Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

# Coordination-based Cooperation Protocol in Multi-agent Robotic Systems

Fang-Chang Lin and Jane Yung-jen Hsu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

## Abstract

*This paper proposes a coordination-based cooperation protocol for the Object-Sorting Task in multi-agent robotic systems. The protocol coordinates the agents to move objects to destinations efficiently and effectively. Every agent autonomously makes subjective optimal decision, then the coordination algorithm resolves their conflicts by considering the global results. Since every agent runs the same algorithm to obtain the common results without further communication, the protocol is efficient. Implementation of the protocol is realized on a distributed modularized agent architecture. Experimental results shows that the protocol is effective and better than a previous proposed help-based protocol. In addition, its performance very closes to a deliberate but high complexity model, genetic algorithms. The protocol can be applied to coordinate multiple autonomous robots for dispatching and transporting components in manufacturing systems.*

## 1. Introduction

Parallel performance and the cooperation requirements of tasks are the main interests of the emerging multi-agent robotic systems.

Some tasks can be executed in parallel for better performance. Tasks such as painting a wide wall or cleaning rooms can be partitioned into several subtasks, each of which is then assigned to one agent. If all agents work independently, i.e. no resource contention or goal conflicts, the performance is linearly speeded up. Otherwise, if there is resource contention (e.g. short of paintbrushes or brooms) or conflicts (e.g. agents have different favorite colors), the performance speedup will be sublinear. On the other hand, some other tasks require explicit cooperation among the agents, e.g. conferences, moving a heavy equipment, etc. A specific multi-agent task, the Object-Sorting Task, described in this paper consists of both parallelism and cooperation characteristics in which explicit cooperation is required to accomplish the task.

There has been much research in multi-agent robotic systems. Some of them proposed mechanism to create the foundation of multi-agent robotic systems. For example, Fukuda's CEBOT system [5] showed the self-organizing behavior of a group of heterogeneous robotic agents; Asama et al. proposed the ACTRESS architecture [4] for connecting equipment, robots and computers together to compose autonomous multi-agent robotic systems by designing underlining communication architecture; Wang proposed several distributed functional primitives for distributed robotic systems [11]. Other researchers worked on solving multi-agent tasks, e.g. Mataric addressed the problem of distributing a task over a collection of homogeneous mobile robots [9]; Arkin et al. assessed the impact on performance of a society of robots in a foraging and retrieval task when simple communication was introduced [2,3]; Alami et al. coordinated the multiple-robot navigation with a plan-merging paradigm for the task of transporting containers in harbors [1]; Lin and Hsu provided a fully distributed cooperation protocol for the Object-Sorting Task in multi-agent robotic systems [7,8].

To solve a multi-agent task, either centralized or distributed approaches can be employed. A centralized model use a powerful agent to plan and schedule the subtasks for every agent. This control agent has global knowledge concerning the environment and the problems. It can deliberately plan for performance, e.g. optimal solutions. However, for tasks with NP complexity, the centralized approach is impractical. Furthermore, the control agent must be powerful enough to achieve satisfactory performance. High design complexity, high cost and low reliability are the other drawbacks of centralized approaches.

On the other hand, a distributed approach decreases design complexity and cost, while increasing the reliability. Agents are autonomous and equal. An agent plans for itself and communicates with the others in order to accomplish the global task. It is reactive because every agent interacts directly with the environment. However, each agent has only local knowledge about the task and the environment. Hence, it cannot make the best decisions alone. Furthermore, negotiation or social cooperation rules for conflict resolution are required to coordinate among them.

To attack the drawback of local knowledge so as to obtain better performance, a *coordination-based cooperation protocol* (CCP) has been developed. It is better than the *help-based cooperation protocol* (HCP) [7]. The Object-Sorting Task (OST) is used to demonstrate the approach.

Section 2 introduces and discusses the OST, and summarizes the HCP. The CCP and its implementation are described in Sections 3 and 4. Simulation and experimental results are shown in Section 5.

## 2. The Object-Sorting Task (OST)

The OST was formally defined and discussed in [8], which showed that the OST is a NP-complete problem for finding the optimal performance. This section addresses its definition, problems, and a summary for the HCP.

