An Object Transportation System with Multiple Robots and Machine Learning

Ying Wang and Clarence W. de Silva, *Fellow, IEEE*

Abstract — **This paper investigates the problem of object transportation, particularly pushing or moving an object to a goal location and orientation, using multiple robots. A multi-agent architecture is established to realize effective cooperation between multiple autonomous intelligent robots, in carrying out the task. Machine Learning is incorporated into the architecture. In the developed approach, the world state of the task is established by fusing sensory information. Two machine learning and optimization methods, Reinforcement Learning (RL) and Genetic Algorithms (GA), are combined to learn a cooperation strategy and based on which, determine the optimal actions to reach the task goal. The outputs of RL and GA are evaluated by an arbitrator using a probabilistic method, which will resolve conflicts and improve the overall performance. The feasibility of the scheme is illustrated through computer simulation.**

I. INTRODUCTION

IN recent years multi-agent robotics has become an active research area in the fields of robotics and computer science [1]-[2]. In the present context, an agent is a computer-based entity situated in some environment with other agents who cooperate in achieving a common goal, and is capable of taking autonomous actions in this environment in order to reach the goal. There are many advantages of multi-agent robotic systems. These include improved system performance, greater efficiency, better fault tolerance, robustness, improved cost effectiveness, distributed sensing and action, and inherent parallelism.

A multi-robot system is usually related to multi-agent systems. However, there exist some differences between these two concepts. Multi-agent technology mainly focuses on the coordination mechanism between agents in achieving the goal of the system. The agent can be a physical entity or a software entity, and the goal can be physical or nonphysical as well. Meanwhile, multi-robot systems usually use physical robots, which are hardware agents, for accomplishing physical tasks, in practice. Multi-agent technology may provide a theoretical foundation and a simulation environment for a multi-robot system; however, a multi-robot system needs to consider physical details and hardware such as robot structural details and configuration, dynamics, actuators, sensors, driving systems, controllers, and so on. A multi-agent architecture may use both hardware agents and software agents.

A useful application of multi-robot systems is the multi-robot object-transportation problem. In this task, several autonomous robots move in a coordinated manner to move an object from a given location to a goal location and orientation, with associated performance requirements such as speed, obstacle avoidance, and risk/damage reduction. This task is important for several reasons. It represents a task that needs the basic capabilities of a multi-robot system, and hence can be used as a benchmark problem to develop a test bed for evaluating the related technologies. Furthermore, an object-transportation system has direct practical applications, as in shipping, storage, construction, parts transfer, safety and security operations, and pathway clearing. In a transportation process, the robots may need to detect and move some obstacles in the path. The robots in such a system need to communicate and cooperate with each other to determine their own optimal transportation strategies, applied forces, directions and magnitudes of the carried out motion steps (displacement and speed), and so on, to complete a common goal. Vision, sound, force, torque and motion sensors may be needed to detect the orientation and position of the object and the robots, changes in the environment, and local dynamic models.

In this paper we develop a multi-robot system integrated with machine learning for carrying out object transportation. Two approaches of machine learning, Reinforcement Learning (RL) and Genetic Algorithms (GA), are used to learn the optimal cooperation strategies for the multi-robot object-transportation process. A computer simulation is implemented in Java language and the results are evaluated.

II. RELATED WORK

Some pioneering work has been done in multi-robot object-transportation applications and in the machine learning field. Mataric, *et al.* [3] studied the cooperative

Manuscript received September 02, 2004.This work was supported in part by National Science and Engineering Research Council (NSERC) of Canada and K.C.WONG Education Foundation of Hong Kong.

The authors are with the Department of Mechanical Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: yingwang@interchange.ubc.ca; desilva@mech.ubc.ca; Phone: 1-604-822-6291; Fax: 1-604-822-2403) .

multi-robot object-pushing problem. In their case, two autonomous six-legged robots were used to push an object to a goal location. Simple communication protocols were used for cooperation between robots. Since the approach was based on a predefined reactive algorithm, there was no mechanism of learning or evolution.

Rus, *et al.* [4] explored approaches to move furniture with teams of autonomous robots. They showed how coordinated pushing by robots could change the pose (position and orientation) of objects. They presented a way to select the active robots (robots which execute pushing motions) and stationary robots (robots which constrain the motion of the object by remaining fixed in place along a specified track) and a method of role switching of the two types of robots, over the course of a task. Their method as well lacked an intelligent learning mechanism.

