Fault Tolerant Coordination of Robot Teams

Jeffrey Coble Diane J. Cook

The University of Texas at Arlington
500 W. 1st St., MS #19066 Rm. 412 Woolf Hall
Arlington TX, 76019
817-272-3321
{coble,cook}@cse.uta.edu

Abstract

Coordinating interaction among robot teams in order to maintain a formation is difficult. The objective of our research is to develop fault tolerant methods of accomplishing this task that will perform well in real world environments. We are examining this problem from a distributed computing perspective and have experimented with symbolic machine learning as a method for addressing uncertainties in communication among distributed, autonomous robots. Our intent is to develop solutions that have a broader applicability to distributed computing environments and applications.

Introduction

Coordinating a robot team in the pursuit of a common goal is difficult. Elements of this problem can be equated to that of communication and coordination in distributed computing applications. In distributed computing applications, issues such as maintaining global state and determining causality among a series of events are fraught with difficulty, even under controlled conditions (Chow and Johnson 1997). There are known algorithms for addressing such problems, however they are designed to work in environments where communication links and processors are reliable. These algorithms quickly fail when a distributed computing application is introduced into a volatile environment, in which communication links may be unstable, the number of nodes may change, or adversaries may introduce errors into the communication channel (Chow and Johnson 1997).

Formation control of robot teams has been an issue in research on autonomous military scout vehicles (Arkin and Balch 1997) (Balch and Arkin 1995), satellite formation flying for communication and surveillance missions, and unmanned space vehicles (Bauer, et. al. 1997) (Corazzini, et. al. 1997), among others. Military scout vehicles often assume different formations, depending on such mission specific factors as terrain and the desired sensor coverage. Satellite formation flying is equally mission specific, depending on such factors as the area to be surveyed and the frequency. Unmanned space vehicles may be constrained by the escape vectors necessary for a satellite formation to resist the earth's gravitational pull.

Although there has been significant work related to coordinating robot teams, very few researchers have focused on expanding their approaches to incorporate fault tolerance. Our research focus is to develop intelligent methods for maintaining fault tolerant coordination in robot teams, with the hope of developing solutions that have a broader applicability to distributed computing environments and applications.

Coordination Strategy

Recent work in formation control has centered around three formation control strategies (Arkin and Balch 1997) (Balch and Arkin 1995). They are:

- **Unit-centered-referenced**: The center of the formation is computed by averaging the x and y positions of each member of the formation. Each robot can then determine its correct position in the formation by calculating its correct position from this center point.

- **Leader-referenced**: Each member of the formation calculates its position as a reference to the leader. The leader only concerns itself with its movement toward the goal, not in maintaining the formation.

- **Neighbor-referenced**: All members of the formation maintain their position relative to one other member.

In terms of creating a fault tolerant method of coordination, each to these approaches has its problems. The unit-centered approach requires that each robot in the formation be able to acquire the global state of the entire formation. This can either be accomplished by a broadcast method, where robots broadcast their new position periodically to the other members of the formation, or a querying method, where robots periodically request new position information from the rest of the members. Both methods can be communication intensive in moderately sized formations. Furthermore, achieving a precise snapshot of the state of all robots at a given time is difficult (Chow and Johnson 1997) and requires significant communication overhead just to ensure the accuracy of such a state at a given time. Coupling these difficulties

with the unreliability of wireless communication and the chance that a robot may be destroyed, unbeknownst to the other robots, makes using the unit-centered method a very challenging prospect for real-world systems. The leader-referenced method eliminates some of the difficulty of acquiring global state but may assume a communication capability that is not realistic. Some robotic formations may be geographically dispersed to the point where some robots may not be within communication range of the leader. The neighbor-referenced method is perhaps the most realistic but may introduce errors in the positional information that increase as they are propagated down the communication tree. Figure 1 illustrates four formations commonly used for military scout vehicles. The arrows indicate the flow of positional information when using the neighbor-referenced coordination approach.

There are clearly communication strategies that are more appropriate for some circumstances than for others. Therefore, it is not our intention to rule out any approach but simply to illustrate some of the potential problems with each. We are currently testing our work using the neighbor-referenced method because we believe it is a practical method of communication for robot teams that are distributed over a large geographic area and in a hostile environment. It is more robust in situations where communication connectivity may be spotty due to adverse topographical conditions.

