# Self-Assembly in Swarm Robotics Systems

Erkin Bahçeci

Department of Computer Engineering
Middle East Technical University
November, 2003

# Abstract

This report describes recent approaches proposed for self-organization and pattern formation, and introduces an evolutionary approach to the pattern formation and self-assembly task. Parallelized genetic algorithms are used in this study to evolve neural network controllers for reactive robots. The robots have probabilistic behaviors through use of random inputs, which improve overall performance.

# Contents

# 1 Introduction

Various animal species show self-organized behaviors among its members, where the individuals stay at a specific orientation and distance with respect to each other while moving, or fill a specific area as homogeneously as possible. These behaviors involve some type of pattern formation, and include bird flocking, fish schooling, and ants moving in chains. This fact brings up the questions of whether similar behavior can be obtained with robots and whether there are beneficial application areas for such behavior other than biological behavior modeling.

Current application areas of pattern formation include search and rescue operations, landmine removal, remote terrain and space exploration, control of arrays of satellites and unmanned aerial vehicles (UAVs). Another possible application might come up with developments in nanorobotics, where multiple nano-scale robots would be required to act and move in a specific formation.

If the robots have complete information, i.e., positions of all robots or GPS data, then the problem is relatively easy. In this case, an almost perfect pattern can be obtained by even distributed computation [9]. However, if the robots do not know everything, then accomplishing the pattern formation task becomes very challenging. In that case, the robots should have enough sensor capabilities to detect their relative positions and orientations with respect to others. They should also communicate and cooperate well enough to handle local robot-robot interactions to build and maintain a global pattern.

In this study, a case is considered where robots have minimal communication and no world representation and are completely reactive. The task of the robots is to form a particular pattern and perform self-assembly, i.e. hold onto each other. The approach used is evolutionary robotics, where controllers for robots are evolved using genetic algorithms with a fitness function representing how well the desired behavior is performed.

# 2 State of the Art

Properties of studies in cooperative mobile robotics can be stated under taxonomies. Cao et al. [20] proposed a taxonomy which has the following axes:

- Group Architecture.

  Centralization.

  Homogeneity.

  Means of communication (environment, sensing, or direct communication).

  Modeling of other agents.

- Resource conflicts.
- Origins of cooperation.
- Learning.

- Geometric problems.

Dudek et al. [4] claimed that these axes are highly interdependent and very broad, which makes it difficult to identify isolated sample points within the taxonomy. They put forward a slightly different set of taxonomy axes:

- Collective size.
- Communication range.
- Communication topology.
- Communication bandwidth.
- Collective reconfigurability.
- Processing ability.
- Collective composition (homogeneity).

The studies mentioned below vary on centralization, communication means, homogeneity, communication bandwith, and processing ability axes. They do not include learning or modeling of other agents.

## 2.1 Centralized Pattern Formation

Centralized studies on pattern formation have path-planning that is done not by each robot but only by a more sophisticated robot or by a stationary computing unit [1, 10, 18, 19]. The decided path for each robot is transmitted to it using a communication channel.

One of the studies involving centralization is done by Egerstedt and Hu [10]. It is a model independent coordination strategy for multi-agent formation control. Their study decouples path planning from path tracking for the problem of constrained formation with a desired path for the group. Path planning, which is done centrally, and the tracking of virtual reference points are handled separately. Egerstedt and Hu formulate the formation constraints and the movement of the robots with the assumption that rotational and translational velocities of the robots can be controlled. The path for a virtual leader is computed which gives reference points for the actual robots to follow. A virtual application of the method on executing coordinated movement in a triangular formation while avoiding an obstacle is presented. In this example, the robots which form the corners of the triangle, go around an obstacle, which falls in between the robots while they pass. This paper proves that if the tracking errors of the robots are bounded or tracking is done perfectly, then the described method stabilizes the formation error.

Koo and Shahruz [18] propose a centralized path-planning method, which aims to make unmanned aerial vehicles (UAVs) fly in a desired formation. This is achieved by computing path of each UAV by a leader UAV initially, which is more capable than others. Only the leader has cameras and sensors. It tells the other UAVs, via a communication system, what trajectories they should track. What UAVs should do is to

take off and fly toward their trajectories and lock onto them. Two cases are considered in experiments: the case where UAVs take off one by one, and where they do it simultaneously. Trajectory computation is the main focus of this study.

