]Department of Computer Science, University of Toronto, Toronto, Ontario, Canada

Abstract

In many cases several mobile robots (autonomous agents) can be used together to accomplish tasks that would be either more difficult or impossible for a single mobile robot. In this paper we present a taxonomy of the different ways in which such a collection of autonomous robotic agents can be structured. This provides a coherent framework for comparing different robot collectives and for creating or modelling such collectives. We show that certain swarms provide little or no advantage over having a single robot, while other swarms can obtain better than linear speedup over a single robot. There exist both trivial and non-trivial problems for which a swarm of robots can succeed where a single robot will fail.

1 Introduction

Although most mobile robotic systems involve a single robot operating in an environment, there are no conceptual reasons why an environment could not be inhabited by a number of robots. For some specific robotic tasks, such as exploring an unknown planet for example, it has been suggested that rather than sending one very complex robot to perform the task it would be better to send a large number of smaller, simpler robots. Such a collection of robots is known as a *swarm*[2]. Using multiple robots rather than a single robot can have several advantages and leads to a variety of design tradeoffs. In particular, large numbers of simple robots may be simpler in terms of individual physical design and thus the ensuing system can be more economical, more scalable and less sensitive to overall failure. Likewise, destruction of a single member of a large swarm may not be catastrophic while the failure of a single subsystem of a conventional robot is usually disastrous. The problem is, of course, that the swarm of robots must have a collective behaviour or set of actions that accomplishes the same behaviour or action that the single more complex robot was required to do. For the swarm to have an intelligent behaviour, the members of the swarm must be able to communicate with each other. This communication can take place directly via some communication channel or indirectly through one robot sensing a change in other robots in its environment.

*The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council.

The behaviour based control strategy put forward by Brooks [5] is quite well known and it has been applied to a collection of simple independent robots, usually for simple tasks. Several authors have considered how a collection of simple robots can be used to solve complex problems. Ueyama *et. al.*[13] propose that groups of highly complex robots should, in principle, be organized in tree-like hierarchies. Hackwood and Beni[8] propose a model in which the robots are particularly simple but act under the influence of signposts robots.

Mataric[11] describes experiments with a homogeneous population of actual robots acting under different communication constraints. The robots either act in ignorance of each other, informed by each other, or intelligently with each other. As inter-robot communication improves, more and more complex behaviours are possible. In the limit, in which all of the robots have complete communication, then the robots can be considered as appendages of a single larger robot. One major goal of a robotic swarm is to distribute not only the sensing (and possibly actions) of the robots, but also the intelligence. What sort of processing can be accomplished by a collection of robots that cannot be accomplished by a single one? What effects do limits on communications and unit processing capabilities have on the potential actions of the swarm?

The information processing ability of the swarm is dependent upon a large number of factors including the number of units, their sensing abilities, their communication mechanisms, etc.[1,12]. In order to more fully understand the properties of various designs of swarms, it is instructive to group swarms into classes and to determine the processing ability of each class. It may be the case that certain swarm organizations have more potential processing ability than others, and that some swarm organizations may be similar to existing parallel models of computation.

In order to more fully understand the capabilities of robot swarms we propose the following taxonomy. This taxonomy is based upon a number of orthogonal axes. After defining the axes, we relate some points in the taxonomy to existing parallel computational models and robotic systems and illustrate some problems which give some indication of the different ways in which swarms can operate and the different problems that they can solve.

2 Swarm taxonomy

Given the variety of possible designs of groups of mobile robots, we suggest that it is useful to organize these concepts along taxonomic axes. The objective here is to both clarify the strength, constraints and tradeoffs of various designs, and also to highlight various design alternatives. We suggest that there are several natural dimensions along which robotic swarms can be classified. These dimensions address the characteristics of the swarm as a whole rather than the architectural characteristics of individual robots. These dimensions follow with key sample points from each dimension noted.

By swarm size: The number of robots in the environment.

- 1 robot. The swarmless swarm case.
- 2 robots. The simplest group; communication is trivial.
- Multiple robots. The number n is small relative to the size of the task or environment.
- $n \gg 1$ robots. There is effectively an infinite number of robots.