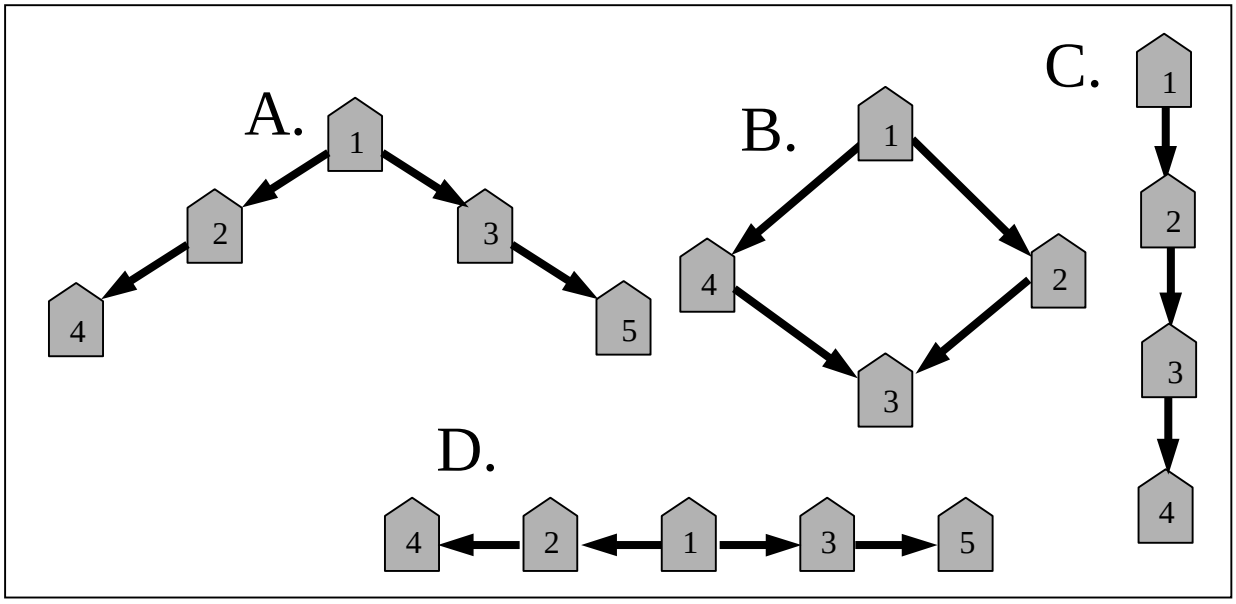

Figure 1. Common military scout vehicle formations. A. is the wedge, B. is the diamond, C. is the column, and D. is the line. The arrows indicate the flow of position information through the formation topology.

Dynamic Topology

We are initially focusing on the issue of the dynamic formation topology. In real environments, robots may be destroyed or encounter obstructions which will temporarily prevent them from communicating with some or all of the remainder of the formation. This problem is similar to one that occurs in distributed computing applications, where the failure of nodes or communication links can change the topology of the configuration.

With respect to topology changes, there are currently two situations in which we are interested. The first is the situation where a robot is destroyed. In such a circumstance, it would be desirable for the robot that was receiving positional information from the destroyed robot to assume the role of its missing neighbor. This would cause the gap in the sensor network to be closed and allow the formation to continue on its path to the goal. Figure 2, diagram A, illustrates the optimal conditions for the robot team. Diagram B illustrates the optimal response to the destruction of a formation member. The second situation in which we are interested is when communication between robots is temporarily obstructed. In such a circumstance, the optimal action for the robots would be to continue on their current course, avoiding any physical obstacles, and maintaining the formation. This situation is illustrated in figure 3.

Now that we have considered these two circumstances, we would like a robot team to be able to distinguish between the two and act appropriately. Our initial work deals with