Belta and Kumar [1] describe another centralized trajectory computation scheme, which makes use of kinetic energy shaping, i.e., instead of using a constant kinetic energy metric, employing a smoothly changing the kinetic energy metric. The method generates smooth trajectories for a set of mobile robots. It has a parameter which determines how close to each other the robots will try to stay.

The disadvantages of the proposed trajectory planning method are that it does not take obstacle avoidance into consideration and that it is not scalable, since the complexity of the computations increase with the size of the team.

A target assignment strategy for formation building problem is described by Kowalczyk [19]. Starting with a scattered group of robots, the algorithm first assigns a target point for each robot in the desired final formation. Then it generates necessary priorities and trajectories for the robots to avoid collisions while moving to their target points. Each robot has an area around some part of its path which is forbidden to enter for robots that have a lower priority. If a forbidden area crosses a robot's trajectory, it waits until the robot with higher priority moves out of its way. The method is tested with non-holonomic, i.e. which cannot move toward every direction (e.g. a car), and holonomic robots. The disadvantages of this method are that it requires centralized execution of the algorithm and a global sensor system. Also scalability of the method is not addressed.

## 2.2 Decentralized Pattern Formation

Centralized computation can be an undesirable feature in multi-robot applications since it decreases the overall robustness of the system. Furthermore, centralized systems are more costly due to global communication channels. Therefore, decentralized approaches are more preferable in autonomous robotics. When there is no central processing or task allocation, failures of a small number of robots do not cause total system breakdown.

Communication and completeness of information known by robots impose a tradeoff between precision and feasibility of forming and maintaining the pattern and the necessity of global information and communication. Studies that require global information or broadcast communication [21, 15, 9] may suffer from lack of scalability or high costs of the physical setup but allow more accurate forming of a greater range of formations. On the other hand, studies using only local communication and sensor data [16, 17, 7, 2, 13, 12, 5, 8] tend to be more scalable, more robust, and easier to build; but they are also limited in variety and precision of formations.

Sugihara and Suzuki [9] achieve pattern formation by providing each robot the global positions of all others. In this study, an algorithm is developed for each pattern. It successfully accomplishes uniform distribution of robots in several pattern formations (circle, polygon, line, filled circle, and filled polygon), and separation of robots into an arbitrary number of nearly equal sized groups. Even though this method is successful and the control mechanism is decentralized, the requirement of complete

information being supplied to all robots may render the method useless in many applications.

Carpin and Parker [15] introduced a cooperative leader following strategy for a team of robots. The robots are able to maintain a specific formation while simultaneously moving in a linear pattern and avoiding dynamic obstacles. The robots use local sensor information and explicit broadcast communication among themselves. The framework handles heterogeneous teams, i.e. comprising of robots with different types of sensors, as well as homogeneous ones.

Two levels of behaviors were implemented for tasks: team-level and robot-level behaviors. Transitions are made when necessary among specific behaviors in these two levels. For example, when a member of the team faces an obstacle, the whole team waits together with that member for it to go away for a certain amount of time. If this time is exceeded that member circumnavigates the obstacle and the team returns to its main task of moving in a formation.

Balch and Hybinette [16, 17] discuss a different kind of strategy for robot formation keeping, which resembles the way molecules form crystals. In this study, each robot has several local *attachment sites* that other robots may be attracted to. This concept is similar to molecular covalent bonding. Possible attachment site geometries include shapes resembling an 'X', an 'I', and a '+', where the robot is the center of the shape and the attachment sites are the ends of the line segments. Various robot formation shapes result from usage of different attachment site geometries just as different crystal shapes emerge from various covalent bond geometries. When a team of robots moving in a formation, they avoid the obstacle by splitting around it and rejoining again one passed it.

This approach is scalable to large robot teams because global communication is not necessary. Local sensor information is sufficient to generate effective formation behavior in large robot teams.

Another method similar to crystal generation which employs a form of probabilistic control is proposed by Fujibayashi, Murata, Sugawara, and Yamamura [8]. This study makes use of virtual springs to bring two agents near. Each pair of robots within a certain range of each other, are connected via a virtual spring. Each agent is classified by the number of neighboring agents within this range (number of connections).

The robots form triangle lattices that have random outlines. To obtain a desired outline, the virtual springs among some robots are broken with a certain probability. The candidate springs to be broken are chosen depending on the number of connections the robots it joins have. This *breaking preference* and the probability of breaking changes from formation to formation as well several other parameters. This algorithm uses only local information and is decentralized. One disadvantage of this method is the difficulty of choosing custom parameters for each formation. Genetic algorithms may be used for this purpose.