Miyata, *et al.* [5] studied cooperative transport by multiple robots in unknown static environments associated with real-time assignment. They suggested an architecture for the task-assignment and motion-planning system. However, intelligent learning and evolution capabilities were not integrated into the architecture.

Stone and Veloso [6] suggested a layered learning approach for multi-agent cooperation. They proposed to introduce machine learning into a multi-agent system. In their paper, neural networks were used to learn the low-level skills of a robot and the decision tree method was employed to learn the high-level strategies. However, their work mainly focused on robot soccer rather than a general multi-robot object-transportation problem.

Asada, *et al.* [7] studied cooperative behavior of mobile robots by using the reinforcement learning (RL) method. Specifically, they introduced RL into multi-robot cooperation. The method was applied to a soccer playing situation, as before.

Liu and Wu [2] investigated multi-agent cooperative behavior evolution in an object-pushing problem. A genetic algorithm was employed as a machine learning tool. The basis of the present paper is somewhat similar to their work. However, the present paper integrates reinforcement learning and genetic algorithms into an object pushing task, where the problems that arise are resolved by arbitrating these two methods.

There are some common shortcomings in the existing work in the area of multi-robot object transportation, as listed below:

1. Inadequate emphasis on the learning mechanism in the object- transportation process.
2. No integration of learning and evolution. (Learning is not optimal and also may fail in a complicated and unknown environment.)
3. Information (model) on local dynamic/kinematic interactions between robots and the object is not explicitly used.
4. Control of robot contact forces is not included in the overall control scheme.
5. Implementations on physical robots are few and not rigorous.

The work presented in this paper undertakes to address and correct some of these shortcomings.

III. Multi-robot Object-Pushing System

A. Problem Definition

The system developed in the present work incorporates two or more autonomous robots to perform coordinated movements for transporting an object (for example, a box) toward a goal location and orientation. Obstacles may be present and have to be accommodated within the task. An experimental system, which is under development in our laboratory for this purpose, is shown in Figure 1.

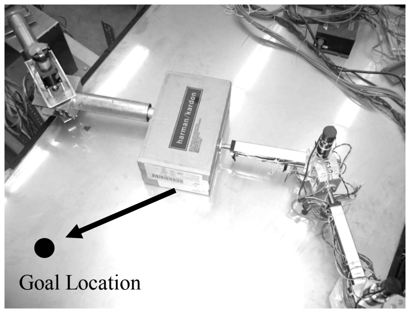

Fig.1. An experimental system for cooperative object pushing.

B. Multi-agent architecture

In this paper, a new multi-agent architecture is established, which is suitable for multi-robot object-transportation tasks. This architecture is shown in Fig.2.

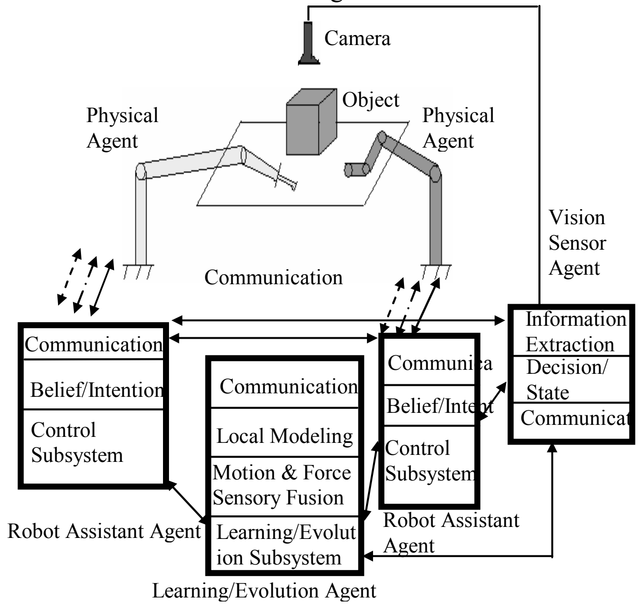

Fig.2. A multi-agent architecture for the cooperative object-transportation system.

In Figure 2, there are two physical agents and four software agents, constituting a multi-agent cooperative object-transportation system. The two physical robots act as the physical agents. They communicate with the four software agents to perform coordinated motions for transporting the object (box) to the goal location. All agents have their own communication subsystems and decision systems. They relate the sensory data to their actions based on the current internal state and the knowledge base. In addition, the two robot-assistant agents contain a force/position control subsystem to drive the physical robots to the desired positions.

