Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems Yokohama, Japan July 26-30, 1993

A Taxonomy for Swarm Robots

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 robot acting alone. Many different models have been suggested for the makeup of such collections of robots. In this paper we present a taxonomy of the different ways in which such a collection of autonomous robotic agents can be structured. 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. Swarms are more than just networks of independent processors - they are potentially reconfigurable networks of communicating agents capable of coordinated sensing and interaction with the environment.*

Introduction

Although most mobile robotic systems involve a single robot operating in an environment, a number of researchers have considered the problems and potential advantages involved in having an environment inhabited by a number of robots. For some specific robotic tasks, such as exploring an unknown planet, it has been suggested that rather than sending one very complex robot to perform the task it would more effective to send a large number of smaller, simpler robots. Such a collection of robots is sometimes described 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.

Swarms offer the possibility of enhanced task performance, task reliability and decreased cost over more traditional robotic systems. Although they have this potential, many possible swarm designs are neither more efficient, nor more reliable, nor more robust than a comparable single (more complex) robot. In order for a swarm to have these advantages the swarm must be designed with these properties in mind. Many possible swarms architectures are neither efficient, inexpensive, or robust in the face of failures of elements of the swarm.

In addition to having these properties, it is essential that the swarm of robots 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 an explicit communication channel or indirectly through one robot sensing a change in other robots in its environment. Intra-swarm communication presents difficulties in terms of swarm efficiency, fault tolerance, and cost.

In this paper we attempt to enumerate key aspects of general multi-robot architectures in attempt to facilitate the comparative analysis of possible swarm models. Although this decomposition of swarm characteristics is not the only one possible, it captures many of the key high-level features of a multi-robot system. We have explicitly avoided application-specific issues and issues related to manipulation.

Many different swarm models have been proposed. The behaviour based control strategy put forward by Brooks [5] is quite well known and it has been applied to collections of simple independent robots, usually for simple tasks. Other authors have also considered how a collection of simple robots can be used to solve complex problems. Ueyama *et. al.* [15] propose a scheme whereby complex robots are organized in tree-like hierarchies with communication between robots limited to the structure of the hierarchy. Hackwood and Beni [10] propose a model in which the robots are particularly simple but act under the influence of "signpost robots". These signposts can modify the internal state of the swarm units as they pass by. Under the action of the signposts, the entire swarm acts as a unit to carry out complex behaviors.

Mataric [13] describes experiments with a homogeneous population of actual robots acting under different communication constraints. The robots either act in ignorance of one another, informed by one another, or intelligently (cooperating) with one another. As inter-robot communication improves, more and more complex behaviors 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 (or robotic "intelligence"). 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? How do we compare the structure of various possible swarms?

The information processing ability of a swarm is dependent upon a large number of factors including the number of units, their sensing abilities, their communication mechanisms, etc.[1, 14]. In order more fully to 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 more fully to understand the capabilities of robot swarms we propose the following taxonomy. This taxonomy is based upon a number of organizational dimensions. After defining the axes we relate some points in the ensuing space to existing parallel computational models and robotic systems and illustrate some problems which illustrate the different ways in which swarms from different classes operate and the different kinds of problems that they can solve.

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 strengths, 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 then 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.

ALONE 1 robot. The swarmless swarm case.

PAIR 2 robots. The simplest group.

LIM-GROUP Multiple robots. The number n is small relative to the size of the task or environment.

INF-GROUP $n \gg 1$ robots. There is effectively an infinite number of robots.

Two robots can, of course, perform problems which are impossible with a single robot. Almost any operation involving simultaneity or near simultaneity of events (such as turning two keys at the same time), is impossible with a single spatially limited robot. Multiple robots can be used to obtain speedups in terms of task performance subject to robot task synchronization.

A number of robot swarms assume either that the time available for the task (or the number of robots applied to the task) is unbounded and this provides a number of simplifications in terms of probabilistic task completion. As a simple example, consider the task of utilizing a number of non-communicating robots to clean an enclosed area. A randomly moving robot (or a swarm of randomly moving robots) will with probability approaching one eventually clean all of the environment. The robots can operate completely independently of each other. Note that if there are time (or robot) constraints, then it may not be possible to utilize such a simple swarm to complete the task.

By communication range: In most systems there are limits to the range of direct communication for any single robot. This is a function both of the communications medium and the robot distribution. We list three key classes for this dimension.

COM-NONE 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"[13], the robots must not only not communicate with each other they must not try to signal each other symbolically 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 symbolic communication. It may not be realistic to have truly ignorant robots.

COM-NEAR Robots can only communicate with other robots which are sufficiently nearby.
This corresponds to the communication mechanism proposed in [10]. Distance, in this context, can be interpreted either topologically (e.g. nearest neighbor) 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 neighborhood.

COM-INF 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[9]. 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 particularly hierarchy [15], or with specific controller robots [10]. 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

TOP-BROAD Broadcast. Every robot can communicate with all of the other robots. It is not possible to send a message to a particular swarm element.

TOP-ADD Address. Every robot can communicate with any arbitrary robot by name or address.