A graph-theoretic framework is proposed by Desai [7] for the control of a team of robots moving in an area with obstacles while maintaining a specific formation. The described method uses control graphs to define behaviors of robots in the formation. This framework can handle change of formations, i.e. transitions between control graphs. This study includes proofs of the mathematical results required to enumerate and classify control graphs. Although the computations for control graphs increase

with the number of robots, the fact that these computations are decentralized allows the methods described to be scalable to large groups.

Another graph-based approach to *moving in formation* problem is introduced by Fierro and Das [13]. It proposes a four-layer modular architecture for formation control. Group control layer is the highest layer and it generates a desired trajectory for the whole group to move for a particular task. Formation control layer implements a physical network, a communication network, and a computational network (control graph). It maintains the formation by using local communication and relative position information. Kinematics control layer deals with the required linear and angular velocities of robots. And finally dynamic control layer handles the task of realizing the necessary speeds given by the kinematics control layer. This four-layer architecture provides an abstraction among tasks required at different levels. For example, a robot with different mass, inertia, and friction can be used only by changing the dynamic control layer. Furthermore a modular adaptive controller is described which can manage control of robots with unknown dynamics and learns the robot dynamics on-the-fly. Hence using a different robot requires no change in the system. The method described is scalable (control algorithms scale linearly) and flexible (it allows various formations). Centralized and decentralized versions of control graph assignment algorithm are also described in the study.

Kostelnik et al. [12] describe an approach for scalable multi-robot formations using only local communication and sensor information. Obstacle avoidance is also provided in this method. It extends ordinary behavior-based approaches with the application of social roles that represent positions in the formation and with the use of local communication to improve performance. As new agents join the formation, the shape is *fixed* by local communications and role changes where necessary. The locally communicated information reaches the leader, i.e. the front most robot, which knows the whole shape of the current formation and which decides on the changes necessary. This information is then propagated to the necessary followers, and the formation is updated.

There is no need to predefine social roles or positions for robots. Everything is done dynamically as the formation grows. This method supports various formations and also switching between them, therefore it is flexible as well as being scalable and local.

A tool is implemented by Dudenhoeffer and Jones [2] to model and simulate collective behavior and interactions of a group of thousands of robots. Using this simulation tool, the problem of hazardous material detection by thousands of micro-robots scattered around a region is tackled.

Social potential fields are utilized for coordinated group behavior where robots are desired to stay at a specific distance from others to obtain optimum coverage of the area. They are also required to wander in this formation to search other parts. The desired behavior is obtained by using subsumption architecture. This study also validates the proposed method in cases where it is possible for agents to die and where agents have imperfect sensor readings. The method uses only local information and is scalable to very large groups of robots.

In their study [5] Mataric and Fredslund use local information to establish and maintain formations among robots. Each robot has a unique ID and each robot has a designated friend robot which it can see through a *friend* sensor. There is also minimal

communication going on among robots: heartbeat signals (robots broadcast their IDs), swerve signals (changing direction), and formation messages. Each robot can learn the number of robots in formation and the type of formation using broadcasted messages. For each formation, each robot has a specified angle which determines the angle it should keep between its front direction and the direction of its friend. This angle is calculated locally. The details of this calculation are given in [5].

This study accomplishes the task of establishing and maintaining formations using only local information and minimal communication. However the possible formations are limited to chain-shaped ones that do not make a backward curve.

## 2.3 Evolutionary Approaches

Evolutionary methods are being more and more exploited in multi-robot systems, especially where the controllers needed are cumbersome to implement manually. Mataric and Cliff [11] have discussed the challenges of using genetic algorithms to evolve controllers for physical mobile robots. They have reviewed papers which use at least one of the following approaches: genetic programming, evolving neural network controllers, evolving morphology, evolution by shaping, evolving control hardware, using real vision as input, evolving in simulation, testing on real robots, and evolving entirely on real robots.

Mataric and Cliff also focus on main difficulties and important issues in evolving on physical robots, evolving in simulation, evaluation, fitness functions, co-evolution, encoding of chromosomes, hierarchical evolution, i.e., where building blocks are evolved entities themselves. The authors draw attention to many details such as criterion of genetic coding, necessary level of noise in simulations, and importance of free parameters in evaluation. The bottom line of this discussion is that it still requires less effort to design mobile robot controllers manually than to evolve them; but if the challenges discussed are overcome, then evolutionary methods may prove to be a practical alternative to manual design.