By communication range:

- Robots cannot communicate with other robots directly.
It is possible for robots to communicate with each other *indirectly* by observing their presence, absence or behavior (as many animals seem to). In order to have "ignorant robots" [11], the robots must not only not communicate with each other they must not try to signal each other through behaviour. For robots to be able to navigate they must be able to sense in at least a limited manner, and this suggests that they should be able to sense other robots. Sensing implies the possibility of communication. It may not be realistic to have truly ignorant robots.
- Robots can only communicate with other robots which are sufficiently nearby.
This corresponds to the communication mechanism proposed in [8]. Distance, in this context, can be interpreted either topologically (e.g. nearest neighbour) or in a Euclidean sense (neighbours within some range). Limited communication distance can occur due to physical communication constraints (the power of the communication signal). It may also be the case that the robots do not communicate through a separate communication channel, but rather communicate by performing some overt action (blinking lights, spinning, etc.), which can only be sensed by other robots in the local neighbourhood.
- Robots can communicate with any other robot.
A classical assumption which is probably impractical in practice if $n \gg 1$.

Note that we have deferred issues related to having multiple robots (autonomous agents) communicating (writing) to a single robot (memory location). This is a classic problem in parallel computation models [7]. As minor modifications in the communication design of parallel machines can result in major changes in the power of the resulting machine [3], we also partition the taxonomy by considering the topology of the inter-unit communication strategy utilized by the swarm.

By communication topology: Robots may not be able to communicate with an arbitrary element of the swarm regardless of its proximity. Robots may only be allowed to communicate within a particular hierarchy [13], or with specific controller robots [8]. Individual robots may have names and messages may be sent to them directly, or messages may be broadcast to all robots. Many possibilities exist. These include

- Broadcast. Every robot can communicate with all of the other robots. It is not possible to send a message to a particular swarm element.
- Address. Every robot can communicate with any other robot by name or address.
- Hierarchy. Robots are arranged in a particular hierarchy and may only communicate through this hierarchy.

By communication bandwidth: Communication may be cheap in terms of the robots' processing time, in that the robot has a special channel for communication, or it may be expensive in that the robot is prevented from doing other work.

- Communication is free. The communication bandwidth is sufficiently high that the communication cost and overhead can be ignored. This is a common assumption in theoretical computational models and can lead to robots that behave as if there was a central intelligence.
- Communication costs are on the close order of the cost of moving the robot from one location to another. This can be thought of as being similar to the mechanism by which bees communicate by performing an intricate dance that is observed by other bees in the neighbourhood.
- Very high cost. Communication costs much more than the cost of moving from one location to another. This suggests very independent robots.
- No communication. Robots are unable to sense each other. As mentioned earlier, this is probably an impractical case.

Swarm reconfigurability: The rate at which the swarm can spatially re-organize itself; roughly equivalent to the rate at which members can move with respect to one another. For example, bees can presumably reconfigure their spatial layout with respect to one another very quickly while soldiers marching in lock-step or cars on a highway cannot. This dimension

is closely related to the communication range of members of the swarm. If the swarm changes its topology, however, then changes in arrangement are not isomorphic to scaling of the communication range. In practice, there may be topological constraints to the allowed reconfigurations (for example, if the members of a robotic swarm drive on roads, then only certain topological changes are allowed irrespective of member velocity).

- Static arrangement. The topology is fixed.
- Coordinated rearrangement. Re-arrangement with members that communicate.
- Dynamic arrangement. The relationship of members of the swarm can change arbitrarily.

By swarm unit processing ability: Each unit of the swarm has a particular model of computation. For simplicity, we will deal only with the common simple sequential computational models. Note that this is a non-continuous dimension.

- Non-linear summation unit[9]. This very simple unit is used in constructing a simulated neural network but may be too simple to be a realistic model for a single robot although it illustrates the near-extremum of this dimension.
- Finite state automaton. This is the computational model preferred by the subsumption architecture computational systems[4].
- Push-down automaton.
- Turing machine equivalent. The computational model assumed by most robotic systems.

By swarm composition: A swarm may be **homogeneous** (made up of units all with the same characteristics) or **heterogeneous**. Even an ensemble of robots that is homogeneous in terms of physical structure may be differentiated by programming or behaviour.

We illustrate the classification of multi-agent systems using some of these dimensions in Figure 1 (since few effective autonomous robotic multi-agent systems exist, we have included illustrative examples of other types of collectives).