### 2.1 Definition

Let $O=\{o_1,\ldots,o_M\}$ be a set of stationary objects that is randomly distributed in a bounded area. Every object, $o_i=(l_i,d_i,n_i)$, is associated with an initial location $l_i$, a destination location $d_i$, and the number $n_i$ of agents for movement. An object $o_i$ can be moved only if there are at least $n_i$ agents available to move it. Let $R=\{r_1,\ldots r_N\}$ be the set of agents and $n_{max}$ be the maximal number of agents to move any single object, an object-sorting task can be completed only if $N$ is not less than $n_{max}$. Agents search for objects and move them to their destinations. When all the objects have been moved to their destinations, the task is finished.

Typical application examples of OST are foraging and retrieval, explosives detection and handling, AGV dispatching for components transportation in FMS, surveillance, etc.

Finding an optimal solution of OST is a high complexity problem. Algorithm *Optimal-OST* describes this search process and its space. It lists all object sequences, then assign agents to the objects for each sequence. There are $M!$ object sequences from $M$ objects, and $C(N,n_i)$ agent assignments for each object $o_i$, where $C(k,r)$ is the number of $r$-combinations from $k$. Hence, the search space is $M!(\Sigma C(N,n_i))$, $1 \le i \le M$. Let $E(C(N,i))$ be the mean of $C(N,i)$, $1 \le i \le N$. The search space becomes $M!(E(C(N,i)))^M$.

**Algorithm** *Optimal-OST.*

```
1. FOR each permutation of objects $o_1,\ldots,o_M$ DO
   (1) Let the object sequence be $s_1,\ldots s_M$.
   (2) FOR each $s_i=(l_i,d_i,n_i)$ in $s_1,\ldots s_M$ DO
      a) Let Comb(r) represent all the r-combinations from
         the agent set $\{r_1,\ldots r_N\}$.
      b) FOR each combination of Comb($n_i$) DO
            Assign the agents to the object $s_i$.
   (3) Calculate the cost of $s_1,\ldots s_M$.
2. The optimal solution is the object sequence combined
   with the assigned agent, which has minimal cost.
```

In this research, the following assumptions were made. The agents are homogeneous mobile robots with the basic capabilities for navigation, obstacle avoidance, object recognition and object handling. The agents have no prior knowledge about the environment, nor the other agents. Finally, the agents communicate with the others by broadcast or point-to-point messages.

There are many operation models associated with different strategies to do the task, e.g. an object is assigned to the agents at random, movement of the objects are controlled by a predefined precedence, actions of the agents and objects are all central-controlled, etc. The performance evaluation is based on the time cost to accomplish the task. Generally speaking, an agent searches for objects, coordinates its object schedule with the other agents, and cooperates with the other agents to move objects. So, an agent spends its cost on search, coordination and cooperation. The cost of the applied operation model is the maximum cost among all agent's costs.

### 2.2 Problems

To solve the OST, several problems need to be addressed .

1. *Search.* All the objects must be found in order to attack the task. The most intuitive way is to let agents search the entire area so that they can find all the objects. How do they search efficiently?
2. *Coordination.* How do the agents coordinate for assigning themselves to a found object? That is, for a given object, which agents should work together to move it ? and who makes the decision?
3. *Deadlocks.* When each agent autonomously selects an object and none of the selected object has enough agents for movement, a deadlock occurs.
4. *Termination* . How do every agent realize the global task has been accomplished?

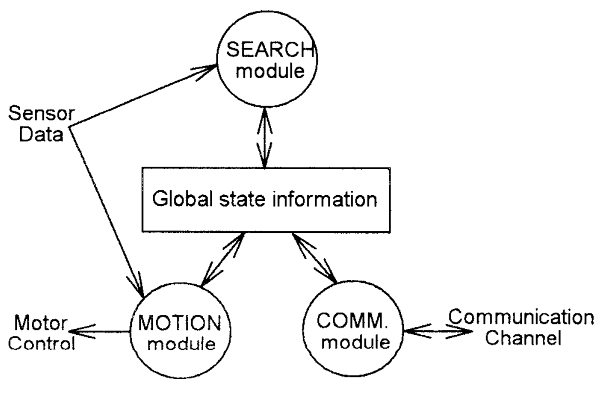

**Fig. 1**: Agent architecture of the Object-Sorting Task

### 2.3 Help-based Cooperation Protocol (HCP)

Our previous work proposed the *help-based cooperation protocol* for the OST [7]. The protocol was implemented in a modularized, reactive agent architecture as showed in Fig. 1. The search module, communication module and motion module are all finite state automata. They share the global state information and change the state information according to their state functions. The search module searches and identifies objects. The motion module performs the function of object movement alone or with other agents. The communication module communicates with the other agents in order to cooperate with them.

