Multi-agent Learning of Heterogeneous Robots by Evolutionary Subsumption

Hongwei Liu[1,2] and Hitoshi Iba[1]

[1] Graduate School of Frontier Science, The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan
[2] School of Computer and Information, Hefei University of Technology
Hefei 230009 China
{Lhw,Iba}@miv.t.u-tokyo.ac.jp

Abstract. Many multi-robot systems are heterogeneous cooperative systems, systems consisting of different species of robots cooperating with each other to achieve a common goal. This paper presents the emergence of cooperative behaviors of heterogeneous robots by means of GP. Since directly using GP to generate a controller for complex behaviors is inefficient and intractable, especially in the domain of multi-robot systems, we propose an approach called Evolutionary Subsumption, which applies GP to subsumption architecture. We test our approach in an "eye"-"hand" cooperation problem. By comparing our approach with direct GP and artificial neural network (ANN) approaches, our experimental results show that ours is more efficient in emergence of complex behaviors.

1 Introduction

Genetic Programming (GP) has proven successful in designing robots capable of performing a variety of non-trivial tasks [7,11]. However, the fields' focus is almost exclusively on single-robot systems. Many tasks can be solved more efficiently when a multi-robot system is used; while some tasks cannot be solved at all with single-robot systems. Therefore, recently more and more researchers have applied evolutionary computation techniques to the design of various types of multi-robot/agent systems [3,4,5,6,8,9].

In a multi-robot system several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence, which has recently attracted much attention. It is very important and interesting to study the emergence of robots' behaviors in multi-robot systems by means of artificial evolution.

Most of the aforementioned researches are on homogeneous systems. Although D. Floreano et al. [3] presented a heterogeneous system, the relationship between the two robots is competitive. In this paper we address the issue in the context of a heterogeneous multi-robot system, in which two real robots, i.e., Khepera, are evolved using GP to solve a cooperative task.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1715–1728, 2003.

Since directly using GP to generate a program of complex behaviors is difficult, a number of extensions to basic GP have been proposed to solve these control problems of the robot. For instance, J. Koza employed GP to generate a subsumption architecture control program [7]. W.F. Punch et al. proposed an approach to solve robot navigation problems, it incorporated subsumption principles into the Echo Augmented Genetic Programming approach [12]. H. Iba et al. studied the emergence of the cooperative behavior in multiple robots/agents by means of GP and proposed three types of strategies, i.e., homogeneous breeding, heterogeneous breeding, and co-evolutionary breeding, for the purpose of evolving the cooperative behavior [4]. They used a heterogeneous breeding approach of GP, evolving a multi-agent learning system, to solve robot navigation and Tile World problems [5]. They also applied the proposed GP system to a homogeneous cooperative multi-robot system and tested their approach in an "escape problem" [8]. These researches showed that GP is efficient in multi-robot/agent learning.

We report an improvement of GP, called Evolutionary Subsumption–which combines the GP with Brooks' subsumption architecture [1] and compare our approach with direct GP and ANN approaches. Our experiments show that this method is effective in solving such complex problems of robot control.

The rest of this paper is organized as follows: in Sect. 2 we will analyse the target system and its complexity, our approaches will be presented in Sect. 3, and in Sect. 4 the experimental result with comparison of evolutionary subsumption and direct GP will be reported. Finally, discussion and some empirical conclusions are presented.

2 Task Domain and Complexity

The approaches are evaluated in an "eye"-"hand" cooperation task. In this task two heterogeneous robots learn complex robotic behaviors by cooperation. One of them, which is mounted with a digital camera, acts as the "eye" and the other, which is mounted with a gripper, acts as the "hand" (Fig. 1). Their task is: the "eye" tries to find a cylindrical object[1], and then navigates the "hand" to pick it up and then navigates it to carry the cylinder to the goal. The two robots are heterogeneous–they have different sensors and actuators, and have different roles in the system. Their behaviors are complex: including tracking, path planning, and communication, etc.

We classify the similar problems into three difficulty levels, according to the relationship of the observer–"eye" and actor–"hand":

[1] There are two cylinders in our system, one is the object that the "hand" needs to grip in the first stage and the other is the goal, which the "hand" needs to put the first object near in the second stage. In the following text, in order to ease the depiction, we use the word 'cylinder' to indicate the object or the goal according to the stage, except where we distinguish them explicitly.