3 Some results and cautionary tales

Swarm robots are a distributed intelligence. Distributed computer processing has been extensively studied by both theoretical computer scientists and mathematicians, as well as by computer designers. Many of the models of swarm robots map almost directly onto pre-existing computational or hardware models. Interesting but highly theoretical results concerning the performance of a particular swarm may be easily obtained but it is important to recognize that PRAM (parallel random access machines) models are highly developed[14]. Nevertheless, due to the mobile

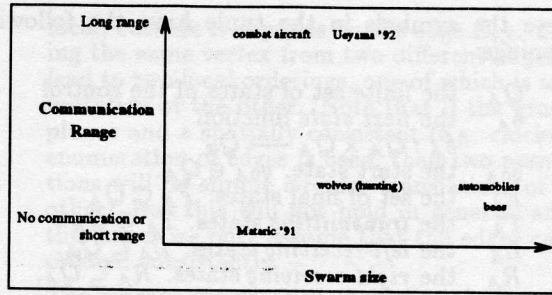


Figure 1: Some multi-agent systems classified

nature of the processors, new and interesting results are possible.

The following two examples illustrate how sufficiently sophisticated swarms can be used to solve particular problems and relates these swarms to the taxonomy given above. Even swarms of simple units can perform sophisticated computations and the performance of swarms can be provably better than that of single robots for certain tasks such as exploration.

3.1 Turing Equivalence

We will show that an unbounded number of finite automata robots with the ability to communicate their state to their neighbours may simulate an arbitrary Turing Machine. This is notable, because this fact makes it possible in principle to construct a spatially distributed intelligence from a large collection of very simple devices. We envisage that the individual automata may be mobile (moving according to their own current state, as assigned by the distributed computation), and thus able to accomplish some interesting actions in the world.

For the purposes of the exposition, we define our automata and Turing machine using the conventions and notation of [10]. Let M be an arbitrary Turing machine, given by

$$M = (Q_M, \Sigma_M, \Gamma_M, \delta_M, q_{0M}, B_M, F_M) \quad (1)$$

where the symbols in the tuple have the following meanings:

- Q_M the finite set of states of the control
- Σ_M the finite set of input tape symbols ($\Sigma_M \subset \Gamma_M, B_M \notin \Sigma_M$)
- Γ_M the finite set of tape symbols
- δ_M the *next move* function
- $\delta_M : Q_M \times \Gamma_M \longrightarrow Q_M \times \Gamma_M \times \{L, R\}$
- q_{0M} the start state. $q_{0M} \in Q_M$.
- F_M the set of final states. $F_M \subseteq Q_M$.

The function δ_M is the "program" of the Turing machine, giving its behaviour on each (state, input) pair of interest. Turing machines are interesting because they seem to capture formally the informal notion of computation.

We define an infinite set of communicating finite automata $A_i, i = 0, 1, 2, \dots$ as

$$A_i = (Q_A, \delta_A, q_{0A}, F_A, T_A, L_A, R_A) \quad (2)$$

where the symbols in the tuple have the following meanings:

- Q_A the finite set of states of the control
- δ_A the *next state* function
 $\delta_A : Q_A \times Q_A \rightarrow Q_A$
- q_{0A} the start state. $q_{0A} \in Q_A$.
- F_A the set of final states. $F_A \subseteq Q_A$.
- T_A the *transmitting* states. $T_A \subseteq Q_A$.
- L_A the *left-receiving* states. $L_A \subseteq Q_A$.
- R_A the *right-receiving* states. $R_A \subseteq Q_A$.

The function δ_A captures the notion of communication. If A_i is in a transmitting state, then the value of δ_A depends only on the current state of A_i . If A_i is in a left-receiving state, then the value of δ_A depends on the pair consisting of the current state of A_i and the current state of A_{i-1} (such states are undefined for A_0). If A_i is in a right-receiving state, then the value of δ_A depends on the pair consisting of the current state of A_i and the current state of A_{i+1} .