The working area is partitioned into disjoint $N$ subareas, and each subarea is assigned to an agent. Each agent exhaustively searches its subarea, requests help from the others once it has found a large object, and selects its partners. After there are enough agents arriving at the found object, they carry the object to its destination. The cooperation protocol defines how and when an agent requests help, how and when the other agents offer help, and deadlock handling in order to coordinate the agents for accomplishing the task.

With this approach, the objects are partitioned into several parts. Each part of the objects is treated by a specified agent that requests help and determines which agents will offer help and become its partners so as to move every object in this part of objects. Since each agent is autonomous, simultaneous selection of partners by several agents may cause a deadlock. Several deadlock handling schemes were also provided. The protocol was realized into each agent's state transition function so that agents reacted and cooperated quickly.

The above approach utilized simultaneous subarea search for parallel performance, and cooperation protocol for solving the coordination, deadlocks, and termination problems. Nevertheless, the performance of OST can be further improved. Subtask assignment and help-based strategies forced agents to request help and wait once finding a large object, which limited the global knowledge of agents. Hence, agents cannot make cost-optimal decisions. The CCP has been developed to address this issue.

## 3. Coordination-based cooperation protocol

As the HCP, the working area is equally partitioned into disjoint subareas for parallel search. Instead of requesting help once an object is found, the object information is broadcast to the others. Instead of partitioning the objects into subtasks, only *search* task is partitioned for parallel performance. With the overall object information, each agent makes its subjective optimal schedule and negotiates with the others for a global schedule. In order to efficiently resolve conflicts, social rules are employed into a coordination algorithm from which decisions are obtained by each agent independently so as to minimize negotiation overhead.

### 3.1 Coordination algorithm

Coordination among agents are built by coordination strategies. Since the OST is a NP-complete problem, there is no polynomial time algorithm for overall optimal object schedule. The utilized strategies are based on balanced cost. Algorithm *Coordination* depicts them. All agents invoke this algorithm to negotiate their individual object schedules.

At first, every agent selects a locally cost-optimal object and broadcasts its selection. Among the selected objects, the best cost-optimal object is scheduled, i.e. enough agents are assigned to it. Finally, every assigned agent updates its cost and location. The process is repeated until all objects are scheduled.

**Algorithm** *Coordination.*
```
1. Every agent selects its current cost-optimal object, and
   broadcasts to the others.
   Let the selected objects be $s_1, \ldots, s_k$ , $1 \leq k \leq N$.
2. Select the best cost-optimal object, Opt, among the se-
   lected objects.
   (1) FOR each object in $s_1..s_k$ DO
       • Let the current object be $o_i=(l_i,d_i,n_i)$.
       • FOR each agent $r_j$ DO
           Let $CA_j$ be the accumulated cost of $r_j$ , and
           $CR_j$ be the cost of $r_j$ to reach $o_i$.
           $C_{ij} = CA_j + CR_j$
       • $M_i$ = the $n_i$-th smallest cost from $C_{i1},C_{i2},\ldots, C_{iN}$
   (2) Opt = the object with minimum $M_i$
3. Assign agents to Opt, and update data.
   Let Opt be $o_p=(l_p,d_p,n_p)$.
   Assigned agents are the $n_p$ agents with smaller cost $C_{pi}$,
   $1 \leq i \leq N$.
   IF I am one of the assigned agents THEN
     (1) append Opt to my object schedule
     (2) accumulate my cost with ( $M_p$ + the cost from $l_p$ to
         $d_p$)
     (3) my location = $d_p$
4. Repeat from step 1 until all object has been scheduled.
```

Obviously, this approach is cost balanced because the selection strategy is to choose the agents with smaller cost. Besides, deadlock-free is guaranteed by the two facts: there is a consistent object order in every agent's object schedule and there are enough agents assigned to every object.

The number of selected objects is at most $N$ because every agent select one object for negotiation at a time. For each selected object, it requires time complexity $O(N)$ to calculate all costs of agents. So, the cost for scheduling an object is $O(N^2)$. Since there are $M$ objects, the complexity of algorithm *Coordination* is $O(MN^2)$.

There are several differences between HCP and CCP.

