IEMS Vol. 8, No. 3, pp. 181-193, September 2009.

Integrating Ant Colony Clustering Method to a Multi-Robot System Using Mobile Agents

Yasushi Kambayashi†
Department of Computer and Information Engineering Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun Saitama, 345-8501, Japan

Masataka Ugajin
Department of Computer and Information Engineering Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun Saitama, 345-8501, Japan

Osamu Sato
Department of Computer and Information Engineering Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun Saitama, 345-8501, Japan

Yasuhiro Tsujimura
Department of Computer and Information Engineering Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun Saitama, 345-8501, Japan

Hidemi Yamachi
Department of Computer and Information Engineering Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-cho, Minamisaitama-gun Saitama, 345-8501, Japan

Munehiro Takimoto
Department of Information Sciences Tokyo University of Science
2641 Yamazaki, Noda, Chiba, 278-0022, Japan

Hisashi Yamamoto
Faculty of System Design Tokyo Metropolitan University
6-6 Asahigaoka, Hino, Tokyo, 191-0065, Japan

Received Date, January 22, 2008; Received Date, August 7, 2009; Accepted Date, August 10, 2009 (Selected from APIEMS 2007)

Abstract. This paper presents a framework for controlling mobile multiple robots connected by communication networks. This framework provides novel methods to control coordinated systems using mobile agents. The combination of the mobile agent and mobile multiple robots opens a new horizon of efficient use of mobile robot resources. Instead of physical movement of multiple robots, mobile software agents can migrate from one robot to another so that they can minimize energy consumption in aggregation. The imaginary application is making "carts," such as found in large airports, intelligent. Travelers pick up carts at designated points but leave them arbitrary places. It is a considerable task to re-collect them. It is, therefore, desirable that intelligent carts (intelligent robots) draw themselves together automatically. Simple implementation may be making each cart has a designated assembly point, and when they are free, automatically return to those points. It is easy to implement, but some carts have to travel very long way back to their own assembly point, even though it is located close to some other assembly points. It consumes too much unnecessary energy so that the carts have to have expensive batteries. In order to ameliorate the situation, we employ mobile software agents to locate robots scattered in a field, e.g. an airport, and make them autonomously determine their moving behaviors by using a clustering algorithm based on the Ant Colony Optimization (ACO). ACO is the swarm intelligence-based methods, and a multi-agent system that exploit artificial stigmergy for the solution of combinatorial optimization problems. Preliminary experiments have provided a favorable result. In this paper, we focus on the implementation of the controlling mechanism of the multi-robots using the mobile agents.

Keywords: Mobile Agent, Multi-robot, Intelligent Robot Control, Swarm Intelligence, Ant Colony Optimization, Clustering Algorithm

† : Corresponding Author

1. INTRODUCTION

When we pass through terminals of an airport, we often see carts are scattered in the walkway and laborers are manually collecting them one by one. It is a laborious task and not a fascinating job. It is much easier when carts are roughly gathered in any way before laborers' performance. On the other hand, multi-robot systems have made rapid progress in various fields. The core technologies of multi-robot systems are now easily available. It is possible each cart has minimum intelligence that makes a cart be an autonomous robot. Realizing, however, such a robot system as an aggregate of autonomous carts has a big problem of cost, and that is the battery. A big powerful battery is heavy and expensive; therefore robot systems with small batteries are desirable. Thus energy saving is an important issue in such a system.

The application we have in our mind is a kind of intelligent carts. Unintelligent ones are commonly found in large airports. Travelers pick up carts at designated points but leave them arbitrary places. It is a considerable task to re-collect them. It is, therefore, desirable that intelligent carts (intelligent robots) draw themselves together automatically. Simple implementation may be making each cart have a designated assembly point, and when they are free, automatically return to those points. It is easy to implement, but some carts have to travel very long way back to their own assembly point, even though it is located close to some other assembly points. It consumes too much unnecessary energy so that the carts have to have expensive batteries.

In order to ameliorate the situation, we employ mobile software agents to locate robots scattered in a field, e.g. an airport, and make them autonomously determine their moving behaviors by using a clustering algorithm based on the Ant Colony Optimization (ACO). ACO is the swarm intelligence-based methods, and a multi-agent system that exploit artificial stigmergy for the solution of combinatorial optimization problems. Ant colony clustering (ACC) is an ACO specialized for clustering objects. The idea is inspired by the collective behaviors of ants and Deneubourg formulated an algorithm that simulates the ant corps gathering and brood sorting behaviors (Deneubourg, 1991). Algorithms inspired by Ant's behaviors are widely used in solving complex problems recently (Toyoda, 2004) (Becker, 2005).

In the previous paper, we presented a basic idea for controlling mobile multiple robots connected by communication networks (Ugajin, 2007). The framework provides novel methods to control coordinated systems using mobile agents. Instead of physical movement of multiple robots, mobile software agents can migrate from one robot to another so that they can minimize energy consumption in aggregation. In this paper, we describe the details of implementation of the multi-robot system using multiple mobile agents and static agents that implement ACO. The combination of the mobile agents augmented by ACO and mobile multiple robots opens a new horizon of efficient use of mobile robot resources. Preliminary experiments have provided a favorable result (Sato, 2007). During the experiments, we have obtained some insights in ACO (Kambayashi, 2009b). We report the observations we have found in the experiments of the ACO implementation by simulation.

Quasi optimal robots collection is achieved in three phases. The first phase collects the positions of mobile robots. One mobile agent issued from the host computer visits scattered robots one by one and collects the positions of them. Upon returning the position collecting agent, as the second phase, another agent, the simulation agent, performs ACC algorithm and produces the quasi optimal gathering positions for each mobile robot. The simulation agent is a static agent that resides in the host computer. As the third phase, a number of mobile agents are issued from the host computer. One mobile agent migrates to a designated mobile robot, and drives the robot to the assigned quasi optimal position that is calculated in the second phase.

The assembly positions (clustering points) are determined by the simulation agent and influenced but not determined by the initial positions of scattered mobile robot. Thus flexibility and robustness are achieved. Instead of implementing ACC with actual mobile robots, one static simulation agent performs the ACC computation, and then the set of produced positions is distributed by mobile agents. Therefore our method eliminates unnecessary physical movements of mobile agents and contributes energy savings.

The structure of the balance of this paper is as follows. In the second section, we describe the background. The third section describes the ACC algorithm we have employed to calculate the quasi optimal assembly positions. The fourth section describes the agent system that performs the arrangement of the multi-robot. The agent system consists of several static and mobile agents. The static agents interact with the users and compute the ACC algorithm, and other mobile agents gather the initial positions of the robots and distribute the assembly positions. The fifth section demonstrates the feasibility of our system by implementing a team of actual multi-robot system. Each robot determines its position and orientation by using RFID. The sixth section shows numerical experiments and discusses observations we have found through the preliminary experiments. Finally, we conclude in the seventh section and discuss future research directions.