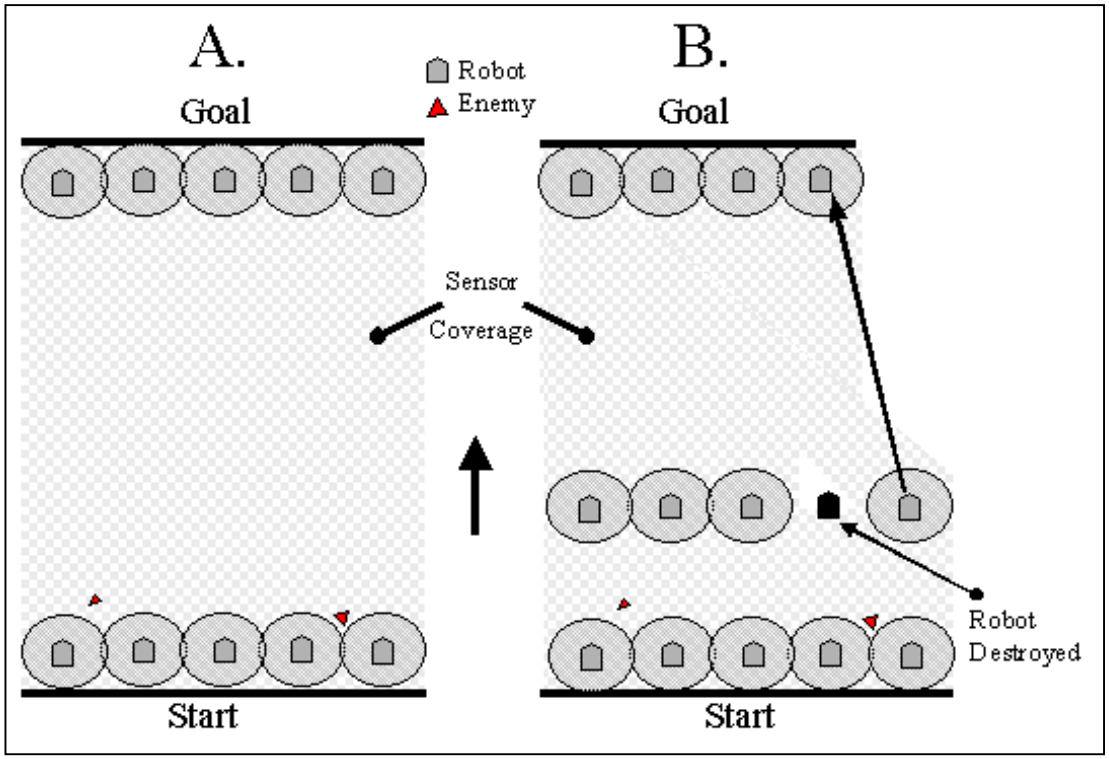

Figure 2. Robots moving over an area to conduct a sensor sweep. The shaded spheres surrounding the robots indicate intersection of communication/sensor ranges. Diagram A depicts the sensor coverage when robots coordinate properly under optimal conditions. Diagram B depicts the optimal response to the destruction of one of the robots in the formation. The robot closes the gap in the formation.

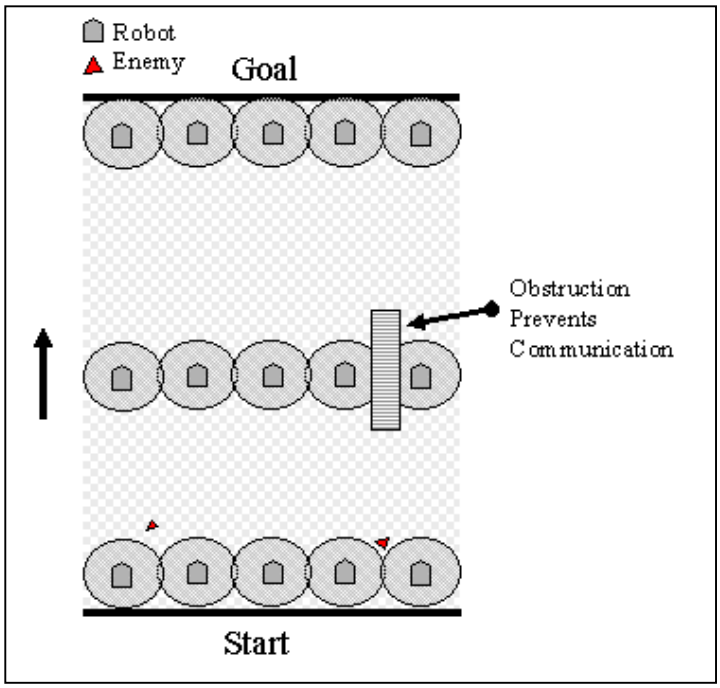

Figure 3. Depicts the optimal response to a temporary communication obstruction. The robots maintain their original roles, as they should.

using machine learning techniques for providing a robot with a means by which it can make these decisions.

