A Genetic Algorithm Approach for the Object-Sorting Task Problem

Fang-Chang Lin and Jane Yung-jen Hsu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.
fclin@robot.csie.ntu.edu.tw yjhsu@csie.ntu.edu.tw

ABSTRACT

Most multi-agent tasks have high complexity. Optimal solutions of them are nearly intractable because of the large complexity. One approach is to use stochastic search techniques such as genetic algorithms to explore the solutions by its implicit parallelism and genetic mechanism. This paper analyzes the complexity of the Object-Sorting Task, shows its NP-completeness, and develops a genetic algorithm to explore the optima. Experimental results show that 1)GA can find an optimal solution quickly for simple problem instances. 2)the results are better than our previous proposed cooperation protocol approach. In addition, the results can serve as a reference foundation of OST to the other approaches.

1. INTRODUCTION

In a multi-agent robotic system, some tasks can be achieved by a single agent, e.g. cleaning up a region, searching for a target in an area, etc. The system performance can be improved if the task can be partitioned into many subtasks and let the agents do the subtasks in parallel. Meanwhile some tasks can only be done by a single agent, e.g. moving an object to cross a single-plank bridge on which only one agent is allowed at a time. However, many other tasks in multi-agent systems require cooperation among the agents and cannot be done by one agent along, e.g. moving a large object which is not movable by any single agent along. These tasks are *multi-agent tasks*.

There is much research on multi-agent robotic systems. Fukuda's CEBOT system[6] demonstrated the self-organizing behavior of a group of heterogeneous robotic agents. Beni and Hackwood's research on swarm robotics demonstrated large scale cooperation in simulation[8]. Brooks et al.[4] developed the lunar base construction robots by using a set of reactive rules. Mataric[12] also studied task performance in a group of mobile robots based on the subsumption architecture[5]. Arkin has demonstrated that cooperation between robotic agents is possible even in the absence of communication[1,2]. Wang tried to solve the problems of multi-agent robotic systems by developing the techniques of Distributed Robotic Systems[16]. Asama et al. proposed the ACTRESS[3] system for distributed robotic systems. Our previous work provided a fully distributed cooperation protocol for the Object-Sorting Task (OST) in a multi-agent environment[11,12].

Although there has not been much work on multi-agent task analysis, it is helpful for designing multi-agent systems through the analysis of multi-agent tasks. The analysis results may provide concrete references for multi-agent system developers. Furthermore, the reference low bound performance in the cooperation protocol approach for OST is an estimation based on an ideal model which may be unrealistic. Therefore, it is better to analyze the complexity of OST and develop another concrete reference foundation for OST so that every approach can compare with it.

This paper analyzes the OST by defining the Object-Sorting Task Problem (OSTP) and showing its NP-completeness. OSTP is defined to find an optimal solution of the OST. For these high complexity problems, traditional approaches are computationally expensive and intractable. Another approach is to use stochastic search techniques such as genetic algorithms (GA) or simulated annealing to explore the optima. GA explores the search space based on the mechanics of natural selection and natural genetics[7]. GA is selected to solve the OSTP in this paper because of its implicit parallelism. A genetic algorithm for OSTP is developed in this paper. The feasibility algorithm developed to assist the NP-completeness proof can be modified to implement the evaluation function of GA. Only a small number of population size and generation were used to explore the large search space. However, the results is satisfied. Experiment shows the results are better than cooperation protocol approach. Besides, the results can serve as a reference of OST.

Section 2 defines the object sorting task. Its complexity analysis is shown in Section 3. Genetic algorithm approach and its experiments are presented in Section 4 and 5. Finally, conclusions and future work are discussed in Section 6.

2. OBJECT-SORTING TASK

One kind of *multi-agent task* is the object-sorting task which is defined below. Cooperation among the agents is explicitly required in order to move the objects and accomplish the task.

Definition: The Object-Sorting Task (OST).
OST=(O,R) is an object sorting task. Let $O=\{o_1,...,o_M\}$ 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 moving it. An object o_i can be

0-7803-2559-1/95 $4.00 © 1995 IEEE

moved only if there are at least n_i agents available to move it. An agent can realize the destination of an object only when it reaches the object. Let $R=\{r_1,...r_N\}$ be the set of agents and $I(r_i)$ be the initial location of r_i. Let 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}. When all the objects have been moved to their destinations and all the agents have returned to their initial locations, the task is done.