2. BACKGROUND

Kambayashi and Takimoto proposed a framework for controlling intelligent multiple robots using higher-order mobile agents (Kambayashi, 2005) (Takimoto, 2007). The framework helps users to construct intelli-

gent robot control software by migration of mobile agents. Since the migrating agents are higher-order, the control software can be hierarchically assembled while they are running. Dynamically extending control software by the migration of mobile agents enables them to make base control software relatively simple, and to add functionalities one by one as they know the working environment. Thus they do not have to make the intelligent robot smart from the beginning or make the robot learn by itself. They can send intelligence later as new agents. They have implemented a team of cooperative search robots to show the effectiveness of our framework, and demonstrated that their framework contributes to energy saving of multi-robots (Takimoto, 2007). They have achieved significant saving of energy (Nagata 2009). Even though they demonstrate the usefulness of the dynamic extension of the robot control software by using the higher-order mobile agents, such higher- order property is not necessary in our setting. We have employed a simple non-high-order mobile agent system for our framework. Our simple agent system should achieve similar performance.

Deneuburg has formulated the biology inspired behavioral algorithm that simulates the ant corps gathering and brood sorting behaviors (Deneubourg, 1991). His algorithm captured many features of the ant sorting behaviors. His algorithm consists of ant's picking up and putting down objects in a random manner. He has further conjectured that robot team design could be inspired from the ant corpse gathering and brood sorting behaviors (Deneubourg, 1991). Wang and Zhang proposed an ant inspired approach along this line of research that sorts objects with multiple robots (Wang, 2004).

Lumer has improved Deneuburg's model and proposed a new simulation model that is called Ant Colony Clustering (Lumer, 1994). His method could cluster similar object into a few groups. He presented a formula that measures the similarity between two data objects and designed an algorithm for data clustering. Chen *et al*. have further improved Lumer's model and proposed Ants Sleeping Model (Chen, 2004). Since the artificial ants in Deneuburg's model and Lumer's model have considerable amount of random idle moves before they pick up or put down objects, and considerable amount of repetitions occur during the random idle moves. In Chen's ASM model, an ant has two states: active state and sleeping state. When the artificial ant locates a comfortable and secure position, it has a higher probability in sleeping state. Based on ASM, Chen has proposed an Adaptive Artificial Ants Clustering Algorithm that achieves better clustering quality with less computational cost.

Algorithms inspired by behaviors of social insects such as ants that communicate with each other by the stigmergy are becoming popular (Dorigo, 1996). Upon observing real ants' behaviors, Dorigo et al found that ants exchanged information by laying down a trail of a chemical substance (pheromone) that is followed by other ants. They adopted this ant strategy, known as ant colony optimization (ACO), to solve various optimization problems such as the traveling salesman problem (TSP) (Dorigo, 1996). Our ACC algorithm employs pheromone, instead of using Euclidian distance to evaluate its performance

3. THE ANT COLONY CLUSTERING

The coordination of an ant colony is composed by the indirect communication through pheromones. In traditional ACO system, artificial ants leave pheromone signals so that other artificial ant can trace the same path (Deneubourg, 1991). In our ACC system, however, we have made objects have a pheromone so that more objects are clustered in a place where strong pheromone sensed. The randomly walking artificial ants have high probability to pick up an object with weak pheromone, and to put the object where it senses strong pheromone. They are not supposed to walk long distance so that the artificial ants tend to pick up a scattered object and produce many small clusters of objects. When a few clusters are generated, they tend to grow.

Since the purpose of the traditional ACC's is clustering or grouping objects into several different classes based on some properties; it is desirable that the generated chunks of clusters are grow into one big cluster so that each group has distinct characteristic. In our system, however, we want to produce several roughly clustered groups of the same type, while to make each robot has minimum movement. (We assume we have one kind of cart robots, and we do not want robots move long distance.) Therefore we set our artificial ants have the following behavioral rules.

(1) When the artificial ant finds a cluster with certain number of objects, it tends to avoid picking up an object from the cluster. This number can be updated later.
(2) If the artificial ant cannot find any cluster with certain strength of pheromone, it just continues a random walk.

By the restrictions defined in the above rules, the artificial ants tend not to convey objects in long distance, and produce many small heaps of objects. When the initially scattered objects are clustered into small number of heaps, then the number of objects that locks the cluster is updated, and the further activities of the artificial ants re-start to produce smaller number of clusters. Locking a cluster means it inhibits artificial ants to pick up objects from the cluster. We describe the behaviors of the artificial ants below.

When the artificial ants are created, they have randomly supplied initial positions and waking direction. Figure 1 shows the directions that the artificial ant can take. The artificial ants can move only four directions,

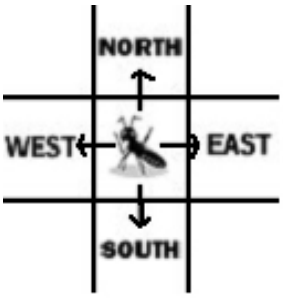


Figure 1. Directions of artificial ant.

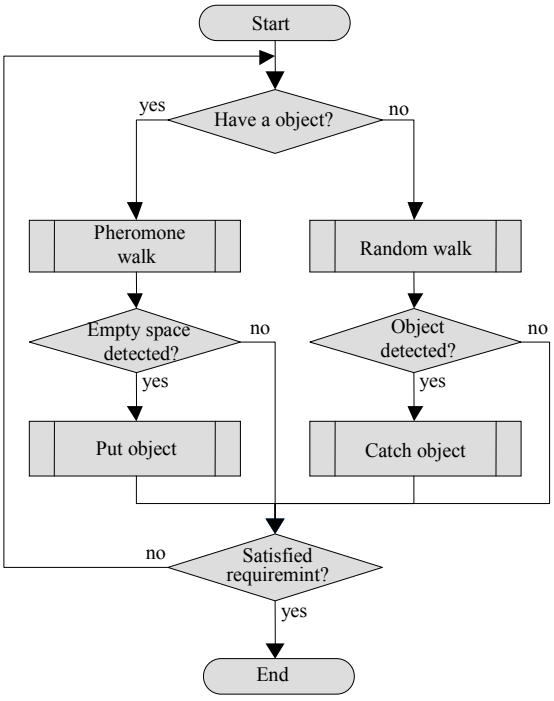


Figure 2. The ACC algorithm.

and can not go along a slant line. While artificial ants perform random walk, when an ant finds an object, the ant probabilistically picks up the object, and continues random walk. While the artificial ant performs random walk, it senses strong pheromone; it probabilistically put the conveying object. The artificial ants repeat this simple procedure until the terminate condition is satisfied. Figure 2 shows the behavior of an artificial ant. We explain several specific actions that each artificial ant performs below.