We have constrained our work to the problem of maintaining a formation of military scout vehicles. The goal is to keep the vehicles in a formation so that they can move from a target location to a goal location and produce

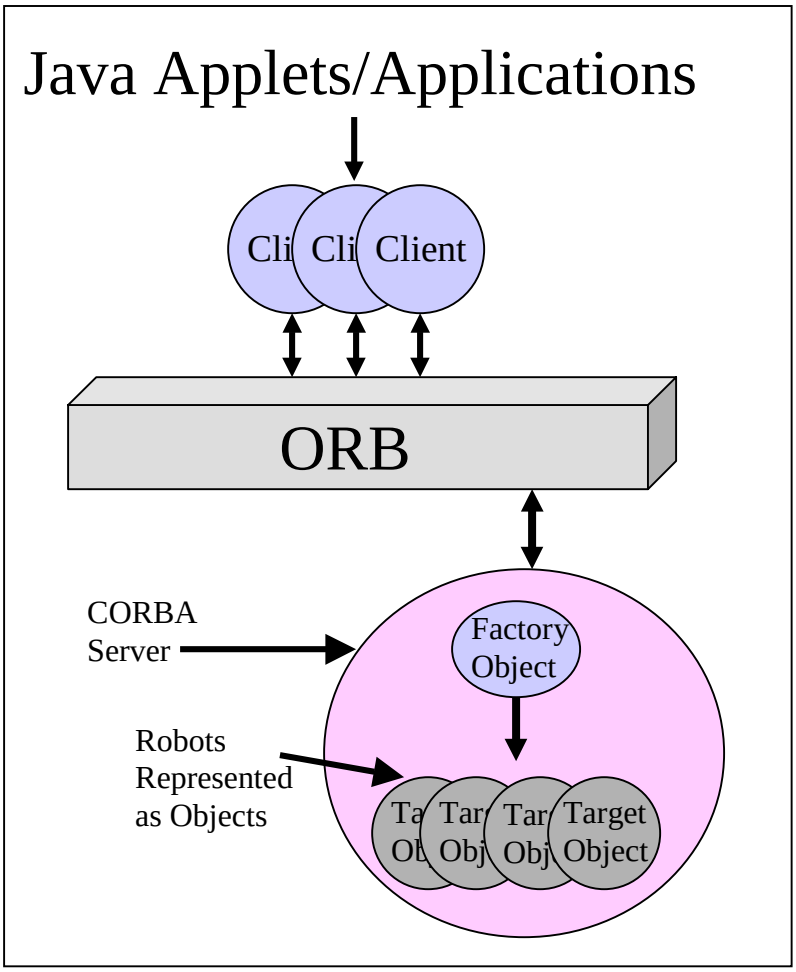

Figure 4. CORBA based architecture for robotic team simulation.

a sensor sweep of a continuous area.

We make the following assumptions about the problem:

1) Communication between the robots will not be reliable. Factors which may influence this are environmental, topographical, failure of a robot counterpart, or adversaries interfering with communication.

For instance, obstacles may block communication for periods of time or a robot in the formation may be destroyed.

2) Sensors, such as visual tracking methods, will not be completely reliable and are very sensitive to range. Some missions may require the robots to be geographically distributed, such as those where it is desirable to maximize the sensor coverage of the formation. In such a case, visual sensors would have limited value for coordinating with other robots.

3) Some positions in a formation may be more vulnerable to destruction than others. Therefore, a robot's position in the formation, and subsequently the position of its neighbor, has meaning when determining the likelihood of a missing neighbor versus a temporary loss of communication.

First, let us consider the reasons why we would want a robot to assume the role of a missing neighbor. The mission of the robots is to provide a sensor sweep of a particular path. Since each robot is capable of sensing a finite region, the sensing regions must overlap in order to capture the global state of a continuous region. Gaps in the sensor coverage diminish the value of the mission since

there is now inconclusive information about the swath covered by the robots.

Simulator Architecture

We are developing a simulator to validate our methods of formation control. Figure 4 illustrates the simulator architecture. The simulator consists of a CORBA server, which contains a single factory object. In general, a factory object is an object that allows clients to invoke methods to create other objects dynamically.