In the present multi-agent architecture, there is a vision sensor agent linked to a video camera. This agent receives and processes current video information of the system, and broadcasts the results to all agents. The locations of the robot end effectors and the object are monitored by the vision agent, which also manages the video information for further use.

Corresponding to each physical agent there is a robot assistant software agent. The assistant agent senses data of the corresponding physical robot, makes decisions and generates control/actuating signal for the physical agent. It also communicates with other software agents to obtain current world model and determine a suitable optimal action.

The multi-agent system has a learning/evolution agent. It plays the role of a monitor, learner, and an advisor. This agent receives the current world state; i.e., the data of the goal and obstacles, from other agents; carries out local modeling and leaning using its intelligent learning/evolution knowledge base; reasons an optimal cooperation strategy; and sends the suggested actions to the two robot-assistant agents.

IV. Learning and Evolution

It is desirable and useful for a multi-robot system to learn from prior experience on cooperation strategies, and improve its future actions on this basis. A system with learning capabilities can cope with difficulties in a dynamic and unknown environment, and suitably adapt to complete the expected task. Two different approaches of machine learning and optimization, Reinforcement Learning and Genetic Algorithms, are incorporated into the multi-robot object-transportation system. These two approaches are outlined next.

A. *Reinforcement Learning (RL)*

A key problem in a multi-robot object-transportation system is how to select the cooperation strategy; i.e., force location and direction of each robot, according to the current world state, for generating an effective strategy of object transportation. This problem can be partly solved by the Reinforcement Learning method.

Specifically, the Q-learning algorithm, a type of Reinforcement Learning, is used to make the robots learn the cooperation strategies. The Q-learning algorithm is outlined below [8]:

```
• For each state $s$ and action $a$, initialize the table entry $\hat{Q}(s,a)$ to zero
• Observe the current state $s$
• Do repeatedly the following:
    ▪ Select an action $a$ and execute it
    ▪ Receive immediate reward $r$
    ▪ Observe the new state $s'$
    ▪ Update the table entry for $\hat{Q}(s,a)$ as follows:
          $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$        (1)
    ▪ $s \leftarrow s'$
```

In this paper, the object is simplified as a rectangular box and the state *s* is represented by the angle α in Fig. 3, which describes the relative location of the goal and the box center. The cooperative action *a* of the three robots is modeled as the vector $(\theta_1, \beta_1, \theta_2, \beta_2, \theta_3, \beta_3)$. Here, θ_i represents the force direction and β_i represents the force location of a robot.

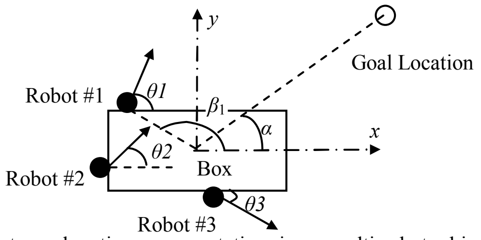

Fig.3. The state and action representation in a multi-robot object-transportation system.

The reward *r* is calculated as

$$r = D_{t_i} - D_{t_{i-1}} \tag{2}$$

where, D_{t_i} is the distance between the box center and the goal location at time t_i.

B. *Genetic Algorithms (GA)*

An approach that is particularly suitable for finding an optimal cooperation strategy for a given world state is Genetic Algorithms [8, 9]. Liu and Wu [2] used GA to realize a multi-robot box-pushing system. In GA, the solution space is represented by a chromosome space, which is a binary code, whose bits are the genes. By mimicking biological evolution, the algorithm finds the most optimal chromosome; i.e., the best cooperation strategy, which corresponds to the largest fitness function value. A typical Genetic Algorithm can be found in [2] and is described as follows:

- Initialize *p* (the number of individuals in the population), *r* (the fraction of the population to be replaced by crossover at each step), *m* (the mutation rate), and *fitness_threshold*.