A behavior-based approach to achieve formation control, which uses evolutionary methods, is described by Cao et al. [21]. Motor schema-based architectures are preferred over subsumption architecture since they were claimed to be more flexible. The robots have four primitive behaviors: move to goal, keep formation, avoid static obstacle and avoid robot behavior. These behaviors are combined according to some control parameters, which are determined by a genetic algorithm. Each member of the population stores a set of parameters, and the populations are evolved using genetic operators. The fitness evaluation function is a simulation run of some specific number of steps. For the *formation keeping* behavior, a leader is selected which broadcasts its next position at each step. The others, i.e. followers, try to maintain the necessary relative position with the leader.

This method shows some degree of adaptation to new obstacles in the environment. A sample result is provided, which shows that a triangle formation is maintained while avoiding obstacles with and without a new obstacle that didn't exist during training. The method uses broadcast communication and local sensor information. Scalability is not verified in this study.

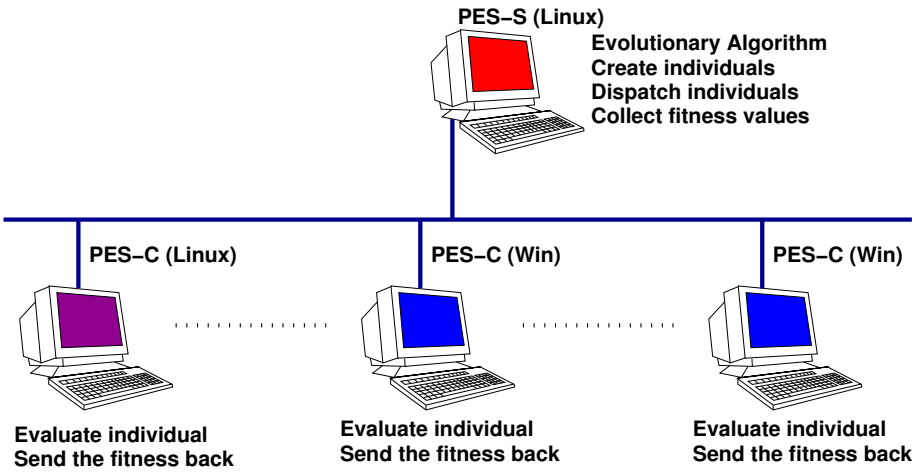


Figure 1: Sketch of PES system is shown. The PES-Server runs on a Linux machine and handles the management of the evolutionary method. It executes the selection and reproduction of the individuals (genotypes) which are then dispatched to a group of PES-Clients (running both Windows and Linux systems). The individuals are then evaluated by the clients and their fitnesses are sent back to the server.

# 3 Infrastructure

Parallel Evolution System (PES) is developed together with Onur Soysal to be used in studies under Kovan Research Lab [1].

PES is a platform to parallelize evolutionary methods on a group of computers connected via a network. It separates the fitness evaluation of genotypes from other tasks (such as selection and reproduction) and distributes these evaluations onto a group of computers to be processed in parallel. PES consists of two components; (1) a server component, named PES-Server, that executes the evolutionary method, the management of the communication with the client computers, and (2) a client component, named PES-Client, that executes programs to evaluate a single individual and return the fitness back to the server. Figure 1 shows the structure of a PES system.

PES provides the user with an easy interface that relieves him from dealing with the communication between server and client processes. PES-Client is developed for both Windows and Linux, enabling the PES system to harvest computation power from computers running either of these operating systems. An easy-to-use framework for implementing evolutionary methods, and the inter-operability of the system distinguishes PES from other systems available and makes it a valuable tool for evolutionary methods with large computational requirements.

PES uses PVM (Parallel Virtual Machine)[6][2], a widely utilized message passing

[1] Kovan Research Lab home page: http://www.kovan.ceng.metu.edu.tr.

[2] Available at http://www.csm.ornl.gov/pvm/pvm_home.html.

library in distributed and parallel programming studies, for communication between the server and the clients. We have also considered MPI[14] as an alternative to PVM. MPI is a newer standart that is being developed by multiprocessor machine manufacturers and is more efficient. However PVM is more suitable for our purposes since (1) it is available in source code as free software and is ported on many computer systems ranging from laptops to CRAY supercomputers, (2) it is inter-operable, i.e. different architectures running PVM can be mixed in a single application, (3) it does not assume a static architecture of processors and is robust against failures of individual processors.