1. Task partition: HCP partitions objects and assign each part to an agent while CCP partitions the *search* task only.
2. Object assignment: An object is assigned to which agents is determined by request/offer protocol in HCP while by coordination protocol in CCP.
3. Object found: An agent deals with an object at a time (request help, select partners, then move object) in HCP while an agent broadcasts the object information in CCP when an object is found.
4. Knowledge: HCP makes decision with local knowledge other than global knowledge in CCP.
5. Object : CCP requires every agent keeps all the object data while HCP does not.

## 3.2 Object identification

In order to clearly identify objects, a consistent representation of objects is necessary during negotiation. In general, assigning an object a unique object ID is appropriate. However, various assignment methods should be suitably applied under different situation. Several methods are discussed:

1. Predefined attributes: Object IDs are predefined so that it can be identified by agents. An object can be identified from its outside look, size, symbol, or even initial location. For example, a two dimensional coordinate can determine the order of object IDs by comparing x-coordinate first then y-coordinate.
2. Coordinator: Where there is no way to specify predefined object IDs, this method can be applied. Every object found is reported to the coordinator, then the coordinator assigns its ID and broadcasts to the others. A trivial way is that object IDs are assigned with increasing sequence.
3. The distributed algorithms [6,10] concerning the order of discrete events may be employed.

## 3.3 A specific agent as the coordinator

The coordination algorithm can be implemented only on a dedicated agent because every agent runs the same algorithm to obtain the object schedule. In this manner, every agent sends its decision to the coordinator instead of broadcast. The coordinator runs the algorithm and sends the results to the others for updating their schedules, costs, and locations.

## 4. Cooperation architecture

The cooperation architecture employed to realize cooperation protocols in multi-agent robotic systems is an agent architecture with embedded cooperation protocol. It is reactive, modularized, and cooperative.

The agent architecture is a finite state automaton (FSA) composed of several functional modules that cooperate and coordinate each other through the state transition function. The FSA implements the cooperation protocol which coordinates the multi-agent robotic system. Every agent working on the architecture cooperates and coordinates with the others to achieve common goals. Every functional module is also a FSA which is a subset of the global FSA. Modularized decomposition simplifies the design complexity of each module.

The agent architecture of the OST is showed in Fig. 1 which have been utilized by HCP and CCP approaches. The transition diagram of the FSA of CCP is showed in Fig. 2 in which circles represent states and arrows represent transitions. At first, agents start searching in SEARCHING state, broadcast found object data. After finishing search, agents enter SELECTING state, exchange local selections, coordinate to obtain the overall object schedule. In DECISION state, every agent picks up its object tasks according to the scheduled object sequence and changes to HELPING state for going toward the target object. When an agent arrives at the target object, its state becomes WAITING. If there are enough agents for object movement, they enter MOVING state and carry the object to its destination. They each repeat the task cycle until all assigned object tasks having been accomplished. Agents enter FINISH state then.

The global FSA is further decomposed into three modularized FSAs. The three functional modules execute concurrently and take care only its own subset transition functions. Fig. 3 shows their transition diagram.

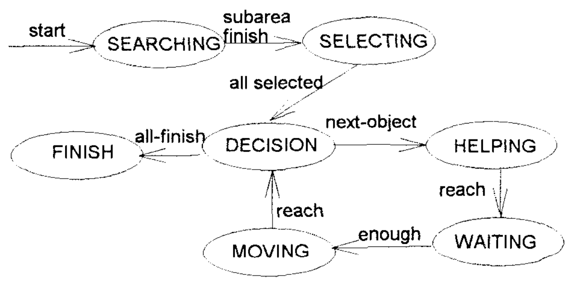

**Fig. 2**: State transition diagrams of coordination-based cooperation protocol

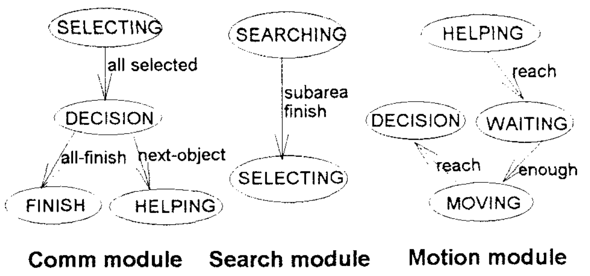

**Fig. 3**: State transition of functional modules

## 5. Simulation

The object-sorting task was simulated in the simulator developed on Sun workstation with graphic user interface showing task execution. The simulator is a testbed for testing different operation models on the object-sorting task. Fig. 4 shows the utilized experimental map.