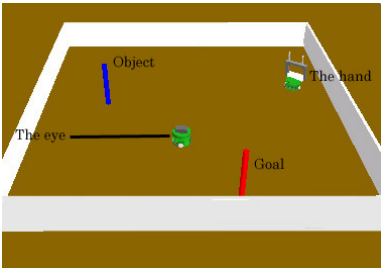

Fig. 1. "Eye"-"hand" cooperation problem

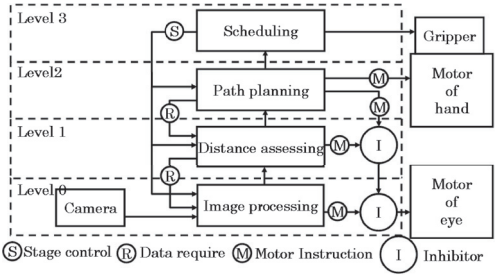

Fig. 2. Evolutionary subsumption approach's layered architecture

Difficulty 1 Fixed "Eye": the "eye" is fixed and usually acts as a bird's eye view; that is, it can see the whole environment from its fixed position. The navigation method is the most simple; but if there are obstacles in the environment it involves route selection problems.

Difficulty 2 Semi-fixed "Eye": although the "eye" can move, the relative position of "eye"–"hand" is fixed or restricted.

Difficulty 3 Unfixed "Eye": the "eye" and the "hand" can move freely, and the relative position of "eye"–"hand" is variable. Usually the "eye" can only see part of the environment.

Our target system belongs to difficulty 3. There are rather simple strategies in difficulty 1 and 2. For instance, in the "escape problem" the navigation of robots belongs to difficulty 2, the strategy is to keep the image of the button in the centre of its view field and to enlarge the image though movement, by getting closer to the object, and finally, touching it [8]. In difficulty 3 the situation is much more complicated. Since the relative position between "eye" and "hand" is variable, the "eye" must track two objects simultaneously. The search space of difficulty 3 has one more dimension than difficulty 2. Specifically, in difficulty 2, the only object that the "eye" needs to observe is fixed; but, in difficulty 3 one of the two objects, the hand, is movable. Therefore, the search space of a system which belongs to difficulty 3 is large and the emergence of robots' rational strategies is very difficult. The "eye" must select suitable viewpoints, observe the environment, and send correct instructions to "hand". Along with the moving of the "hand", the "eye" must be able to adjust its position and send new instructions according to the new situation.

3 Methodology

3.1 Design of Architecture

In our target system, the two robots need to coordinate their behaviors to achieve the goal. They have explicit division of roles and need to be synchronized by communication. This paper is concerned with how such cooperative behaviors can be established efficiently; that is, what kind of architecture should be employed and how to synthesize such an architecture. We employ the evolutionary subsumption approach and compare it with other approaches, such as the direct GP approach.

According to analysis in Sect. 2, this problem belongs to difficulty 3 and its search space is very large, it is intractable to search for a direct solution using Genetic Programming. The divide-and-conquer approach is an intuitive and efficient method when we encounter complex problems. Being a divide-and-conquer approach, the subsumption architecture decomposes the problem into a set of levels [1] and each level implements a task-achieving behavior. We employed the subsumption architecture, dividing the whole behavior into several simple behaviors. Then each level is automatically generated by Genetic Programming respectively; the lower level is formed by Genetic Programming at first, and then uses lower levels' output as nodes of the next level of Genetic Programming.

3.2 Evolutionary Subsumption

The control system is divided into 4 levels: level0 image processing, level1 distance assessing, level2 path planning, and level3, scheduling. See Fig. 2. The rest of this section will introduce each level of the architecture.

Level0 Image Processing. This level gets an input image, detects whether the "hand" and cylinders appear in the view or not, and calculates the width of their image. In order to fix our attention on the task of coordination and not immerse ourselves in the field of machine vision, we use particular colors to identify the "hand" and cylinder. See Fig. 3, input at this level is one scan line of the image and outputs are W_{hand}, D_{hand}, W_{obj}, D_{obj} (i.e., the offset from center of image and the width), and two Boolean variables B_{hand} and B_{obj}, they indicate whether the"hand" and the cylinder are within the image.