To simulate an arbitrary Turing machine M , we set the components of the A_i as shown below.

$$Q_A = Q_M \times \Gamma_M \times C \quad (3)$$

$$q_{0A} = (q_{0M}, B_M, \lambda) \text{ for } i > 0 \quad (4)$$

$$q_{0A} = (q_{0M}, B_M, \tau) \text{ for } i = 0 \quad (5)$$

$$F_A = \{(q, X, s) | q \in F_M, X \in \Gamma_M, s \in C\} \quad (6)$$

$$T_A = \{(q, X, \tau) | q \in Q_M, X \in \Gamma_M\} \quad (7)$$

$$L_A = \{(q, X, \lambda) | q \in Q_M, X \in \Gamma_M\} \quad (8)$$

$$R_A = \{(q, X, \rho) | q \in Q_M, X \in \Gamma_M\} \quad (9)$$

The set $C = \{\tau, \lambda, \rho\}$ labels the communications mode (transmit, left-receive, right-receive). The basic idea is that each automaton simulates both the finite control of M and one square of M 's tape. At any time, the automaton corresponding to the current tape square is in transmit mode (one of the τ states). The automata to the left of the transmit automaton are in right-receive mode (one of the ρ states). The automata to the right of the transmit automaton are in left-receive mode (one of the λ states).

The transition executed by the transmit automaton does two things. First, it causes a state change to propagate outwards from the transmit automaton to the surrounding automata. Second, it causes one of the neighbours of the transmit automaton to become the new transmit automaton, simulating a move of the read-write head of the Turing machine.

For each transition $\delta_M(q, X) = (p, Y, L)$, corresponding to a left move of the Turing machine head, we define corresponding transitions for the A_i :

$$\delta_A((q, X, \tau), (\text{no rec. state})) = (p, Y, \lambda) \quad (10)$$

$$\delta_A((r, Z, \rho), (q, X, \tau)) = (p, Z, \tau) \quad (11)$$

$$\delta_A((r, T, \rho), (p, Z, \rho)) = (p, T, \rho) \quad (12)$$

$$\delta_A((r, Z, \lambda), (q, X, \tau)) = (p, Z, \lambda) \quad (13)$$

$$\delta_A((r, T, \lambda), (p, Z, \lambda)) = (p, T, \lambda) \quad (14)$$

The first transition specifies that the current transmit automaton goes into left-receive mode, after executing

the desired state change. The automaton state records both the Turing machine state and the symbol written by the Turing machine. The second transition specifies that the automaton immediately to the left of the transmit automaton also records the Turing machine state change in its state and becomes the new transmit automaton. The tape symbol recorded in its state remains unchanged. The remaining transitions specify that all other receive automata states that will arise should lead to a propagation of the Turing machine state information, while preserving the Turing machine tape content information.

For each transition $\delta_M(q, X) = (p, Y, R)$, corresponding to a right move of the Turing machine head, there are similarly corresponding transitions for the A_i :

$$\delta_A((q, X, \tau), (\text{no rec. state})) = (p, Y, \rho) \quad (15)$$

$$\delta_A((r, Z, \lambda), (q, X, \tau)) = (p, Z, \tau) \quad (16)$$

$$\delta_A((r, T, \lambda), (p, Z, \lambda)) = (p, T, \lambda) \quad (17)$$

$$\delta_A((r, Z, \rho), (q, X, \tau)) = (p, Z, \rho) \quad (18)$$

$$\delta_A((r, T, \rho), (p, Z, \rho)) = (p, T, \rho) \quad (19)$$

There is a delay of one state-transition time at each automaton as the encoding of the Turing machine state propagates outward from the transmit automaton. Since this delay is the same for all operations, however, it has no effect on the outcome of any computation. Delay effects would become important if the states of the automata could be affected by the world they inhabit, which is inevitable for interesting systems.

The alternative of broadcast communications has also been considered. In order to adapt the above simulation to this case, we replace left-receive and right-receive modes with a single receive mode. In order for control to pass from one transmit automaton to a new transmit automaton, each automaton must know its own index number, and the broadcast transmission must include the index number of the transmit automaton. This is a tradeoff in which we exchange the uniformity of the individual automata in favour of a machine with a synchronous update of its components.

3.2 Exploring an unknown environment

In earlier work[6] we demonstrated that a single robot lacking metric information is unable to explore its environment, but that if the robot is equipped with a marker that can be put down and picked up at will then the robot can do so. We also developed algorithms for a single robot with a large number of pebbles. These results can be readily extended to robotic swarms with very limited communication distances by replacing the pebbles with individual members of a robotic collectivity.