In our simulator, the clients initiate a connection to the CORBA server and binds to the factory object. The client can now invoke a method within the factory object to create target objects that represent the robot vehicles. The target objects can be instantiated with such parameters as formation type and communication strategy.

The client is responsible for visually representing the movement of the robots to the user. However, all communication and state maintenance is done at the server level, between the target objects. The client only represents the result of this interaction to the user. The CORBA server is multi-threaded so that any number of clients may invoke a simulation. The clients are written in JAVA. This choice was made simply for platform independence and so that clients could be embedded within a web browser. The CORBA Server, factory object, and target objects are written in C++. The availability of a C++ CORBA ORB (Object Request Broker) was the deciding factor for this implementation, but alternative object-oriented languages could be substituted.

By representing the robots as objects, we are able to support a variety of communication methodologies. Furthermore, this architecture allows us to create a discrete event simulation (Shen 1996), which allows us to study the effects of ordered event delivery and the results of introducing incremental error levels, which is represented by dropping messages in a random or patterned manner. CORBA provides a powerful tool for interaction among objects (Baker 1997) and provides distributed access to those objects.

We are in the process of implementing our simulator. Future work will illustrate the use of the simulator to test various communication strategies and incremental error rates. The simulator will demonstrate our integration of intelligent methods of error recovery and will provide us with an automated mechanism for generating training data for machine learning algorithms.

Machine Learning Techniques

Our initial approach at introducing fault tolerance into the robot coordination problem revolves around machine learning techniques. For this first study, we are using C4.5 (Quinlan 1993), a symbolic machine learning algorithm, to learn a decision tree that will classify contextual information and allow the robots to react to dynamic changes in a set of parameters. A decision tree generally represents a disjunction of conjunctions. Every branch of the tree (a path from the root to a leaf) represents a conjunction of attribute values. The entire tree, considered as a whole, represents a disjunction of these conjunctions (Mitchell 1997). In our work, we are considering the following attributes:

- *Number_of_Robots*: This attribute represents the number of robots in the formation.
- *Communication_Delay*: This is a continuous value that represents the communication delay that a robot experiences when expecting information from its neighbor. The values range from 0 to 1, with anything above 0 being considered abnormal. We are currently using increments of 0.1
- *Formation_Type*: wedge, line, diamond, column. These are military scout vehicle formations.
- *Obscurance_Level*: This is a continuous value that represents the level of topographical obscurance that a robot believes is present in the environment. The values range from 0 to 1, with anything above 0 meaning that the environment offers some amount of potential communication obscurance. We are currently using increments of 0.1
- *My_Role*: This is an integer value that is less than or equal to the number of vehicles in the formation.

The C4.5 algorithm takes as input a set of training examples, designed to explain some of the possible scenarios of the aforementioned attributes, coupled with a classification value. The classification values are *Change Role* and *Don't Change Role*. We would like our robot to decide, based on its real-time assessment of the attribute values, whether or not to assume the role of its neighbor or to continue in its current role. So, the robot is attempting to ascertain whether or not its neighbor has been destroyed or just temporarily obstructed from communicating.

Preliminary Results

We evaluated this symbolic learning method by generating a set of training data and running a cross validation experiment. To create the data, we set the *Number_of_Robots* equal to thirty, an arbitrary starting point, and made some general assumptions about the vulnerability of various positions in the four formations illustrated in figure 1. For instance, we assumed that a robot whose position was closer to the exterior of the line formation was more likely to be attacked by an enemy and subsequently destroyed. To reflect this in the training data, we weighted certain positions in the formations on a linear scale of vulnerability to destruction. So, we used the *Number_of_Robots*, *Formation_Type*, and *My_Role* attributes to create this weight. In addition to this weight, we considered the *Communication_Delay* value and the

Obscurance_Level value. To classify the training examples, we normalized and averaged these three values and classified anything over 0.5 as *Change Role* and anything under 0.5 as *Don't Change Role*.

Our experiment consisted of 1,728 training examples. For each of the four formations, we generated a set of training examples where the number of vehicles in the formation ranged from three up to thirty. For each of these formations, we generated a training example for each possible role and we then randomly assigned a *Communication_Delay* value and an *Obscurance_Level* value to each one.