The base behavior for all the artificial ants is the random walk. The artificial ant that does not have any object is supposed to move straight. The direction along which the artificial ant moves is randomly determined when it is created. Every ten steps, the ant randomly changes the direction as well. The ant also performs side steps from time to time to create further randomness.

While an artificial ant performs random walk, when it finds an isolated object, it picks up the object, and continues random walk. When the artificial ant finds an object, it determines whether it picks up or not based on whether the object is locked or not, and the strength of pheromone the object has. If the object is not locked, picking up or not is probabilistically determined through the formula (1). No artificial ant can pick any locked objects. Whether an object is locked or not is also determined by the formula (1). Here, p is the density of pheromone and k is a constant value. p itself is a function of the number of objects and the distance from the cluster of objects. The formula simply says that the artificial ant does not pick up an object with strong pheromone.

$$f(p) = 1 - \frac{(p+l)*k}{100} \tag{1}$$

Currently, we choose p as the number of adjacent objects. Thus, when an object is completely surrounded by other objects, p is the maximum value nine. We choose k value thirteen in order to prevent any object surrounded by other eight objects be picked up. Then, the probability $f(p)=0$ (never pick it up). l is constant value to make an object be locked. Usually l is zero (not locked). When the number of clusters becomes less than two third of the number of total objects and p is greater than three, l becomes six. When the number of clusters becomes less than one third of the number of total objects and p is greater than seven, l becomes three. When the number of clusters becomes less than the number of the user setting and p is greater than nine, l becomes one. Any objects meet these conditions are locked. This "lock" process prevents artificial ants remove objects from growing clusters, and contributes to stabilizing the clusters' relatively monotonic growth.

When an artificial ant picks up an object, it changes its state into "pheromone walk." In this state, an artificial ant tends probabilistically move toward a place it senses the strongest pheromone. The probability that the artificial ant takes a certain direction is $n/10$, where n is the strength of the sensed pheromone of that direction. Figure 3 shows the strengths of pheromones and their scope. This mechanism makes the artificial ants move toward to the nearest cluster, and subsequently makes the moving distance minimal.

An artificial ant carrying an object determines whether to put the carrying object or to continue to carry. This decision is made based on the formula (2). Thus, the more it senses strong pheromone, the more it tends to put the carrying object. Here, p and k are the same as in the formula (1). The formula simply says when the artificial ant bumped into a locked object; it must put the carrying object next to the locked object. Then, the probability $f(p)=1$ (must put it down).

$$f(p) = \frac{p*k}{100} \tag{2}$$

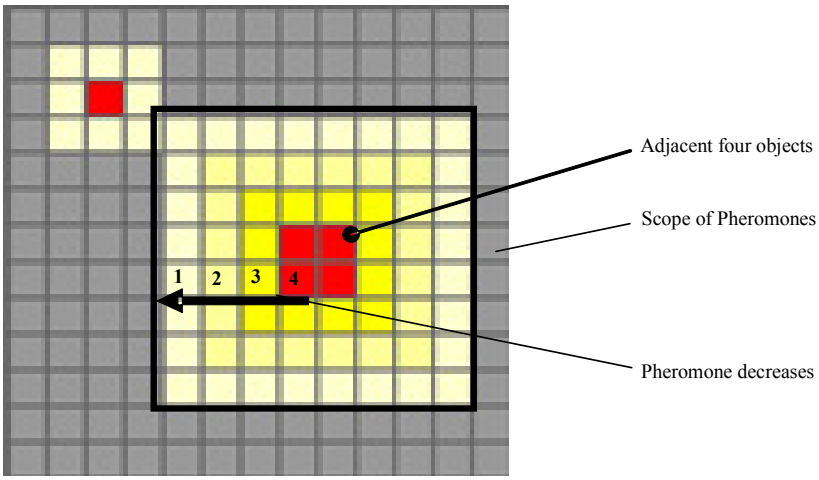


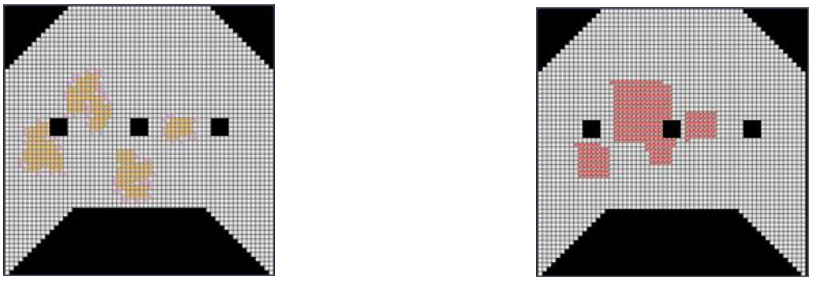
Figure 3. Strengths of pheromones and their scope.

Conventional ACC algorithms terminate when all the objects are clustered in the field, or predefined numbers of steps are executed (Deneubourg, 1991) (Chen, 2004). In such conditions, however, the clustering may be over before obtaining satisfactory clusters. Therefore we set the terminate condition of our ACC algorithm that the number of resulted clusters is less than ten, and all the clusters have more or equal to three objects. This con-dition may cause longer computation time than usual ACC, but preliminary experiments display producing rea-sonably good clustering.

Though the above algorithm guarantees objects are assembled, the resulted arrangement may not be the optimal one. Figure 4 (1) shows an initial state of object arrangement, and the ACC produced a roughly assembled arrangement shown in Figure 4 (2). In our system, objects are mobile robots and it must physically move to the cluster positions. If we assign the resulted positions to the robot and tell them to move there, a robot may have to travel long way. Therefore, as the last phase, we reassign the coordinate to each robot, and make it move to the nearest cluster position. Figure 5 (2) shows the arrangement produced by the reassigned cluster positions.

(1) Initial state (2) A result of clustering

Figure 4. Resulted arrangement of ACC.



(1) A result of clustering (2) Assembly positions are reassigned

Figure 5. Reassigned cluster positions.

4. MOBILE AGENTS

Robot systems have made rapid progress in not only their behaviors but also in the way they are controlled. Also, multi-agent systems introduced modularity, reconfigurability and extensibility to control systems, which had been traditionally monolithic. It has made easier the development of control systems on distributed environments such as intelligent multi-robot systems.