PES wraps and improves PVM functionality. It implements a time-out mechanism is implemented to detect processes that have crashed or have entered an infinite loop. It provides *ping*, *data* and *result* message facilities. Ping messages are used to check the state of client processes. Data messages are used to send task information to client processes and result packages are used to carry fitness information from clients.

PES consists of two components: PES-Server and PES-Client.

## 3.1 PES-Server

PES-Server provides a generic structure to implement evolutionary methods. This structure is based on Goldberg's basic Genetic Algorithm[3] and is designed to be easily modified and used by programmers. The structure assumes that fitness values are calculated externally. In its minimal form, it supports tournament selection, multi-point cross-over and multi-point mutation operators.

PES-Server maintains a list of potential clients (computers with PES-Client installed), as specified by their IP numbers. Using this list, the server executes an evolutionary method and dispatches the fitness evaluations of the individuals to the available clients. The assignment passes the location of the executable to be run on the client as well as the parameters that represent that particular individual and the initial conditions for the evaluation. Then it waits for the clients to complete the fitness evaluation and get the computed fitness values back.

PES-Server contains fault detection and recovery facilities. Using the *ping* facility the server can detect clients that have crashed and assign the uncompleted tasks to other clients. In its current implementation, the server waits for the evaluation of fitness evaluations from all the individuals in a generation before dispatching the individuals from the next generation.

## 3.2 PES-Client

PES-Client acts as a wrapper to handle the communication of the clients with the server. It fetches and runs a given executable (to evaluate a given individual) with a given set of parameters. It returns the fitness evaluations, and other data back to the server.

Client processes contain a loop that accepts, executes and sends result of tasks. Client processes reply to *ping* signals sent by the PES-Server to check their status. Crashed processes are detected through this mechanism.

PES-Clients are developed for single processor PC platforms running Windows and Linux operating systems. Note that to use clients with both operating systems the fitness evaluating program should be compilable on both systems. In its current

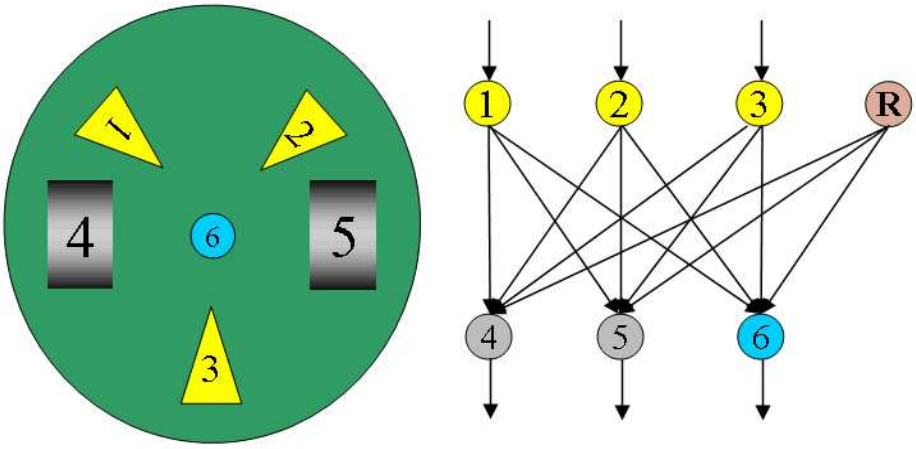


Figure 2: The robot structure (top view) and the neural network controller. The components of the robot and the corresponding neurons in the controller are shown with the same numbers. 1, 2, 3: Microphones; 4, 5: Wheels; 6: Speaker; R: Random input.

implementation, these clients have the fitness evaluation component embedded within them to simplify the communication. Yet, once the clients are installed, the fitness evaluation component of the system can be updated using *scp* (secure copy) utility from the server.

# 4 Robot Structure

The robots used in this study have two wheels, one omni-directional speaker, and three uni-directional microphones which can receive sounds emitted by other robots within a certain range and in a $120\,\mathrm{deg}$ region. Their controller is a single-layer neural network which takes microphone data as input and gives out motor wheel speeds and a speaker signal which is either on or off. The controller also has a probabilistic input implemented as a random number generator which causes dynamic and nondeterministic global behaviors in robot groups. The robot model and the controller can be seen in Figure 2.