- Generate p individuals at random to produce the first generation of population P
- Calculate the *fitness(i)* for each i (individual) in P
- While $(\max_i fitness(i)) < fitness\_threshold$, then repeat:
  Produce a new generation P_s:
  1. Probabilistically select *(1-r)p* members of P to be included in P_s. The selecting probability is given by: $\Pr(i) = \frac{Fitness(i)}{\sum_{j=1}^{p} Fitness(j)}$ (3)
  2. Execute the Crossover operation to generate *(rp)/2* pairs of offspring and add them to P_s
  3. Mutate m percent of the numbers of P_s
  4. Update $P \leftarrow P_s$
  5. Calculate *Fitness(i)* for each i in P
- Return the individual with the highest fitness in P

In the present work, the above algorithm is realized to find the optimal cooperation strategy between robots so that the object is transported to the goal location as quickly as possible. In Figure 3, all possible vectors of $(\theta_1, \beta_1, \theta_2, \beta_2, \theta_3, \beta_3)$; i.e., the cooperation strategies, constitute a vector space R^6, which is the set of all possible cooperation strategies between the three robots. The objective of the Genetic Algorithm is to find the best vector in R^6 under the current world state. Here, the vector $(\theta_1, \beta_1, \theta_2, \beta_2, \theta_3, \beta_3)$, which gives the force directions and locations of the three cooperating robots, is used as the individual in the genetic algorithm. In particular, it is expressed as a 30-bit binary code in Figure 4.

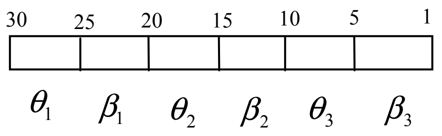

Fig. 4. The binary code expression of the individuals in the GA.

First, ten individuals are generated at random to constitute the first generation of population. Then the above Genetic Algorithm is employed to search for the best cooperation strategy.

The fitness function is calculated as

$$Fitness = F/(1+\cos\theta)^2 / |\Gamma| * S \quad (4)$$

Here, F is the net force magnitude of the three robots, θ is the angle between the net force vector and the goal location vector, Γ is the net torque, and S is the area of the triangle formed by the three robots.

The fitness function has some physical meaning. First, the cooperation strategy, which generates a high net force to point to the goal location, will receive a high fitness value so that the object will be transported towards the goal location effectively and quickly. Second, the strategy with a low net torque will gain a high fitness value because it will suppress unnecessary rotation of the object and move it quickly. Third, the one with good spacing between robots will be encouraged because the operation difficulty would be increased and the success probability would be decreased if the robots clustered in a small area. In other words, clustering of the robots is punished.

In each step of the transportation process, the GA is used to find the best cooperation strategy between the robots based on the current world state. Then it is employed to move the object for a while. Next, the cooperation strategy is searched again because the object has been moved and the world state may have been changed. The above operations are repeated until the object is moved to the goal location.

The Genetic Algorithms do not guarantee an optimal output. The crowding phenomena [8], in which very similar individuals take over a large fraction of the population, is observed in our simulation work. It greatly slows the evolution process and generates bad cooperation strategies. Therefore, the Reinforcement Learning method given in section A has to be integrated with GA to generate a more robust strategy selection mechanism in our project.

C. *Integration of learning and evolution*

Reinforcement Learning may face some difficulties in the cooperative object-transportation process. For example, the agent runs the risk that it will over-commit to actions that are found to have high $\hat{Q}$ values during the early stages of training, while failing to explore other actions that also have high values [8]. In addition, RL is unable to capture slow environmental changes. Specifically, if the environment changes very slowly, RL will not be able to distinguish the difference between the world states and as a result, it will not update the corresponding items in its knowledge base.

These limitations of RL may be overcome at least in part by using Genetic Algorithms. Because genetic algorithms simulate the biological evolution by using reproduction, crossover and mutation operations, they provide means of coping with slow environmental changes. In addition, the mutation operation in GA makes it possible to select an entirely different action from those found in early training to have high $\hat{Q}$ values through a Reinforcement Learning algorithm. Furthermore, GA provides a degree of optimality to an action.

GA also benefits from Reinforcement Learning. Integrated with RL, GA can expedite the evolution process and deal with quick environmental changes. Moreover, the crowding phenomenon and its consequences are suppressed in part.

From the foregoing observations it is clear that integration of learning and evolution may be useful for decision making related to tasks in complex environments.