On the other hand, excessive interactions among agents in the multi-agent system may cause problems in the multi-robot environment. In order to lessen the problems of excessive communication, mobile agent methodologies have been developed for distributed environments. In the mobile agent system, each agent can actively migrate from one site to another site. Since a mobile agent can bring the necessary functionalities with it and perform its tasks autonomously, it can reduce the necessity for interaction with other sites. In the minimal case, a mobile agent requires that the connection be established only when it performs migration (Binder, 2001). Figure 6 shows a conceptual diagram of a mobile agent migration. This property is useful for controlling robots that have to work in a remote site with unreliable communication or intermittent communication. The concept of a mobile agent also creates the possibility that new functions and knowledge can be introduced to the entire multi-agent system from a host or controller outside the system via a single accessible member of the intelligent multi-robot system (Kambayashi, 2005).

The model of our system consists of mobile robots and a few kinds of mobile agents. All the controls for the mobile robots as well as ACC computation performed in the host computer are achieved through the static and mobile agents. They are: 1) user interface agent (UIA), 2) operation agents (OA), 3) position collecting agents (PCA), 4) clustering simulation agents (CSA), and 5) destination agents (DA). All the agents except UIA and CSA are mobile agents. The mobile agent traverses mobile robots scattered in the field one by one and collect their coordinates. Figure 7 shows the interactions of the cooperative agents to control a mobile robot.

The followings are details of each agent:

1) *User Interface Agent (UIA)*: User interface agent (UIA) is a static agent that resides on the host

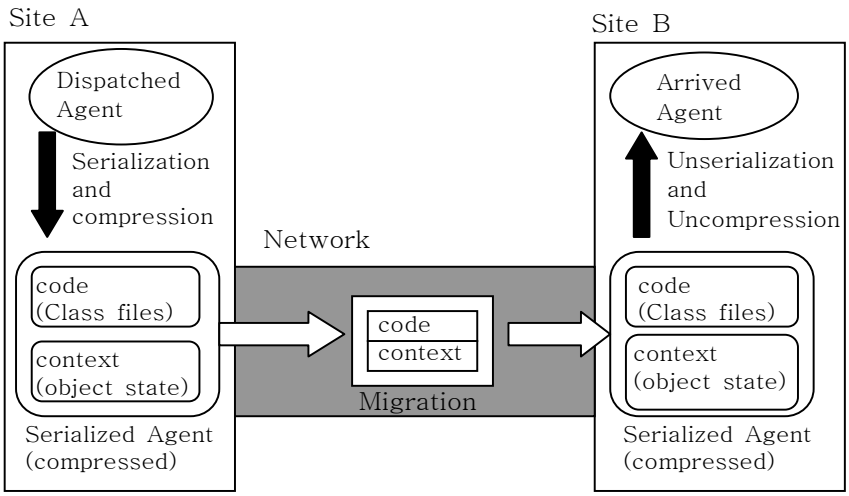


Figure 6. A mobile agent is migrating from site A to site B.

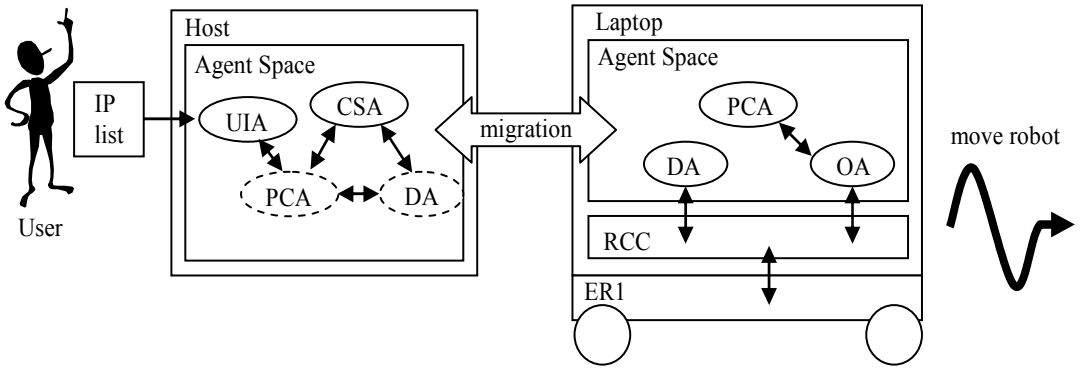


Figure 7. Cooperative agents to control a mobile robot.

computer and interacts with the user. It is expected to coordinate entire agent system. When the user creates this agent with a list of IP addresses of the mobile robots, UIA creates PCA and pass the list to it.

2) *Operation Agent (OA)*: Each robot has at least one operation agent (OA). It has the task that the migrating robot is supposed to perform. Each mobile robot has its own OA.
3) *Position Collecting Agent (PCA)*: A distinct agent called position collecting agent (PCA) traverses mobile robots scattered in the field one by one and collects their coordinates. PCA is created and dispatched by UIA. Upon returning to the host computer, it hands in the collected coordinates to the clustering simulation agent (CSA) for ACC.
4) *Clustering Simulation Agent (CSA)*: The host computer has the static clustering simulation agent (CSA). This agent actually perform the ACC algorithm by using the coordinates collected by PCA as the initial positions, and produced the quasi optimal assembly positions of the mobile robots. Upon terminating the computation, CSA creates a number of destination agents (DA).
5) *Destination Agent (DA)*: The quasi optimal arrangement coordinates produced by the CSA are delivered by destination agents (DA). One destination agent is created for one mobile robot, and it has the coordinate the mobile robot is supposed to drive. The DA drives its mobile robot to the designated assembly position.

OA detects the current coordinate of the robot on which it resides. Our robot system employs RFID (Radio Frequency Identification) to get precise coordinates. We set RFID tags in a regular grid shape under the floor carpet tiles. The tags we chose have a small range so that the position-collecting agent can obtain fairly precise coordinates from the tag (Kambayashi, 2009a).

Each robot has its own IP address and UIA hands in the list of the IP addresses to PCA. First, PCA migrates to an arbitrary robot and starts hopping between them one by one. It communicates locally with OA, and writes the coordinates of the robot into its own local data area. When PCA gets all the coordinates of the robots, it returns to host computer. Upon returning to the host computer, PCA creates CSA and hands in the coordinate data to CSA which computes the ACC algorithm.

CSA is the other static agent and its sole role is ACC computation. When CSA receives the coordinate data of all the robots, it translates the coordinates into coordinates for simulation, and then performs the clustering. When CSA finishes the computation and produces a set of assembly positions, it then creates the set of procedures for autonomous robot movements.

Then CSA creates DA that conveys the set of procedures to the mobile robots as many as the number of robots. Each DA receives its destination IP address from PCA, and the set of procedures for the destination robot, and then migrates to the destination robot. Each DA has a set of driving procedures that drives its assigned robot to the destination, while it avoids collision. OA has the basic collision detection and avoidance procedures, and DA has task-specific collision avoidance procedures.