# 5 Conclusion

In evolutionary robotics, the fitness of an evolved controller is evaluated by running simulated robots with the controller on physics based simulations, which provide realistic physics and therefore ensure the effectiveness of the evolved controller on physical robots. However they are costly in terms of processing power, which affects the level

of their utility considerably. To overcome the processing demand problems, Parallel Evolution System (PES) is developed to parallelize the genetic algorithm.

The simulations in this study are done using Open Dynamics Engine (ODE) which is an open-source 3D dynamic physics simulator.

The focus of this thesis is on evolving controllers for pattern formation and self-assembly tasks. Because of the probabilistic inputs, the overall behavior of robot groups shows interesting variations from normally expected behaviors that cause local minimum points in fitness space. The probabilistic inputs help the group of robots get out of local minima and find a configuration with a higher fitness.

For example, consider the clustering problem for robots. When probabilistic inputs are not used, the robots tend to form and maintain small clusters. When probabilistic inputs are utilized, robots seem to have a tendency to leave their current cluster with a certain probability and look for other robot clusters. Eventually, this causes formation of larger and larger clusters, and finally one big cluster as desired. So, probabilistic inputs can improve performance of reactive robot systems and will be utilized also in this thesis.

So far, the development of PES and most of the simulator is completed. The work to be done includes extension of robots in the simulator to attach to other robots, experimenting with several pattern formation and attachment tasks, and trying different uses of probabilistic inputs.

# References


[1] Belta C. and Kumar V. Trajectory design for formations of robots by kinetic energy shaping. In *Proceedings. ICRA '02. IEEE International Conference on Robotics and Automation*, 2002.

[2] Dudenhoeffer D.D. and Jones M.P. A formation behavior for large-scale micro-robot force deployment. In *Simulation Conference Proceedings*, 2000.

[3] Goldberg D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. MA: Addison-Wesley, 1989.

[4] Dudek G., Jenkin M., and Milios E. *A taxonomy for multi-agent robotics*. A K Peters Ltd, 2002. Chapter 1.

[5] Fredslund J. and Mataric M.J. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[6] Kowalik J. *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press Scientific and Engineering Computation, 1994.

[7] Desai J.P. Modeling multiple teams of mobile robots: a graph theoretic approach. In *Proceedings. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[8] Fujibayashi K., Murata S., Sugawara K., and Yamamura M. Self-organizing formation algorithm for active elements. In *Proceedings. 21st IEEE Symposium on Reliable Distributed Systems*, 2002.

[9] Sugihara K. and Suzuki I. Distributed algorithms for formation of geometric patterns with many mobile robots. *J. Robot. Syst.*, 13(3):127–139, 1996.

[10] Egerstedt M. and Hu X. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001.

[11] Mataric M. and Cliff D. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 13(3):67–83, 1996.

[12] Kostelnik P., Samulka M., and Janosik M. Scalable multi-robot formations using local sensing and communication. In *RoMoCo '02. Proceedings of the Third International Workshop on Robot Motion and Control*, 2002.

[13] Fierro R. and Das A.K. A modular architecture for formation control. In *RoMoCo '02. Proceedings of the Third International Workshop on Robot Motion and Control*, 2002.

[14] Hempel R. The mpi standard for message passing. In Gentzsch, Wolfgang, and Harms, editors, *High-Performance Computing and Networking, International Conference and Exhibition, Proceedings, Volume II: Networking and Tools*, volume 797, pages 247–252, 1994.

[15] Carpin S. and Parker L.E. Cooperative leader following in a distributed multi-robot system. In *Proceedings. ICRA '02. IEEE International Conference on Robotics and Automation*, 2002.

[16] Balch T. and Hybinette M. Behavior-based coordination of large-scale robot formations. In *Proceedings. Fourth International Conference on MultiAgent Systems*, 2000.

[17] Balch T. and Hybinette M. Social potentials for scalable multi-robot formations. In *Proceedings. ICRA '00. IEEE International Conference on Robotics and Automation*, 2000.

[18] Koo T.J. and Shahruz S.M. Formation of a group of unmanned aerial vehicles (uavs). In *Proceedings of the American Control Conference*, 2001.

[19] Kowalczyk W. Target assignment strategy for scattered robots building formation. In *RoMoCo '02. Proceedings of the Third International Workshop on Robot Motion and Control*, 2002.

[20] Cao Y., Fukunaga A. S., and Kahng A. B. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, 1997.

[21] Cao Z., Tan M., Wang S., Fan Y., and Zhang B. The optimization research of formation control for multiple mobile robots. In *Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002.