In the beginning, each agent starts traveling from its initial location. After an agent has reached an object, the agent and its partners can move the object to its destination if there are enough agents to do that. Otherwise the agent can stay by the object and wait its partners or it can continue traveling. After the object has been moved to its destination, these agents can continue traveling to the other objects. Finally, the agents return to their initial locations.

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. No matter what operation model and strategies are arranged, its solution can be represented by the sequence of objects moved by each agent. This sequence is called *Agent-Object Sequence* (AOS). For each agent r_i, Let $s_{ij} \in O$ represent the j-th object moved by r_i (and its partners, if necessary). An AOS is represented by the form:

$$r_1 : s_{11}s_{12}...s_{1k_1}$$
$$r_2 : s_{21}s_{22}...s_{2k_2}$$
$$...$$
$$r_N : s_{N1}s_{N2}...s_{Nk_N}$$

For example, the number of objects that r_1 has involved to move is k_1 and their sequence is $s_{11}s_{12}...s_{1k_1}$. Let $I(s_{ij})$ be the initial location of s_{ij}, and $D(s_{ij})$ be the destination of s_{ij}. The cost of s_{ij} is defined as

$$c_{ij} = a_{ij}+w_{ij}+m_{ij}$$

where

a_{ij} : the cost of r_i moving from $I(r_i)$ to $I(s_{i1})$ when j=1, or the cost of r_i moving from $D(s_{i(j-1)})$ to $I(s_{ij})$ when $1<j\le k_i$.

w_{ij} : the cost of r_i after reaching $I(s_{ij})$ and waiting for enough agents to move s_{ij}.

m_{ij} : the cost of r_i moving s_{ij} from $I(s_{ij})$ to $D(s_{ij})$.

All a_{ij}, w_{ij}, and m_{ij} are defined by its operation model and strategies, e.g. central model may use distance to define them. Additional search cost is combined into a_{ij} if the agents must search for the objects. Let f_i be the cost of r_i moving from $D(s_{ik_i})$ to $I(r_i)$, i.e. return to its initial location. The cost of agent r_i is defined as

$$C_i = (\Sigma_j\ c_{ij}) + f_i$$

For example, C_1 is the cost of r_1 spent on its k_1 objects and returning to its initial location. Obviously, the maximum C_i ($1\le i\le N$) is the maximum cost to achieve the AOS. Hence, the cost of the AOS is defined as

$$C = \text{Max } C_i,\ 1\le i\le N$$

3. COMPLEXITY OF OSTP

The OSTP is defined to analyze the time complexity of the OST. An OSTP instance is represented by OSTP=(O,R,K). First we show OSTP is in NP, then show that a well-known NP-complete problem which is Traveling Salesman Problem (TSP)[9] can be reduced to OSTP. That is, the OSTP is NP-complete.

Object-Sorting Task Problem (OSTP)

Given an object sorting task OST=(O,R) and a constant K. Is there a cost $C \le K$?

Traveling Salesman Problem (TSP)

Given a set of cities $V=\{s, v_1, .., v_m\}$, and $Dist(v_i,v_j)$ is the cost traveling from v_i to v_j, $v_i,v_j \in V$. A tour is a path starting from s and going through all other cities in V exactly once and returning to s. Is there a tour whose cost at most a given constant K ?

3.1 OSTP is in NP

The data structure used by Algorithm 1 is defined as follows:

1. Each object o_i is represented by an object record:
   - state: state value can be one of {*stay*, *reach*},
   - l_i, d_i, *and* n_i : referred to $o_i=(l_i,d_i,n_i)$,
   - wait_queue: a queue recording the agents waiting on the object.
2. Each agent r_i is represented by an agent record:
   - sequence of moved objects: $s_{i1}s_{i2}...s_{ik_i}$
   - cost: accumulated cost.
3. Event queue: a list sorted on the key *cost* with non-decreasing order, each record contains the following:
   - type: event type, can be one of {*RO*(reach object), *RD*(reach destination), *RH*(reach home)},
   - cost: sort key,
   - agent, object: the associated agent and object of this event.

Algorithm 1. Feasibility of *Agent-Object Sequence*.

Input: OST=(O,R), an AOS and the associated definitions for all a_{ij}, w_{ij}, m_{ij} and f_i.