We have implemented the prototype of the multi-agent system for mobile robot control by using Agent Space (Satoh, 1999). Agent Space is a framework for constructing mobile agents that is developed by Satoh. By using its library, the user can implement mobile agent environment by Java language.

In Agent Space, mobile agents are defined as collections of call-back methods, and we have to implements interfaces defined in the system. In order to create a mobile agent, the application calls create method. An agent migrates to another site by using move and leave methods. When an agent arrives, arrive method is invoked. Migration is achieved through duplicating itself at the destination site. Thus move method and leave method are used as a pair of methods for actual migration. Figure 8 shows the move method for example. The other methods are implemented similarly. The users are expected to implement a destructor to erase the original agent in the leave method. Agent Space also provides service API's such as move method to make migrate agents and invoke method to communicate to another agent. Figures 9 and 10 show how PCA and DA work, respectively.

The following is the PCA implementation:

(1) UIA invokes create method to create the mobile agent PCA, and hands in the list of the IP addresses of mobile robots to PCA.
(2) PCA invokes move method so that it can migrate to the mobile robot specified in the top of the list of IP addresses.
(3) Invoke leave method.
(4) The agent actually migrates to the specified mobile robot.
(5) Invoke arrive method in the destination robot, and the PCA communicate to OA in order to receive the coordinate of the robot.
(6) Check the next entry of the IP address list; if PCA

```
If (comeBackHost == true){
    try {
        URL url = new URL("matp://hostIP:hostPort");
            context.move(url);
    }
    catch(InvalidURLException ex){
    ex.printStackTrace();
    }
    catch (IOException ex){
    …
    }
```

Figure 8. The move method.

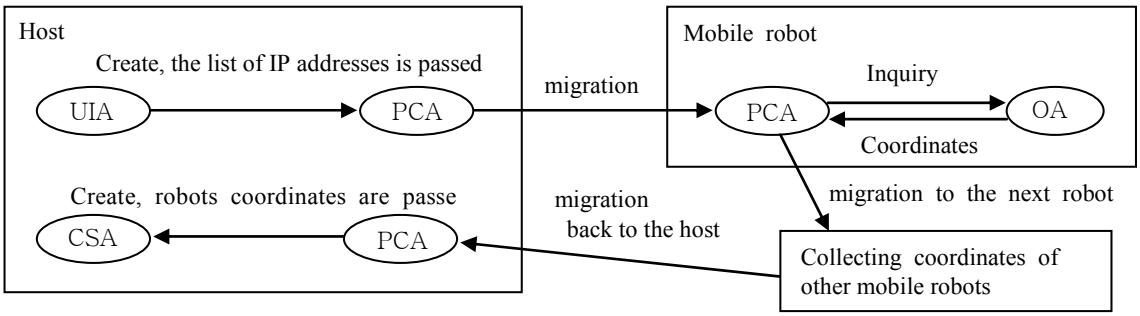


Figure 9. The position collecting agent (PCA).

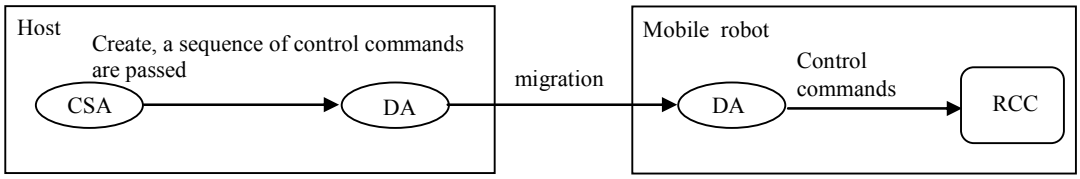


Figure 10. The destination agent (DA).

visits all the mobile robots, it returns to the host computer, otherwise migrates to the next mobile robot with the IP address of the next entry in the list.

The following is the DA implementation:

(1) CSA creates the mobile agents DA as many as the number of the mobile robots in the field.
(2) Each DA receives the IP address to where the DA is supposed to migrate, and the sequence of the command to drive the robot.
(3) The agents actually migrate to the specified mobile robots.
(4) Each DA invokes arrive method in the destination robot, and it communicate with the robot control software called RCC (Robot Control Center) in the notebook computer on the robot in order to drive the mobile robot.

5. THE ROBOTS

In this section, we demonstrate that the model of mobile agents with the ant colony clustering (ACC) is suitable for controlling the intelligent multiple robots. We have employed the ER1 Personal Robot Platform Kit by Evolution Robotics Inc. (Evolution Robotics, 2008) as the platform for our prototype. Each robot has two servomotors with tires. The power is supplied by a rechargeable battery. It has a servomotor controller board that accepts RS-232C serial data from a host computer. Each robot holds one notebook computer as its host computer. Our control mobile agents migrate to these host computers by wireless LAN. One RCC resides on each host computer. Figure 11 shows the team of mobile multi-robots work under control of mobile agents.

In the previous implementation, an agent on the robot calculates the current coordinate from the initial position, and that computed coordinates are different from actual positions (Ugajin, 2007). The current implantation employs RFID (Radio Frequency Identification) to get precise coordinates (Kambayashi, 2009a). We set RFID tags in a regular grid shape under the floor carpet tiles as shown in Figure 12. The tags we chose have small range so that the position collecting agent can obtain fairly precise coordinates from the tag.

For driving robots in a quasi-optimal route, one needs not only the precise coordinates of each robot but also the direction each robot faces. In order to determine the direction that it is facing, each robot moves straight ahead in the direction it is currently facing and obtains two positions (coordinates) from RFID tags under the carpet tiles. Determining current orientation is important because there is a high cost for making a robot rotates through a large angle. It is desirable for each robot be assigned rather simple forward movements rather than complex movement with several direction-changes when there are obstacles to avoid. Therefore whenever OA is awake, it performs the two actions, i.e. obtaining the current position and calculating the current direction.