The integration scheme used in the present work is shown in Fig.5:

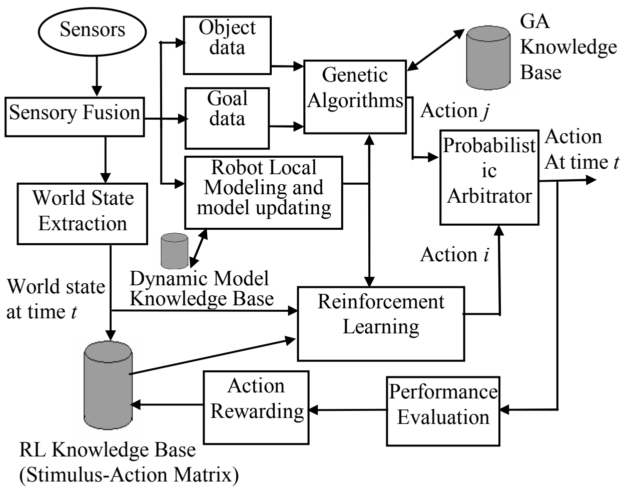

Fig. 5. The integration scheme of Reinforcement Learning and Genetic Algorithms in action selection in the multi-robot object transportation system.

In Figure 5, there are two mechanisms for making action decisions. One is based on Genetic Algorithms (GA) and the other is based on Reinforcement Learning (RL). Using this information, the arbitrator probabilistically decides on the action that will be performed.

A block representing local model generation and dynamic model updating is integrated with GA and RL. The robot force/motion sensors and the camera send their data to the sensory fusion block, which processes this information and establishes the world state model (The object and goal data). Meanwhile, the models of the robots, objects and their interactions are identified and adjusted based on their offline counterparts in the knowledge base. Then the new models are used to update the dynamic-model knowledge base, which are indispensable in the physical robot control in the low level. Moreover, RL and GA algorithms also access this knowledge base, and benefit from the current world model in their decision-making process.

V. SIMULATION AND RESULTS

Java language is used here to simulate a multi-robot object transportation system based on the architecture and learning/evolution mechanism presented in previous sections. The environment dimension is designated as 400*400 and the object is simplified as a rectangular box with a width of 60 and a height of 40. There are three robots, denoted by circles in the subsequent figures, which transport the box to the goal location. The multi-thread programming in Java is used to realize the parallel operations of robots.

Our research is expected to be applicable not only in transportation tasks with small displacements as in Figure 2, but also in more challenging tasks associated with a complex work space; for example, a planetary exploration task, in which a team of robots search a specific type of ore and cooperatively transport these objects to a spacecraft. In that case, it is not easy for an agent or a robot to possess the global information. In other words, each robot only has local sensing capability. Furthermore, it may be difficult to have a global decision-making agent that will tell the robots how to cooperate. Therefore, in the present simulation, each robot is equipped with local vision, and intelligent decision-making capability is also embedded into them although the object information is still assumed to be global.

In the beginning of the simulation, the locations of the robots, the box, and the goal are selected randomly. Because the workspace may be very large and the robots are not given any information about the location of the box, the random search strategy may be a good option. The three robots will wander in the beginning; i.e., move autonomously in random directions, to find the box. If a robot reaches a wall or another robot, it randomly changes the moving direction until reaching the next location. When a robot reaches the box, it stays there and sends a message with coordinates of the box to the other robots, who respond by arriving at the box. Then the first robot will serve as the leader in the robot team, in the subsequent operation of transporting. In this manner, the “wandering” process ends and the “transportation process” begins. The leader robot uses the mechanism of integrated learning and evolution, as outlined in the previous sections, to compute the optimal cooperation strategy and sends commands to the other robots. The command message includes the force direction and location of each robot in the team. When the other robots receive the command message from the leader, they move to the designated location and inform the leader robot that they are ready to carry out the task. This process is incorporated in the simulation presented in Figure 6.

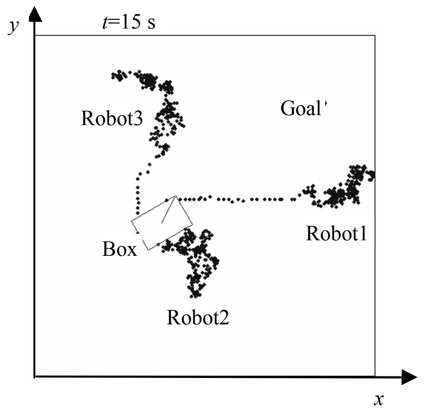

Fig. 6. The robots move randomly and find the box.

The leader robot checks the “ready” status of all robots and broadcasts the message “Begin transportation” to all the robots. The box is moved and oriented under the net force and torque applied by the robots, as determined by the action scheme. After moving the box through one step, the leader robot determines the new optimal cooperation strategy based on the current world state and sends new commands to the other robots, for carrying out the next step of transportation. This process is repeated until the box is transported to the goal location and orientation. A simulation result of the box-transportation process is shown in Figure 7.