We model the robot's environment as an embedding of an undirected graph G :

$$G = (V, E) \quad (20)$$

with set of vertices V and set of edges E . The vertices are denoted by:

$$V = \{v_1, \dots, v_N\} \quad (21)$$

We restrict the world model to graphs G that contain no cycles of length ≤ 2 , i.e. the graph contains no degenerate or redundant paths. This restriction prohibits the world from having multiple edges between two vertices or an edge incident twice at the same vertex.

The definition of an edge is extended slightly to allow for the explicit specification of the order of edges incident upon each vertex of the graph embedding. This ordering is obtained by enumerating the edges in a systematic (e.g. clockwise) manner from some standard starting direction. An edge $E_{i,j}$ incident upon v_i and v_j is assigned labels n and m , one for each of v_i and v_j respectively.

A robot can move from one vertex to another by traversing an edge (a *move*), and it can sense the presence or absence of a particular unit from the swarm at its current location. We limit sensing to nodes. All of the robots are homogeneous, but we identify a specific controller robot which will be used to herd all of the other robots.

Assume that a robot is at a single vertex, v_i , having entered the vertex through edge $E_{i,l}$. In a single move, it leaves vertex v_i for vertex v_j by traversing the edge $E_{i,j}$, which is r edges after $E_{i,l}$ according to the edge order at vertex v_i . This is given by the transition function:

$$\delta(v_i, E_{i,l}, r) = v_j \quad (22)$$

We assume the following property about the transition function:

if $\delta(v_i, E_{i,l}, r) = v_j$ and $\delta(v_j, E_{j,k}, s) = v_k$, then $\delta(v_j, E_{j,k}, -s) = v_i$.

This implies that a sequence of moves is invertible, and can be retraced. We also assume that there does not exist a $t \neq -s$ such that $\delta(v_j, E_{j,k}, t) = v_i$.

A single move is thus specified by the order r of the edge along which the robot exits the current vertex, where r is defined with respect to the edge along which the robot entered this vertex. Note that in the special case of a planar embedding of a graph, enumeration of edges in a clockwise fashion satisfies the above assumption.

A robot's perception is of two kinds, robot-related and edge-related perception.

Robot-robot-related Perception. Assume that a robot is at vertex v_i , having arrived via edge $E_{i,j}$. The robot-related perception of the another robot is a K -tuple $B_s = (bs_1, bs_2, \dots, bs_K)$, where bs_k has a value from the set $\{present, not-present\}$, according to whether robot k is present at vertex v_i .

Edge-related Perception. A robot can determine the relative positions of edges incident on the vertex v_i in a consistent manner, e.g. by a clockwise enumeration starting with $E_{i,j}$. As a result, it can assign an integer label to each edge incident on v_i , representing the order of that edge with respect to the edge enumeration at v_i . The label 0 is assigned arbitrarily to the edge $E_{i,j}$, through which the robot entered vertex v_i . The ordering is

local, because it depends on the edge $E_{i,j}$. Entering the same vertex from two different edges will lead to two local orderings, one of which is a permutation of the other. Note that if the graph is planar and a spatially consistent (e.g. clockwise) enumeration of edges is used, then two permutations will be simple circular translations of each other. But this will not hold in general, and in this paper we only assume that the edges can be ordered consistently.

The sensory information that the robot acquires while at vertex v_i is the pair consisting of the marker-related perception at that vertex and the order of edges incident on that vertex, with respect to the edge along which the robot entered the vertex. If the robot visits the same vertex twice, it must relate the two different local orderings produced and unify them into a single global ordering, for example by finding the label of the 0-th edge of the second ordering with respect to the first ordering. Determining when the same vertex has been visited twice and generating a global ordering for each vertex is part of the task of the exploration algorithm.

In earlier work [6] we demonstrated that it is not possible for a single robot to decisively explore and map an unknown environment with this sort of limited sensory information without either additional robots to assist it, or the ability to mark locations it visits. This is consistent with human intuition: fairy tales, and mythology are full of stores of heroes who escaped becoming lost within a maze by dropping markers or unwinding string as they went. The basic problem is that when the explorer enters an environment he cannot always determine if he has visited this location before.