Output: Feasibility of the AOS, and its cost C if it is feasible.

STEP:

1. Initialize the values of all agent records and object records.
   (1) For each object record:
   state := *stay*, wait_queue := Null.
   (2) For each record of agent r_i : cost:= a_{i1}.

2. For each object o_i :

IF the total number of agents involved to move it < n_i ,

THEN the solution is infeasible, return.

3. For each agent r_i :

Enqueue a *RO* event of r_i to reach object s_{i1} with cost := a_{i1}.

4. Event handling.

WHILE event queue is not empty DO

(1) Get the first event *e*, and let system cost C := *e*.cost.

(2) CASE *e*.type = *RO* : (agent r_i reaches object o_j.)

Enqueue r_i into o_j.wait_queue.

IF all the agents involved to move the object have arrived THEN

For each agent r_i in o_j.wait_queue

(a) Accumulate w_{ij} and m_{ij} to r_i.cost.

(b) Enqueue a *RD* event for r_i with cost := C + m_{ij}.

(3) CASE *e*.type = *RD* : (agent r_i reaches destination of object o_j.)

o_j.state := *reach*.

IF o_j is not the last object in r_i's object sequence THEN

(a) Let s_{ij} be the next object in r_i's object sequence.

(b) Accumulate a_{ij} to r_i.cost.

(c) Enqueue a *RO* event for r_i to reach s_{ij} with cost:= C + a_{ij}.

ELSE

(a) Accumulate f_i to r_i.cost.

(b) Enqueue a *RH* event for r_i with cost:= C + f_i.

(4) CASE *e*.type = *RH* :

Do nothing.

5. For each object o_i DO

IF o_i.state $\neq$ *reach* THEN the solution is infeasible, return.

6. The solution is feasible, and its cost is C.

Given an AOS and associated cost functions, Algorithm 1 will determine its feasibility and its cost if it is a feasible solution. Algorithm 1 uses an event queue to simulate the performance of the input AOS. Every entry of the event queue contains event type, *cost*, and associated agent and object. The event queue is sorted on *cost* with nondecreasing order. Event type can be an agent have reached an object's initial location(*RO*), an agent have reached an object's destination(*RD*), or an agent have returned to its initial location(*RH*). When a *RO* event occurs, the agents can move the object if all the agents involved to move the object have arrived, then the algorithm adds their *RD* events to the event queue. Otherwise, the agent must wait. When a *RD* event occurs, the agent continue approaching to its next object, i.e. adding a *RO* event. If that object is its last object, the agent will return to its initial location, i.e. adding a *RH* event. Because an agent has at most *M* objects to move, i.e. $k_i \leq M$, and each object has 2 events (*RO* and *RD*) for the agent. The maximal number of event is *MN* for *RO* and *RD* each, and *N* for *RH*. It is obviously that the time complexity of one event loop is O(*N*). So, the time complexity for event handling is $O(MN^2)$. The other steps of Algorithm 1 are also bounded by $O(MN)$. So Algorithm 1 is a polynomial time algorithm.

Algorithm 2. Nondeterministic algorithm for OSTP.
Input: An Object Sorting Task Problem OSTP=(*O*, *R*, K).
Step:

1. Randomly generate an AOS for the OSTP.

$r_1 : s_{11}s_{12}\cdots s_{1k_1}$

$r_2 : s_{21}s_{22}\cdots s_{2k_2}$

...

$r_N : s_{N1}s_{N2}\cdots s_{Nk_N}$

And the associated definitions for all a_{ij}, w_{ij}, m_{ij} and f_i .

2. Apply Algorithm 2 to check the AOS. If it is feasible and its cost C≤K, the answer is yes. Otherwise, the answer is unknown.

Algorithm 2 is a nondeterministic algorithm for OSTP. It nondeterministically generates an (AOS) and invokes Algorithm 1 to determine its feasibility and cost in order to answer the question. Because the time complexity of Algorithm 1 is polynomial time, Algorithm 2 is a polynomial time algorithm also. So, OSTP is in NP.

3.2 NP-completeness of OSTP

By showing that TSP can be reduced to OSTP, Theorem 1 proves the NP-completeness of OSTP.

Theorem 1: OSTP is NP-complete.
Proof:

First we show that OSTP is in NP, then prove that TSP can be reduced to OSTP.