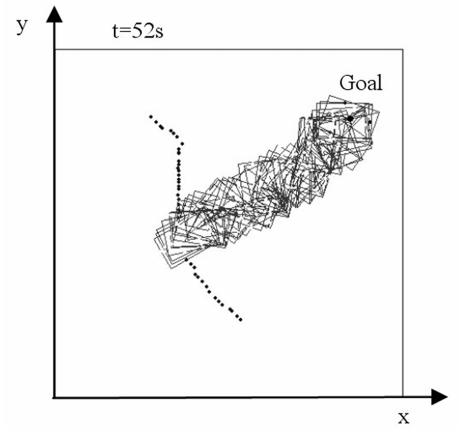

Fig. 7. A simulation result of box transportation.

Fig.8 presents the special case where the goal location is suddenly changed from *(10,380)* to *(340,60)* at *t=46*s. It is seen that the robots quickly adapt themselves to this change and select a new cooperation strategy to transport the box to the new goal location.

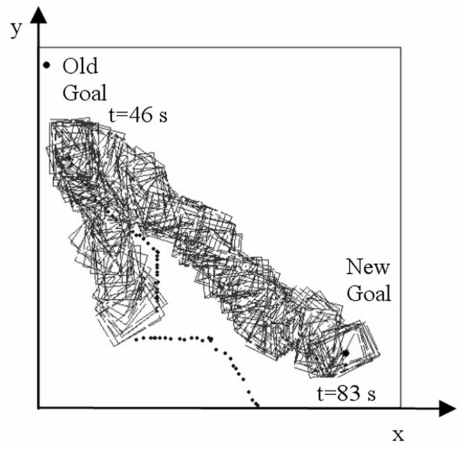

Fig. 8. Due to a sudden change in the goal location from *(10,380)* to *(340,60)* at *t=46s*, the robots quickly adjust their cooperation strategy and continue to move the box to the new goal location.

These simulation results indicate the effectiveness of the present approach of integrated learning and evolution in multi-robot object-transportation.

VI. Conclusion

This paper investigated the problem of object transportation using multiple robots. A multi-agent architecture was established to realize effective cooperation between multiple autonomous intelligent robots. Two methods of machine learning and optimization, Reinforcement Learning and Genetic Algorithms, with their characteristic advantages, were integrated to learn the cooperation strategy for achieving a common goal. The integration scheme, which makes use of an arbitrator, was able to improve the overall performance of the task. Simulation results showed the effectiveness of the approach. Future work will include the introduction of obstacles into the environment, implementation using physical robots in laboratory, and rigorous testing and evaluation of performance.

Acknowledgment

Funding for the work reported in this paper has come from the National Science and Engineering Research Council (NSERC) of Canada and the K.C.WONG Education Foundation of Hong Kong.

References

[1] G. Weiss, *Multiagent Systems*, The MIT Press, Cambridge, MA, 1999.
[2] J. Liu and J. Wu, *Multi-agent Robotic Systems*, CRC Press, Boca Raton, FL, 2001.
[3] M. J. Mataric, M. Nillsson, and K. T. Simsarian, "Cooperative multi-robot object-pushing, " in *Proc. of IEEE/RSJ Int. Conf. on Human Robot Interaction and Cooperative Robots*, Pittsburgh, PA, pp. 556-561, 1995.
[4] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots, " in *Proc. of IEEE/RSJ International Conference on Human Robot Interaction and Cooperative Robots*, Pittsburgh, PA, pp. 235-242, 1995.
[5] N. Miyata, J. Ota, T. Arai and, H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment," *IEEE Trans. on Robotics and Automation*, Vol. 18, pp. 769-780, October 2002.
[6] P. Stone and M. Veloso, "Layered approach to learning client behaviors in the ROBOCUP soccer server," *Applied Artificial Intelligence*, Vol. 4, pp. 165-188, 1998.
[7] M. Asada, E. Uchibe, and K. Hosoda, "Cooperative behavior acquisition for mobile robots in dynamically changing real world via reinforcement learning and development," *Artificial Intelligence*, Vol. 110, pp. 275-292, 1999.
[8] T. M. Mitchell, *Machine Learning*, McGraw-Hill, New York, NY, 1997.
[9] F.O. Karray and C.W. de Silva, *Soft Computing and Intelligent Systems Design*, Addison Wesley, Pearson, New York, 2004.