Since each carpet tile has nine RFID tags, as shown in Figure 13, the robot is supposed to obtain the current position as soon as OA gets the sensor data from the RFID module. If OA can not obtain the position data, it means the RFID module can not sense a RFID tag, OA makes the robot rotate until the RFID module senses a RFID tag. Once OA obtains the current position, it drives the robot a short distance until the RFID module detects the second RFID tag. Upon obtaining two positions, it is a simple computation to determine the direction that

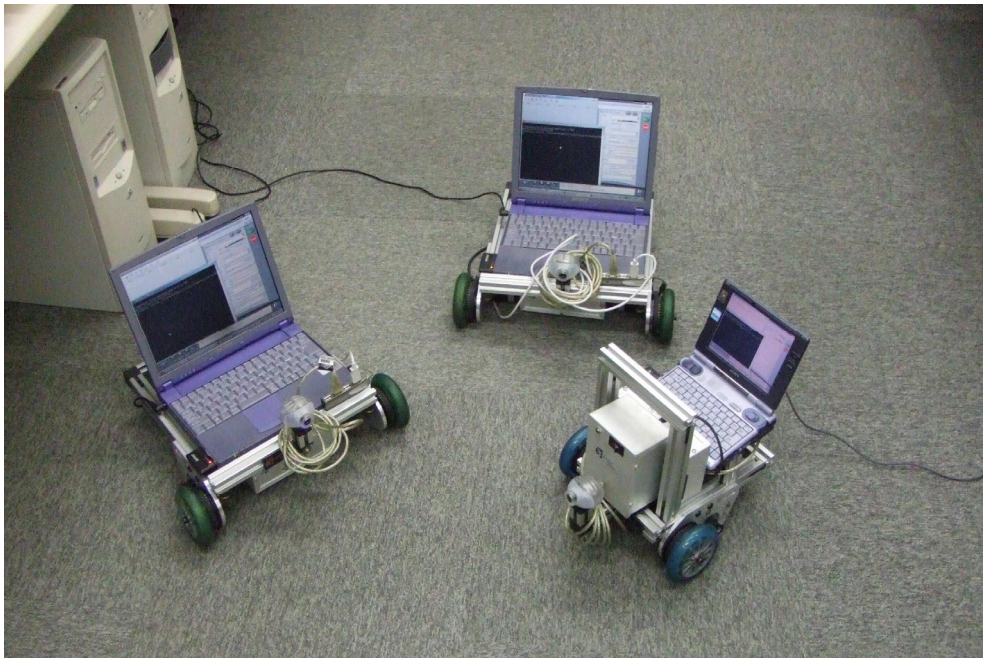


Figure11. A team of mobile robots work under control of mobile agents.

Figure 12. RFID under a carpet tile.

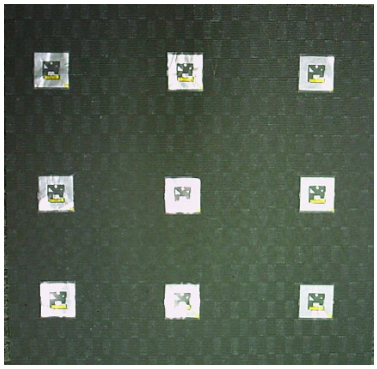


Figure 13. RFID tags in square formation.

which the robot is moving, as shown in Figure 14.

In a real situation, we may find the robot close to the wall and facing it. Then we can not use the simple method just described above. The robot may move forward and collide with the wall. In order to accommodate such situations, we make RFID tags near the wall have special signal to the robot that tells it that it is at the end of the field (near the wall) so that the robot that senses the signal can rotate to the opposite direction, as shown in Figure 15. This is only required in our experimental implementation because the current collision detection mechanism does not otherwise work well enough to avoid collision with the wall. Therefore we employ the special signal in the RFID tags. When the robot finds wall while moving to obtain the direction, it arbitrarily changes the direction, and starts obtaining two positions again.

6. NUMERICAL EXPERIMENTS

We have conducted several numerical experiments in order to demonstrate the effectiveness of our multi-robot system using the ACC algorithm by using a simulator. The simulator has an airport-like field with sixty times sixty grid cut two equilateral triangles with sides fifteen from top left and right, and one trapezoid with top side thirty, bottom side fifty-eight and height fifteen from the bottom as shown Figure 16 (a) through (c). We have compared the results of our ACC algorithm with an algorithm that performs grouping objects at predefined assembly positions. The desirable arrangements are those of low moving costs and fewer numbers of clusters. Therefore, we defined comparing fitness value that is represented in the formula (3). Here, *Clust* represents the number of clusters, and *Ave* represents the average distance of all the objects moved.

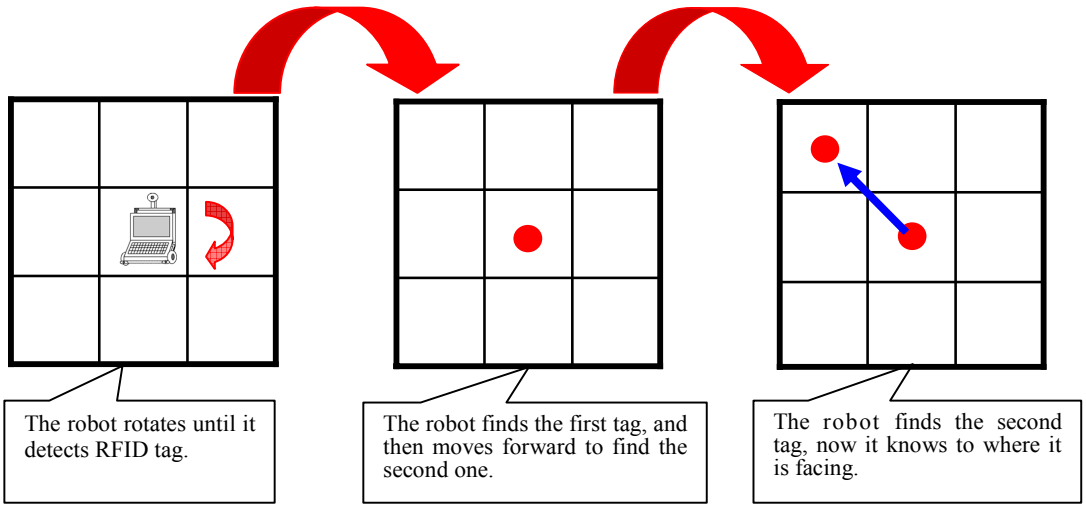


Figure 14. Determining orientation from two RFID tag positions.