In this earlier paper it was also demonstrated that as long as the explorer had a single unique marker which could be dropped and picked up at will it was possible for the explorer (or robot) to fully map his environment in $O(N^3)$ steps. The basic technique was to use the invertibility of the path taken by the robot, and the fact that the marker is unique, to disambiguate locations which could be confused. The algorithm proceeds by building a known map, and whenever a potentially new location is encountered, the marker is dropped in the new location and all nodes in the known map are visited. If the marker exists in the known graph then this new location corresponds to an existing location. On the other hand, if it is not found then this location really is new. In either case, additional information has been found, and more of the graph has been explored. In practice, there is usually more sensory information available than this result assumes and hence better performance can be achieved.

The approach with pebbles can be easily modified to work with swarms of robots. Take the case of a swarm of two robots. One robot (the controller robot) can treat the other robot like the pebble, having the pebble robot move on the controller robots command. Environmental exploration in this sensory deprived

world cannot be solved by a single robot but it can be solved by a swarm of two robots. If there is a large number of robots (say considerably more robots than nodes in the graph) then it is possible to have a much simpler swarm. Each robot simply moves until it finds an empty node and then stays there. As all of the robots start out in the same node then, in at most N moves there will be a robot at each node. As each node is now uniquely identified, it is straightforward for a robot to begin from the start node and visit all of the (now unique) nodes in the graph by traversing every edge twice ($O(N^2)$). If the swarm has the ability to communicate farther than the current node even more complex

The number of robot moves used in exploring a graph with a small number of robots has a bound of $O(N^3)$. This results from the need to go back and actually visit all of the locations in the graph to solve the "have I been here before" problem. With a large swarm of robots this worst-case bound will be reduced to $O(N^2)$ (and often lower in practice, for example $O(N)$ for a planar graph) since the swarm makes a costly re-examination of the known graph unnecessary. Different swarm compositions have other interesting implications which are unfortunately beyond the scope of this paper.

4 Discussion

In this paper we have suggested a taxonomy for the characterization of multi-agent robotic systems (swarms) based on the global properties of the swarm itself. These properties are:

swarm size,
communication range,
communication topology,
communication bandwidth,
swarm reconfigurability,
swarm unit processing ability, and
swarm composition.

This permits comparative analysis of different multi-robot systems to be performed and suggests a framework for the analysis of the advantages and abilities of different swarm architectures. In addition, we have briefly illustrated this with examples of two computational models based on swarm architectures at different points in this multi-dimensional taxonomy: generalized Turing-equivalent computation using "dumb" robots, and graph exploration and mapping using a mostly homogeneous swarm of communicating robots.

References

- [1] R. C. Arkin and J. D. Hobbs. Dimensions of communication and social organization in multi-agent robotics systems. In *SAB 92*, 1992.
- [2] G. Beni and J. Wang. Swarm intelligence in cellular robotic systems. In *Proc. NATO Advanced Workshop on Robotics and Biological Systems*, Il Ciocco, Tuscany, Italy, 1989.
- [3] P. Boas. Machine models and simulations. Technical Report CT-89-02, Institute for language, logic and information, University of Amsterdam, 1989.
- [4] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. of Robotics and Automation*, 2:14-23, 1986.
- [5] R. A. Brooks. Intelligence without reason. Technical Report AI Memo No. 1293, MIT, 1991.
- [6] G. Dudek, M. Jenkin, E. Milios, and David Wilkes. Robotic exploration as graph construction. *IEEE Trans. on Robotics and Automation*, 7(6):859-864, 1991.
- [7] F. E. Fich, P. L. Radge, and A. Widgerson. Relations between concurrent-write models of parallel computation. *SIAM J. Comput.*, 17:606-627, 1988.
- [8] S. Hackwood and G. Beni. Self-organization of sensors for swarm intelligence. In *Proc. Int. Conf. on Robotics and Automation*, pages 819-829, 1992.
- [9] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computing*. Addison Wesley, Redwood City, CA, 1991.
- [10] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation*. Addison Wesley, Reading, MA, 1979.
- [11] M. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 830-835, 1992.
- [12] E. Nitz and R. Arkin. Communication of behavioral state in multi-agent retrieval tasks. In *Int. Conf. Cons. R. and A.*, Hawaii, 1993. (to appear).
- [13] T. Ueyamam, T. Fukuda, and F. Arai. Configuration of communication structure for distributed intelligent robot system. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 807-812, 1992.
- [14] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*. Elsevier, Amsterdam, 1990.