1. OSTP is in NP.

Because Algorithm 2 is a NP algorithm for OSTP, OSTP is in NP.

2. TSP can be reduced to OSTP.

For a given TSP instance, construct an instance of OSTP as the following:

(1). $R=\{r_1\}$, $I(r_1)$=s.

(2). For each $v_i \in V$, $1 \leq i \leq m$, construct $o_i=(v_i,v_i,1)$. So $w_{ij}=0$, $m_{ij}=0$ for all s_{ij}.

(3). a_{ij} and f_i are defined by Dist function.

Clearly, there is a C at most K for the OSTP instance iff the TSP instance has a tour whose cost at most K.

4. GENETIC ALGORITHM APPROACH

There are several issues in using GA for OSTP: coding a solution, fitness function, initial population, reproduction, crossover and mutation. These issues are discussed in the following subsections.

This approach guarantees that for each generation, every chromosome in the population is a feasible solution of OST. First, it generates the initial population with all feasible solutions. Next, reproduction, crossover and mutation operators reserve the feasibility. It avoids wasting time on infeasible solutions.

4.1 Coding and fitness

Because each length of object list, k_i, in AOS representation is not a fixed number, it is difficult to operate on AOS and reproduce next generation with all feasible AOS. To generate all legal solutions, it is necessary to code the solution other than AOS. If we list the order that each object arrived to its destination, for example, an object sequence is acquired. In addition, the agents to move an object can be listed also. Another representation for a solution of OST is *Object-Agent Sequence*(OAS) which combines object sequence and serviced agent lists for all objects:

$$s_1 : r_{11} r_{12} \cdots r_{1h_1}$$
$$s_2 : r_{21} r_{22} \cdots r_{2h_2}$$
$$\cdots$$
$$s_m : r_{m1} r_{m2} \cdots r_{mh_m}$$

The object sequence is $s_1 s_2 \ldots s_m$, and $r_{i1} r_{i2} \ldots r_{ih_i}$ is the agent list of object s_i whose required number of agents is h_i. Many criteria can be used to determine an object sequence, here are several examples:

- when an agent selected an object
- when an agent reached an object
- when an object started to move
- when an object reached its destination

The search space of OSTP can be derived from OAS representation. There are M! object sequences from M objects, and $C(N,n_i)$ agent lists for each object o_i, where N is the number of agents and C(n,r) is the number of r-combinations from n. Hence, the search space for OSTP 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 $M!(E(C(N,i)))^M$. Table 1 lists the order of search spaces for different number of objects when N=10 agents.

M	10	20	30	40	50	100
Order	10^{26}	10^{58}	10^{92}	10^{128}	10^{164}	10^{358}

Table 1: The order of search spaces for N=10 agents

The chromosome for OST is represented by OAS such that the first part is an object sequence, the second part is agent lists of objects.

Chromosome:

object sequence	agent list of o_1	agent list of o_2	agent lists $o_3 \cdots o_{m-1}$	agent list of o_m
$s_1 s_2 \cdots s_m$	$r_{11} r_{12} \cdots r_{1n_1}$	$r_{21} r_{22} \cdots r_{2n_2}$	$\cdots$	$r_{m1} r_{m2} \cdots r_{mn_m}$

For example, for the following OAS:

o_2: r_2, r_4

o_1: r_5, r_1, r_3

o_3: r_3, r_4

its corresponding chromosome is : 2 1 3 5 1 3 2 4 3 4.

In order to evaluate a chromosome, an OAS is transferred to a corresponding AOS first. It is accomplished by assigning objects for each agent according to the object sequence in OAS. Next, It is obvious that Algorithm 1 can be invoked to evaluate the cost, C. Finally, because OSTP is a minimization problem, the fitness function is defined as $(1/C)^2$ to fit the characteristics of GA.

4.2 Initial population and reproduction

In order to generate feasible solutions in initial population, an algorithm is applied. At first, generate AOS, then simulate the sequence by Algorithm 1 and generate corresponding OAS. Two methods, random and load-balance, for generating AOS were developed and compared. Load-balance method was better and used in the experiments. Random method assigns n_i agents for o_i by random while load-balancing method selects n_i agents for o_i according to a randomly generated cyclic agent sequence. Here is a sketch of load-balance method:

1. Randomly generate a permutation sequence of agents: $a_1, \ldots, a_n$
2. Randomly generate a permutation sequence of objects: $s_1, \ldots, s_m$
3. For each object s_i, let n be its required number of agents. Assign n agents to the object according to the agent sequence. The agent sequence is referred cyclically.