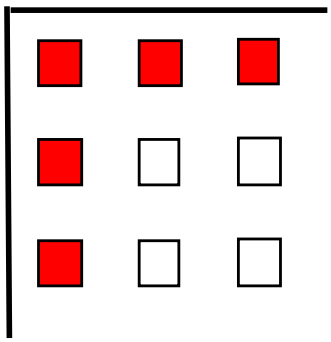


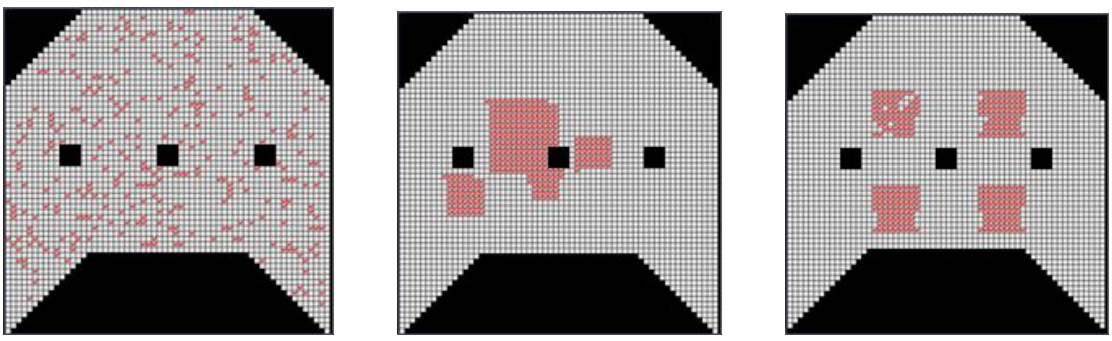
Figure 15. RFID tags near the wall emit special signal.

$$Eval = Ave + Clust \tag{3}$$

The results of the experiments are shown in Tables 1 through 5. The results of our ACC algorithm are displayed in the column "Ant Colony Clustering," and the results of gathering objects predefined assembly positions are displayed in the column "Specified Position Clustering." We have implemented both algorithms in the simulator. In the "Specified Position Clustering," we set four assembly positions. For the experiments, we set three cases that are the number of objects 200, 300 and 400. In every case, we set the number of artificial ants 100. We have performed five trials for each setting, and obtained the evaluation values calculated by the formula (3), as shown in Tables 1 through 3. In all three cases, the computations are over less than six seconds under the computing environment of the experiments is JVM on Pentium IV 2.8GHz and Windows XP. Since the physical movements of robots take much more time than computation, we can ignore the computation complexity of the ACC algorithm.

In every case, our ACC algorithm produced a better result than that for predefined assembly positions. We can observe that as the number of objects increases, the average moving distance of each object decreases. We can, however, observe that increasing the number of ants does not contribute the quality of clustering as shown in Table 4 and 5.

In the first implementation, we did not employ a "lock" process. Under those conditions we observed no convergence. The number of clusters was roughly 300.



(a) Initial state (b) Assembled by ACC (c) Predefined positions

Figure 16. Clustered objects constructed by assembling at positions computed by ACC and at predefined positions.

When we add the "lock" process, clusters start to emerge, but they tend to stabilize at about thirty clusters. Then the artificial ants see only immediately adjacent places. When we add the feature to artificial ants that they can dynamically change their scope in the field, the small clusters converge into a few big clusters. Then the artificial ants can see ten to twenty positions ahead. As a result of our experiments, we realize that we need to con-

Table 1. Airport Field (Objects: 200, Ant Agent: 100).

Airport Field	Ant Colony Clustering				Specified Position Clustering			
	Cost	Ave	Clust	Eval	Cost	Ave	Clust	Eval
1	2279	11.39	4	15.39	5826	29.13	4	33.13
2	2222	11.11	3	14.11	6019	30.09	4	34.09
3	2602	13.01	2	15.01	6194	30.97	4	34.97
4	2433	12.16	3	15.16	6335	31.67	4	35.67
5	2589	12.94	4	16.94	6077	30.38	4	34.38

Table 2. Airport Field (Objects: 300, Ant Agent: 100).

Airport Field	Ant Colony Clustering				Specified Position Clustering			
	Cost	Ave	Clust	Eval	Cost	Ave	Clust	Eval
1	2907	9.69	4	13.69	8856	29.52	4	33.52
2	3513	11.71	4	15.71	9142	30.47	4	34.47
3	3291	10.97	4	14.97	8839	29.46	4	33.46
4	3494	11.64	3	14.64	8867	29.55	4	33.55
5	2299	7.66	6	13.66	9034	30.11	4	34.11

Table 3. Airport Field (Objects: 400, Ant Agent: 100).

Airport Field	Ant Colony Clustering				Specified Position Clustering			
	Cost	Ave	Clust	Eval	Cost	Ave	Clust	Eval
1	4822	12.05	1	13.05	11999	29.99	4	33.99
2	3173	7.93	6	13.93	12069	30.17	4	34.17
3	3648	9.12	4	13.12	12299	30.74	4	34.74
4	3803	9.51	3	12.51	12288	30.72	4	34.72
5	4330	10.82	5	15.82	12125	30.31	4	34.31

Table 4. Airport Field (Objects: 300, Ant Agent: 50).

Airport Field	Ant Colony Clustering				Specified Position Clustering			
	Cost	Ave	Clust	Eval	Cost	Ave	Clust	Eval
1	3270	10.9	5	15.9	9156	30.52	4	34.52
2	2754	9.18	5	14.18	9058	30.19	4	34.19
3	3110	10.36	4	14.36	9006	30.02	4	34.02
4	3338	11.12	3	14.12	9131	30.43	4	34.43
5	2772	9.24	5	14.24	8880	29.6	4	33.6

Table 5. Airport Field (Objects: 300, Ant Agent: 200).

Airport Field	Ant Colony Clustering				Specified Position Clustering			
	Cost	Ave	Clust	Eval	Cost	Ave	Clust	Eval
1	3148	10.49	4	14.49	8887	29.62	4	33.62
2	3728	12.42	3	15.42	8940	29.8	4	33.8
3	2936	9.78	5	14.78	8923	29.73	4	33.73
4	3193	10.64	3	13.64	9408	31.36	4	35.36

tinue to improve our ACC algorithm.

7. CONCLUSION AND FUTURE WORKS

We have presented a framework for controlling mobile multiple robots connected by communication networks. Mobile and static agents collect the coordinates of scattered mobile multiple robots and implement the ant colony clustering (ACC) algorithm in order to find quasi-optimal positions to assemble the mobile multiple robots. Making mobile multiple robots perform the ant colony optimization is enormously inefficient. Therefore a static agent performs the ACC algorithm in its simulator and computes the quasi-optimal positions for the mobile robots. Then other mobile agents carrying the requisite set of procedures migrate to the mobile robots, and so direct the robots using the sequence of the robot control commands constructed from the given set of procedures.

Since our control system is composed of several small static and mobile agents, it shows an excellent scalability. When the number of mobile robots increases, we can simply add the increases number of mobile software agents to direct the mobile robots. The user can enhance the control software by introducing new features as mobile agents so that the multi-robot system can be extended dynamically while the robots are working. Also mobile agents decrease the amount of the necessary communication. They make mobile multi-robot applications possible in remote site with unreliable communication. In unreliable communication environments, the multi-robot system may not be able to maintain consistency among the states of the robots in a centrally controlled manner. Since a mobile agent can bring the necessary functionalities with it and perform its tasks autonomously, it can reduce the necessity for interaction with other sites. In the minimal case, a mobile agent requires that the connection be established only when it performs migration (Binder, 2001). This property is useful for controlling robots that have to work in a remote site with unreliable communication or intermittent communication. The concept of a mobile agent also creates the possibility that new functions and knowledge can be introduced to the entire multi-agent system from a host or controller outside the system via a single accessible member of the intelligent multi-robot system (Kambayashi, 2005) (Takimoto, 2007). While our imaginary application is simple cart collection, the system should have a wide variety of applications.