We used C4.5 to cross validate the results across ten blocks of data. C4.5's cross validation operation broke up the data into ten equal groups, so that each group's number of cases and class distribution was as uniform as possible. C4.5 then constructed ten decision trees, using nine of the ten blocks as training data while holding one different block out for test data each time. The average error rate for this ten-fold cross validation was 13.8%. This error rate is comparable to other successful applications of symbolic machine learning and is along the lines of what we expected for a problem that is suitable for a symbolic machine learning solution. Further analysis of our method for generating training data may yield ways to reduce this error rate further.

The method we used for generating the classification values for the training data is only designed to illustrate the combination of a potentially large set of simple attributes. The completion of our simulator will allow us to more accurately generate training data and classification values. It will also allow us to experiment with other attribute values, possibly dependent upon various communication methods, and potentially a more complex classification scheme.

These early results lead us to believe that symbolic machine learning is an appropriate method for dealing with some of the uncertainties present in this and other distributed computing applications. Further work will explore sensitivities in the various parameters and validate a robots ability to employ machine learning techniques in its real-time response mechanisms.

Future Work

This paper represents the beginning of a research project to examine solutions to the coordination of robot teams in the context of distributed computing applications. Our future work will include completion of our simulator so that we may adequately test and examine our results. We will continue to examine symbolic machine learning methods and the integration of these methods into real-time robotic control algorithms.

The continuous nature of the attributes in this domain strongly indicates the potential value of probabilistic reasoning. Such methods would not return a discreet classification value, but rather a probability distribution for the desired attributes. In the case of robotic formation control, we believe it is possible to construct a Bayesian Network (Russell and Norvig 1995) that would encode the attribute values and their probabilistic relationship to each other. The information that the robot would receive from the Bayesian Network would be a probabilistic assessment of the likelihood that the robot's neighbor was destroyed. It seems reasonable that the robot would gradually adjust its course based on the strength of this likelihood assessment. In the future, we will be assessing the ability to learn a Bayesian Network from simulated (or real) data and to use that Bayesian Network to reason about the robot's course adjustments, based on real-time environment parameters.

We will continue our investigation of these solutions in the general domain of distributed computing environments and applications.

Conclusions

Our initial work has focused on evaluating the applicability of symbolic machine learning methods to deal with the uncertainty that is inherent to distributed computing environments.

We have tested our approach on a simple configuration of a robotic team coordination problem. We have selected a set of attributes that a robot must consider when determining whether its neighbor has been destroyed or is temporarily obstructed from communicating. These tests were conducted with the C4.5 machine learning algorithm. Our preliminary results indicate that the decision tree generated by C4.5 is capable of providing an effective tool that will allow a robot to resolve its dilemma in this circumstance.

References

1. Arkin, R.C. and Balch, T.R., *Cooperative Multiagent Robotic Systems* AI-based Mobile Robots: Case Studies of Successful Robot Systems , D. Kortenkamp, R.P. Bonasso, and R. Murphy (eds), MIT Press, in press.
2. Baker, S., *CORBA Distributed Objects Using Orbix*, ACM Press, NY, NY, 1997
3. Balch, T. and Arkin R., *Motor Schema-based Formation Control for Multiagent Robot Teams*, Proceedings of the International Conference on Multiagent Systems. San Francisco, CA April 1995.
4. Bauer, F., et. al., *Satellite Formation Flying Using an Innovative Autonomous Control System (AUTOCON) Environment*, AIAA Guidance, Navigation, and Control Conference August, 1997
5. Chow, R. and Johnson, T., *Distributed Operating Systems & Algorithms*. Addison Wesley Longman, Inc., 1997
6. Corazzini, T., et. al, *GPS Sensing for Spacecraft Formation Flying*, Proceedings of the Institute of

Navigation GPS-97 Conference, Kansas City MO, September 1997.
7. Mitchell, T., *Machine Learning*, McGraw-Hill Publishers, NY, NY, 1997.
8. Quinlan, J., *C4.5: Programs for Machine Learning*, Morgan Kaufman Publishers, Inc, San Mateo, CA, 1993.
9. Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall Inc, 1995.
10. Shen, C., *A CORBA Facility for Network Simulation*, in the Proceedings of the 1996 Winter Simulation Conference (WSC96), Coronado, California, December 1996.
11. Suzuki, I. And Yamashita, M., *Distributed Anonymous Mobile Robots---Formation and Agreement Problems*, in the Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO '96), Siena, Italy, June 1996.