Because there are enough agents for each objects, no objects cannot be moved due to the lack of agents. Furthermore, there is a consistent object sequence for each agent in the AOS, no deadlocks can occur. So, all individuals in initial population are feasible solutions.

Finally, roulette wheel selection and elitism were used in the experiments for reproduction[7].

4.3 Crossover and mutation

The OAS representation includes two kinds of information, object sequence and agent lists. They are different and must be separately processed when applying GA operators such as crossover and mutation. For object sequence, it is a permutation of the object and traditional bit crossover

operators cannot be applied. There are a few crossover schemes proposed for permutation representation [13,14,17]. This paper utilized a simple crossover operator: partially matched crossover (PMX) [7]. Mutation was applied by swapping any two genes in the object sequence. For agent lists, uniform crossover scheme[15] was applied to each object's agent list. Crossover was carried out by swapping the corresponding agent lists of the two selected chromosome. If a mutation occurred at an object's agent list, a selected agent in the list was replaced with any other agent not in the list.

5. EXPERIMENTS

The GA approach for OSTP was carried out on a 500x200 area in which the objects were randomly generated for 10 agents. Fig. 1 depicts the working environment. Each agent was located in a subarea initially, and returned to its base location after finishing its work. The experiment results were compared with the distributed cooperation protocol approach [10]. Cost is defined as the time to wait or travel on the area. It is proportional to the distance. Search cost of subarea was added into the agent cost because it was included by the cooperation protocol approach.

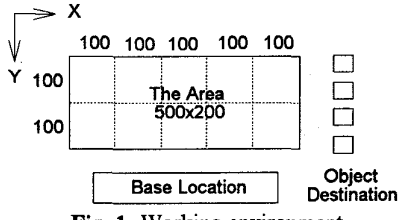

Fig. 1: Working environment

5.1 Parameters

The parameters for the experiments were:

Agent:
- number of agent, N: 10
- initial location: (0,1) relative to its subarea

Object:
- $o_i=(l_i,d_i,n_i)$ parameters were randomly generated.
- number of objects, M: 1, 10, 20, 30, 40 and 50

GA:
- population size: 50
- generation: 50
- crossover rate: 0.6
- mutation rate: 0.02

The search space is very large when $M \geq 10$ from Table 1, it is difficult to choose a large enough population size and the number of generation. Hence, the experiments used fixed population size and generation to explore the effect of the GA approach.

5.2 Results

For each problem instance, GA was run three times. For average performance consideration, only the medium result of the three was taken.

At first, the experiment tried to find an optimal solution for simple problem instances, i.e. M=1. It showed GA found the optimal solution smoothly. Optimal solutions were found less than 10 generations.

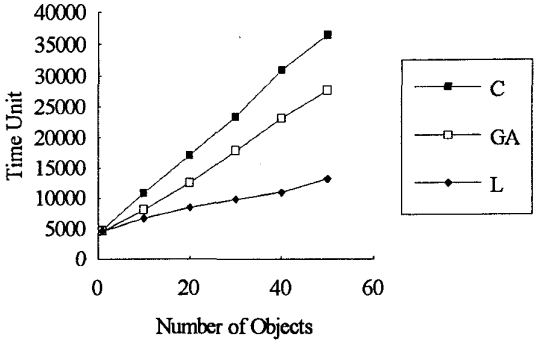

Fig. 2: Costs of different number of objects. C is the cost of the cooperation protocol approach, GA is the cost of GA approach, and L is the estimated low bound.

Next, the experimental results were compared with the cooperation protocol approach for M = 10, 20, 30, 40 and 50. There were 10 problem instances experimented for each M objects and the result was the average of the ten values. Fig. 2 shows GA has better performance than the cooperation protocol. The cooperation protocol is fully distributed and the agents has no idea about the objects or the other agents. It is simple and reactive but not optimal. On the other hand, GA is a centralized model with implicit parallelism. It must have a global view and take long running time in order to explore the search space.