TOP-TREE Tree. Robots are linked in a tree and may only communicate through this hierarchy.

TOP-GRAPH Robots are linked in a general graph. This is a more general connectivity scheme than the tree and is more robust since redundant links can prevent the entire swarm from becoming disconnected.

Communication strategies based on either tree-like or Address based communication topologies are likely to be highly sensitive to failure of particular robots in the swarm. Failure of a particular robot will isolate robots on either side of the failed node in the hierarchy. Addressing implies distinctive roles for individuals; resulting in reduced interchangeability. Note that the actual set of robots that can communicate directly at any time is a function both of this dimension and of the communication range and robot distribution in space.

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.

BAND-HIGH 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.

BAND-MOTION Communication costs of the same order of magnitude of the cost of moving the robot between locations. 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 neighborhood.

BAND-LOW Very high cost. Communication costs much more than the cost of moving from one location to another. This suggests very independent robots.

BAND-ZERO No communication. Robots are unable to sense each other. As mentioned earlier, this is probably an impractical case if coordinated collective behaviour is desired.

Note that low bandwidth may be acceptable if the primary reason for using multiple robots is redundancy rather than efficiency.

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. Changes in topology, however, will alter the nearest-neighbor relationships and thus are not equivalent to simple 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).

ARR-STATIC Static arrangement. The topology is fixed.

ARR-COMM Coordinated rearrangement. Re-arrangement with members that communicate.

ARR-DYN Dynamic arrangement. The relationship of members of the swarm can change arbitrarily.

Static swarm arrangement is likely to result in very fragile swarms.

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.

PROC-SUM Non-linear summation unit[11]. 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.

PROC-FSA Finite state automaton. This is the computational model preferred by the subsumption architecture computational systems[4].

PROC-PDA Push-down automaton.

PROC-TME 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.

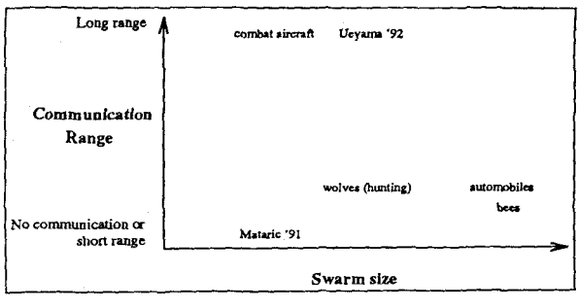

Figure 1: Some multi-agent systems classified

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). Two dimensional plots are typically not the best medium for displaying multidimensional data. The full taxonomic labelling of the example swarms shown in Figure 1 are given in Table 1.

Emergent properties - specific cases

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[16]. Nevertheless, due to the mobile nature of the processors, new and interesting results are possible.

The following 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.

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 [12]. 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

Swarm	Size	Comm. range	Comm. topology	Comm. bandwidth	Reconfigur-ability	unit processing	composition
combat aircraft	multiple	long	broadcast	free	dynamic	PROC-TME	Hetero
wolf pack	multiple	nearby	broadcast	$\approx$ movement	dynamic	PROC-TME	$\approx$ Homog
automobiles (incl. driver)	multiple	nearby	nearby	$\approx$ movement	coordinated	PROC-TME	Hetero
bees	infinite	nearby	nearby	$\approx$ movement	dynamic	PROC-TME	Hetero
Mataric92	multiple	varies	varies	varies	dynamic	PROC-TME	Homog
Hackwood92	multiple	nearby	broadcast	free	dynamic	PROC-TME	Hetero
Brooks	multiple	n.a.	n.a.	n.a.	n.a.	PROC-FSM	Homog

Table 1: Full taxonomic labelling of the sample swarms

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, $i = 0, 1, 2, ...$ 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
$\delta_A : Q_A \times Q_A \longrightarrow Q_A$
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:

$$\delta_A((q, X, \tau), (\textit{no rec. state})) = (p, Y, \lambda) \quad (10)$$
$$\delta_A((r, Z, \rho), (q, X, \tau)) = (p, Z, \tau) \quad (11)$$
$$\delta_A((r, T, \rho), (p, Z, \rho)) = (p, T, \rho) \quad (12)$$
$$\delta_A((r, Z, \lambda), (q, X, \tau)) = (p, Z, \lambda) \quad (13)$$
$$\delta_A((r, T, \lambda), (p, Z, \lambda)) = (p, T, \lambda) \quad (14)$$

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:

$$\delta_A((q, X, \tau), (\textit{no rec. state})) = (p, Y, \rho) \quad (15)$$
$$\delta_A((r, Z, \lambda), (q, X, \tau)) = (p, Z, \tau) \quad (16)$$
$$\delta_A((r, T, \lambda), (p, Z, \lambda)) = (p, T, \lambda) \quad (17)$$
$$\delta_A((r, Z, \rho), (q, X, \tau)) = (p, Z, \rho) \quad (18)$$
$$\delta_A((r, T, \rho), (p, Z, \rho)) = (p, T, \rho) \quad (19)$$

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.

Exploring an unknown environment. In [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 collective.