The performance was evaluated with the number of time steps. The following code fragment sketches the actions performed by the agent at each time step.

```
for each agent $i$ ,
        do the search module of agent $i$;
        do the motion module of agent $i$;
        do the comm. module of agent $i$.
```

Each object was randomly generated for its destination, initial location, and the number of required agents. This experiment generated 10 sets of objects for each number of objects $M$ using $n_{max}$=10.

The experiment was run by varying the number of agents $N$, and the number of objects $M$. For a given number $M$ of objects, the execution time was the average of the execution time run from the generated 10 object sets of $M$. The actual execution time performed by HCP is represented by H. The time performed by CCP is C. There were two references for comparison, an ideal model as an estimated lower bound, L, and the time, G, obtained from the genetic algorithm developed for the OST [8].

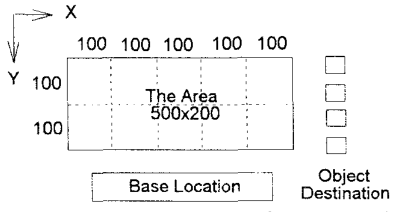

**Fig. 4**: The simulation map for 5x2 partition

The estimated lower time is the execution time under the operation model that the other agents are always available for offering help when an agent finds a large object. The available agents are assumed to be in the central point of their subareas. An agent chooses its partners which are closer to it. The execution time of each agent is its subarea search time and all the processing time for the objects located in its subarea.

Experimental results shows that the CCP has the same features of HCP, i.e. stable behavior and speedup effect. Furthermore, the performance of CCP is better than the HCP.

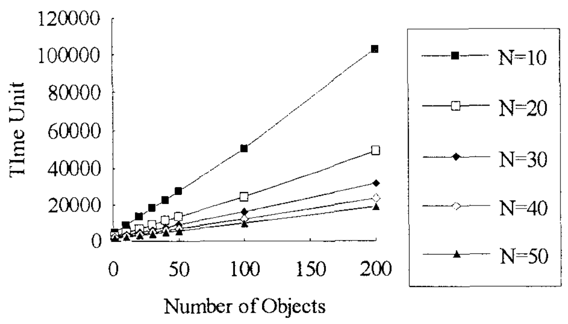

**Fig. 5**: The execution time for different number of objects when N=10,20,30,40, and 50 agents

When the number of objects increases, the execution time increases linearly with it for all $N$. It shows that the CCP is very stable under different workloads. When the generated object sets are applied to $N \geq 10$, the execution time decreases with the increasing number of agents as showed in Fig. 5.

Table 1 lists the speedup ratio relative to the execution time unit of $N$=10. For most cases with $M$>40, the speedup is linear or superlinear. There is obvious improvement on objects processing time because more agents make more objects movement in parallel. It shows that the protocol can effectively utilize the increased agent-power. On the other hand, for small number of objects, e.g. $M$=1, more agents decrease *search* time due to smaller subarea. But, there is no obvious improvement on object processing time since there is no significant increasing parallelism. Hence, the results provide a useful reference when adding more agents for speedup. Where there is a superlinear speedup, adding more agents can be beneficial. In contrast, under sublinear speedup condition, the benefit of adding more agents is not as significant.

Fig. 6 compares the four models: HCP, CCP, estimated lower bound, and GA under different number of objects for $N$=10 agents. The CCP is better than the HCP. Furthermore, its results very close to the GA. GA is a centralized, high computation complexity, and time consuming search process. On the contrary, CCP uses a distributed model to provide efficient reactivity and good performance which very close to the deliberate approach. The experimental shows that the performance of CCP is better than the HCP as well when $N$=20,30,40, and 50 agents.