The search space of OSTP is extremely large as shown in Table 1. It is hard to find an optimal solution. Although GA doesn't guarantee an optimal solution, its results can serve as a good reference to the cooperation protocol approach. The reference low bound[10], L, of the cooperation protocol is an ideal estimated value and may not be a realistic solution. On the contrary, all the results of GA are feasible solutions which provides a concrete reference foundation of OST.

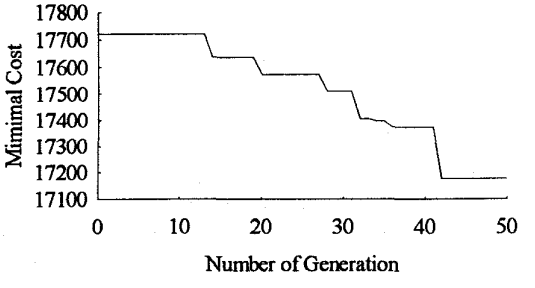

Fig. 3: The minimal cost vs. number of generation of a typical GA running process when M=30 objects.

Fig. 3 shows a typical variation of the minimal cost when the generation grows. The minimal cost gradually decreases with the growing generation. Although a small number of population size and generation were employed, the result is satisfied. The more generation grows, the closer the minimal cost approaches to the optima.

6. CONCLUSIONS

The object sorting task, like most multi-agent tasks, have high complexity. This paper analyzes the object sorting task, shows its NP-completeness and provides a GA approach for the problem. Coding a solution of OSTP for GA is difficult. However, OAS representation is developed to solve the problem. Another problem is to evaluate an OAS. This problem is solved by transferring an OAS to AOS which can be evaluated by Algorithm 1.

For simple problem instances of OSTP, GA can find an optimal solution very quickly. Because the search space is very large when there are many objects, it is hard to explore an optimal solution within a limited running time. The crossover and mutation operators in the GA approach change the object sequence and agent lists in order to explore more areas of the space. Although GA doesn't guarantee optimal solutions for large search space. It provide a concrete reference to other approaches for the object-sorting task.

More experiments using the GA approach for OSTP, which compare performance of different crossover and mutation operators, will be worthwhile for further research.

7. REFERENCES

[1] R. C. Arkin,"Cooperation without Communication: Multi -agent Schema Based Robot Navigation", *Journal of Robotic Systems*, Vol. 9(3), April 1992, pp. 351-364.

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

[4] R. A. Brooks, P. Maes, M. Mataric and G. More, "Lunar Base Construction Robots", *IEEE International Workshop on Intelligent Robots and Systems (IROS '90)*, pp. 389-392, Tsuchiura, Japan, 1990.

[5] R. A. Brooks, "A Robust Layered Control System For A Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, March 1986, pp. 14-23.

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

[7] D. E. Goldberg, *Genetic Algorithms in Optimization, and Machine Learching*, Addision-Wesley Company, Inc. 1989.

[8] S. Hackwood and S. Beni, "Self-organization of Sensors for Swarm Intelligence", *Proc. of 1992 IEEE International Conference on Robotics and Automation*, Nice, pp. 819-829, 1992.

[9] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Compute Science Press, Inc. 1978.

[10] F. C. Lin and J. Y.-j. Hsu, "A Decentralized Cooperation Protocol for Autonomous Robotic Agents", *Proc. of The Second International Symposium on Autonomous Decentralized Systems (ISADS 95)*, Arizona, pp. 420-426, Apr. 1995.

[11] 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 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, pp. 2580-2585, May 1995.

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

[13] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of Permutation Crossover Operators on the Traveling Salesman Problem", *Proc. of the 2rd International Conference on Genetic Algorithms and their Applications"*, L. Erlbaum, pp. 224-230, 1987.

[14] G. J. E. Rawlins(Ed.), "Genetic Operators for Sequencing Problems", Foundations of Genetic Algorithms, 1991.

[15] G. Syswerda, "Uniform Crossover in Genetic Algorithms", *Proc. of the 3rd International Conference on Genetic Algorithms and their Applications"*, Morgan Kaufmann, Publishers, pp. 2-9, 1989.

[16] 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.

[17] D. Whitley, T. Starkweather, and D. Fuquay,"Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator", *Proc. of the 3rd International Conference on Genetic Algorithms and their Applications"*, Morgan Kaufmann, Publishers, pp. 133-140, 1989.