We have implemented a team of mobile robots to show the feasibility of our model. In the current implementation, an agent on the robot can obtain fairly precise coordinates of the robots from RFID tags (Kambayashi, 2009a).

The ACC algorithm we have proposed is expected to makes the total distances of moving objects minimal. We have analyzed and demonstrated the effectiveness of our ACC algorithm through implementing a simulator, and then performing several numerical experiments with various setting. We have so far observed favorable result from the experiments.

Comparing with the time for robot movements, the computation time for the ACC algorithm is negligible. Even if the number of artificial ants increases, the computation time is expected to increase linearly, and the number of objects should not influence the computation complexity (Sato, 2007). Because one step of each ant's behaviors is simple and we can assume it takes constant execution time. Even though it is obvious, we need to confirm the fact by the numerical experiments. One big problem is that how we should take account of the collision avoidance behaviors of robots into the simulation. We have to investigate the real robot movements much further.

During the experiments, we experienced the following unfavorable situations (Sato, 2007):

(1) Certain initial arrangements of objects causes very long periods for clustering,
(2) If one large cluster is created at the early stage of the clustering and the rest of the field has scarce objects, then all the objects are assembled into one large cluster. This situation subsequently makes aggre-gate moving distance long, and
(3) As a very rare case, the simulation does not converge.

Even though such cases are rare, these phenomena suggest that our ACC algorithm is still under development, and has certain room for improvement. As we mentioned in the previous section, we need to design the artificial ants have certain complex features that changes their scope to adapt their circumstances. Such dynamic scope range is one direction for our future work.

On the other hand, when certain number of clusters are emerged and stabilize, we can coerce them into several (three or four) clusters by calculating the optimal assembly points. This calculating the coerce assembly points should be one of the other directions for our future work. Another direction is that we may compute the number of clusters and their rough positions prior to performing the ACC algorithm, so that we can save much computation time. In many ways, we have much room for improving our assembly point calculation method before integrating everything into one working multi-robot system.

ACKNOWLEDGEDGEMENT

This work is supported in part by Japan Society for Promotion of Science (JSPS), with the basic research program (C) (No. 20510141), Grant-in-Aid for Scientific Research.

REFERENCES

Becker M. and Szczerbicka, H. (2005), Parameters Influencing the Performance of Ant Algorithm Applied to Optimisation of Buffer Size in Manufacturing, *Industrial Engineering and Management Systems*, **4**(2), 184-191.

Binder, W. J. and Villazon, G. H. (2001), Portable Resource Control in the J-Seal 2 Mobile Agent System, *Proceedings of International Conference on Autonomous Agents*, 222-223.

Chen, L., Xu, X., and Chen, Y. (2004), An adaptive ant colony clustering algorithm, *Proceedings of the Third IEEE International Conference on Machine Learning and Cybernetics*, 1387-1392.

Deneuburg, J., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C. and Chretien, L. (1991). The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot, *Proceedings of First Conference on Simulation of Adaptive Behavior*: *From Animals to Animats*, MIT Press, 356-363.

Dorigo, M. and Gambardella, L. M. (1996), Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman, *IEEE Transaction on Evolutionary Computation*, **1**(1), 53-66.

Evolution Robotics Ltd. Homepage (2008), http://www.evolution.com/.

Kambayashi, Y., Sato O., Harada, Y., and Takimoto, M., (2009a), Design of an Intelligent Cart System for Common Airports, *Proceedings of 13th International Symposium on Consumer Electronics*, CD-ROM.

Kambayashi, Y. and Takimoto, M. (2005), Higher-Order Mobile Agents for Controlling Intelligent Robots, *International Journal of Intelligent Information Technologies*, **1**(2), 28-42.

Kambayashi, Y., Tsujimura, Y., Yamachi, H., Takimoto M., and Yamamoto, H. (2009b), Design of a Multi-Robot System Using Mobile Agents with Ant Colony Clustering, *Proceedings of Hawaii International Conference on System Sciences*, IEEE Computer Society, CD-ROM.

Lumer, E. D. and Faieta, B. (1994), Diversity and Adaptation in Populations of Clustering Ants, *From Animals to Animats 3: Proceedings of the 3rd International Conference on the Simulation of Adaptive Behavior*, MIT Press, 501-508.

Nagata, T., Takimoto, M., and Kambayashi, Y. (2009), Suppressing the Total Costs of Executing Tasks Using Mobile Agents, *Proceedings of the 42nd Hawaii International Conference on System Sciences*, IEEE Computer Society, CD-ROM.

Satoh, I. (1999), A Mobile Agent-Based Framework for Active Networks, *Proceedings of IEEE System*, *Man and Cybernetics Conference*, 71-76.

Sato, O., Ugajin, M., Tsujimura, Y., Yamamoto, H., and Kambayashi, Y. (2007), Analysis of the Behaviors of Multi-Robots that Implement Ant Colony Clustering Using Mobile Agents, *Proceedings of the Eigth Asia Pacific Industrial Engineering and Management System*, CD-ROM.

Takimoto, M., Mizuno, M., Kurio, M., and Kambayashi, Y. (2007), Saving Energy Consumption of Multi-Robots Using Higher-Order Mobile Agents, *Proceedings of the First KES International Symposium on Agent and Multi-Agent Systems*: *Technologies and Applications, Lecture Notes in Artificial Intelligence 4496*, Springer-Verlag, 549-558.

Toyoda, Y. and Yano, F. '2004). Optimizing Movement of a Multi-Joint Robot Arm with Existence of Obstracles Using Multi-Purpose Genetic Algorithm, *Industrial Engineering and Management Systems*, **3**(1), 78-84.

Ugajin, M., Sato, O., Tsujimura,Y., Yamamoto, H., Takimoto, M., and Kambayashi, Y. (2007), Integrating Ant Colony Clustering Method to Multi-Robots Using Mobile Agents, *Proceedings of the Eigth Asia Pacific Industrial Engineering and Management System,* CD-ROM.

Wang T. and Zhang, H. (2004). Collective Sorting with Multi-Robot, *Proceedings of the First IEEE International Conference on Robotics and Biomimetics*, 716-720.