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

$$G = (V, E) \tag{20}$$

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

$$V = \{v_1, ..., v_N\} \tag{21}$$

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:

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

We assume the following property about the transition function: **if** $\delta(v_i, E_{i,l}, r) = v_j$ **and** $\delta(v_j, E_{i,j}, s) = v_k$, **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-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 $B_s = (bs_1, bs_2, ..., bs_K)$, where bs_k has a value from the set $\{present, not\text{–}present\}$, 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 [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 comes to a location 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 robot's command. Environmental exploration in this sensorily 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 $2N$ moves there will be a robot at each node. (If the number of robots is assumed to be unbounded then only N moves are required.) 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 robots have the ability to communicate with other robots at greater distances (eg. COMM-INF) even more complex algorithms are possible.

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.

Self organizing swarms. In [7] we show how a swarm composed of Turing-machine equivalent homogeneous elements (PROC-TME); capable of communicating only with other nearby elements (COM-NEAR), utilizing a broadcast communication topology (TOP-BROAD), free communication cost (BAND-HIGH), and with coordinated swarm reconfigurability (ARR-COMM), can group itself into long communication chains capable of transmitting messages from one end of the chain to the other. Elements of the swarm attempt to link into chains with other nearby swarm elements by identifying nearby swarm elements and utilizing a self-organizing mechanism to drive the swarm elements into one-dimensional chains. The self-organizing mechanism forces swarms to join chains whenever possible. The resulting chain is relatively insensitive to loss of swarm elements in that the chain will attempt to reform around the lost element. Each swarm element has a unique identification number which is used to force the chains to form in particular ways and to simplify the task of having chains join together.

Elements of the swarm are in one of two possible modes. They are either in a communication chain or they are not. If they are not in a chain they move randomly until they find another element in the swarm and attempt to join that elements chain or just with that element if neither of them are currently in a chain. When a robot is in a chain it moves with the chain attempting to retain communication up and down the length of the chain. When a chain meets another robot (or another chain) they join based on knowledge of the identification number of the head and tail of the two groups that have met.

Once formed, chains can be used for transport or communication. A reliable fixed link between two points can be formed by having robots with particular identification numbers fixed to the start and end points and having the chain form between them. Note that the chains are not particularly efficient. Nor are there guarantees that the elements will eventually form up in the desired way.

Robust positioning. In [8] we show how a collection of interacting autonomous robots can define a local coordinate system with respect to one another without reference to environmental features. This simplifies tasks requiring robots to occupy or traverse a set of positions in the environment, such as mapping, conveyance and search. We argue for an approach to positioning in which sensing errors remain localized, and in which dead-reckoning plays no role. This involves a robot-based representation of the environment, in which metric information is used locally to determine the relative positions of neighbouring robots, but the global map is a graph capturing the neighbour relations among the robots. We show that many tasks can be solved without reference to a global coordinate system, but that global metric maps may be constructed as desired, with small errors in the vicinity of any chosen position.

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 computational models based on swarm architectures at different points in this multi-dimensional taxonomy: generalized Turing-equivalent computation using "dumb" robots, graph exploration and mapping using a mostly homogeneous swarm of communicating robots, and the establishment of a reliable communication chain utilizing a homogeneous swarm of communicating robots with broadcast communication.

Swarms can be powerful mechanisms for performing highly complex tasks. They offer the promise of fault tolerance, high performance and low cost. Unfortunately many of the swarms that have been proposed in the literature do not have all three of these properties. For example, the Turing-machine and mapping swarms presented here is not particularly fault tolerant. Failure of any of the swarm elements or of the intra-swarm communication process will result in failure of the entire swarm. On the other hand the communicating swarm does provide fault tolerance - but pays for this in terms of performance. The communication chains are not necessarily efficient mechanisms for information transport and the swarm elements are not necessarily used in an efficient manner. The task of constructing realistic, efficient, inexpensive, and fault tolerant robotic swarms is a particularly difficult problem.

Acknowledgment

The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council. The authors would also like to thank John Zelek and Martin Levine for their helpful comments during the preparation of this paper.

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] G. Dudek, M. Jenkin, E. Milios, and David Wilkes. On the utility of multi-agent autonomous systems. In *IJCAI-93 Workshop on dynamically interacting robots*, 1993. (to appear).

[8] G. Dudek, M. Jenkin, E. Milios, and David Wilkes. Robust positioning with a multi-agent robotic system. In *IJCAI-93 Workshop on dynamically interacting robots*, 1993. (to appear).

[9] F. E. Fich, P. L. Radge, and A. Widgerson. Relations between concurrent-write models of parallel computation. *SIAM J. Comput.*, 17:606–627, 1988.

[10] S. Hackwood and G. Beni. Self-organization of sensors for swarm intelligence. In *Proc. Int. Conf. on Robotics and Automation*, pages 819–829, 1992.

[11] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computing.* Adison Wesley, Redwood City, CA, 1991.

[12] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computation.* Addison Wesley, Reading, MA, 1979.

[13] M. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 830–835, 1992.

[14] 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).

[15] T. Ueyama, 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.

[16] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity.* Elsevier, Amsterdam, 1990.