**Table 1**: The speedup effect. Each table entry is the ratio= (execution time unit of each $N$=10, 20, 30, 40, and 50) / (execution time unit of $N$=10 agents).

| #objects | Number of agents, $N$ | | | | |
|---|---|---|---|---|---|
| $M$ | 10 | 20 | 30 | 40 | 50 |
| 1 | 1 | 1.62 | 2.07 | 2.40 | 2.66 |
| 10 | 1 | 1.86 | 2.47 | 3.07 | 3.76 |
| 20 | 1 | 2.02 | 2.85 | 3.57 | 4.26 |
| 30 | 1 | 2.03 | 2.98 | 3.79 | 4.51 |
| 40 | 1 | 1.98 | 2.93 | 3.82 | 4.53 |
| 50 | 1 | 2.05 | 3.01 | 3.95 | 4.83 |
| 100 | 1 | 2.06 | 3.11 | 4.14 | 5.08 |
| 200 | 1 | 2.09 | 3.24 | 4.34 | 5.38 |

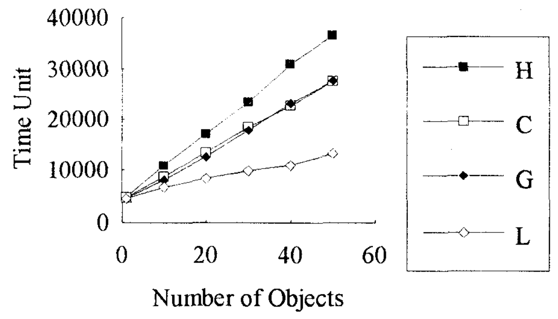

**Fig. 6**: Performance comparison of four models when N=10 agents. H: help-based protocol, C: coordination-based protocol, G: genetic algorithm, and L: estimated lower bound.

## 6. Conclusion

In this paper, we provided a coordination-based protocol for coordinating multi-agent robotic systems to do the Object-Sorting Task. The protocol has stable behavior and speedup effect. Furthermore, its performance is better than the help-based protocol and very close to the genetic algorithm which requires high computational complexity to search for optimal solutions.

By combining the centralized and distributed approaches, this paper has demonstrated a model for multi-agent tasks that offers efficient reactivity as well as comparable performance with a global search algorithm.

The CCP requires every agent to have complete information about all objects. The storage demand expands with the number of objects. In fact, the coordination for a global object schedule can start during subarea search. Coordination can be activated by a certain number of objects found, area searched, or the coordinator. Performing coordination in smaller batches can reduce storage requirements.

## 7. References

[1] R. Alami, F. Robert, F. Ingrand, and S. Suzuki, "Multi-robot Cooperation through Incremental Plan-Merging", *Proc. of IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 1995, pp. 2573-2579.

[2] R.C. Arkin, T. Balch and E. Nitz, "Communication of Behavioral State in Multi-agent Retrieval tasks", *Proc. of 1993 IEEE International Conference on Robotics and Automation*, GA, May 1993.

[3] R.C. Arkin and J.D. Hobbs, "Dimensions of Communication and Social Organization in Multi-Agent Robotic Systems", *Proc. Simulation of Adaptive Behavior 92*, Honolulu, HI, Dec. 1992.

[4]. H. Asama, A. Matsumoto, and Y. Ishida, "Design of an Autonomous and Distributed Robot System: ACTRESS", *Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems '89*, Tsukuba, Japan, Sep. 1989, pp. 283-290.

[5] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Structure Decision Method for Self Organizing Robots Based on Cell Structure - CEBOT", *Proc. of IEEE International Conference on Robotics and Automation*, Scottsdale Arizona, 1989, pp. 695-700.

[6] L. Lamport, "The Mutual Exclusion Problem: Part II - Statement and Solutions", *JACM*, Vol. 33, No. 2, Apr. 1986, pp. 327-348.

[7] F.C. Lin and J.Y.-j. Hsu, "Cooperation and Deadlock-Handling for an Object-Sorting Task in a Multi-agent Robotic System", *Proc. of IEEE Inter. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995, pp. 2580-2585.

[8] F.C. Lin and J.Y.-j. Hsu, "A Genetic Algorithm Approach for the Object-Sorting Task", *Proc. of IEEE International Conference on Systems, Man, and cybernetics*, Vancouver, Canada, Oct 1995.

[9] M. Mataric, "Minimizing Complexity in Controlling a Mobile Robot Population", *Proc. of 1992 IEEE International Conf. on Robotics and Automation*, Nice, 1992, pp. 830-835,

[10] J. Wang, "Establish a Globally Consistent Order of Discrete Events in Distributed Robotic Systems", *Proc. of 1993 IEEE International Conference on Robotics and Automation*, GA, May 1993, pp. 853-858.

[11] J. Wang, "Operating Primitives Supporting Traffic Regulation and Control of Mobile Robots under Distributed Robotic Systems", *Proc. of IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 1995, pp. 1613-1618.