Level1 Distance Assessing. Level1 takes level0's output as its input and assesses the distance of "eye"–"hand" and "eye"–cylinder. Therefore the task of level1 is a symbolic regression problem:

$$f(W_{hand}, D_{hand}, W_{obj}, D_{obj}, B_{hand}, B_{obj}) = \{Dis_{hand}, V_{hand}, Dis_{obj}, V_{obj}\} \quad (1)$$

Where Dis_{hand} and Dis_{obj} are the assessed distances, and V_{hand} and V_{obj} indicate whether the assessed distance is valid or not. These values will be used by the higher levels. If the objects, i.e., the "hand" and cylinder, do not appear in

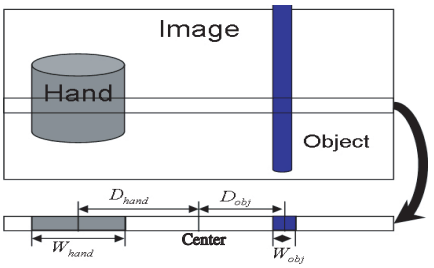

Fig. 3. Image processing approach of level0

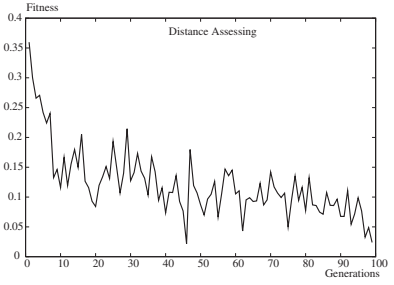

Fig. 4. Fitness of distance assessing of level1

viewfield of "eye" or are too far from the "eye" then V_{hand} and V_{obj} will be set to "False", otherwise they will be set to "True".

This level is trained separately. For each generation, before training we generate 10 maps, which randomly specify the position and orientation of the "eye", the "hand", and the cylinder. These 10 maps will be kept constant within one generation; in the next generation they will be reformed, i.e., will be different from the prior generations. The fitness is defined as the average error between the assessed value and the real value in the 10 maps.

The function set consists of F={IFLTE, PROGN2, Data_req, IFhand, IFobj}, the terminal set consists of T={W_{hand}, D_{hand}, W_{obj}, D_{obj}, $Scan$, $Const$}. The IFLTE and PROGN2 have the same functionality as in LISP and Data_req calls the level0 to refresh its output. IFhand and IFobj are based on the B_{hand} and B_{obj} defined by level0, they take two arguments and evaluate their first argument if B_{hand}/B_{obj} is true otherwise they evaluate their second argument. The $Scan$ in the terminal set makes the "eye" rotate to scan the environment, $Const$ means constant number. The result of evolution is shown in Fig. 4. After 100 generations' evolution the error is less than 0.05, this means that the average error of assessed distance in 10 maps is less than 5cm.

Level2 Path Planning. The task of this level is to generate rational motor instructions. In our approach we used central-control architecture. It generates motor instructions for both "eye" and "hand". The rational instructions for the "eye" are to get a better viewpoint and the rational instructions for the "hand" are to drive it closer to the cylinder. As we will observe in our experimental results in Sect. 4 the two robots will learn to coordinate with each other gradually and the rational strategy will emerge along with the evolutionary procedure.

Level3 Scheduling. This level determines when the "hand" should pick up the cylinder and when it should put the cylinder down. Since the procedure of this level is fixed, we can write the program for this level manually.

4 Experiments with Evolutionary Subsumption

4.1 Environment and Experimental Setting

We used Webots of Cyberbotics for the experiments. Although in simulation the robots can obtain extra information, for example the absolute coordinates etc., in order to ease the transition from simulator to real robots we did not use such information. In our experiments we only used the information which a real khepera robot could acquire through its sensor or turret.

The size of the environment is 100×100 cm with high 10 cm white walls, so that the "eye" can recognize "hand" and cylinder easily. There are no obstacles in the environment. In order to keep things simple we used special colors to identify the cylinder and the goal. The "eye" robot was equipped with a k6300 digital camera turret and the "hand" robot was equipped with a Gripper turret. There is a wireless channel through which the "eye" and the "hand" can exchange messages (Fig. 1). The initial positions of "eye", "hand", cylinder, and goal are placed randomly. The limit of steps is 2 times the linear distance between "hand" and cylinder. The actions of a robot are simplified to 4 actions: MF move forward, MB move backward, MR turn right about 30 degree and move forward, and ML turn left about 30 degree and move forward.

We used a layered training method to train each level of the subsumption architecture sequentially. As described in Sect. 3.2 we first evolve the lower levels then the higher levels. When evolving, the higher levels "subsume" the lower levels. Thus the performance of level2 is the performance of the whole system.

At the beginning of each generation we generate maps by randomly placing the "eye", "hand", and cylinders. These maps are kept constant only within one generation. They will be regenerated before evolution to the next generation. The number of maps, i.e., the fitness cases, increase with the generations from 1 to 10. That is, along with evolution the difficulty of the task is increased, finally each individual must be evaluated on 10 maps. This method is able to prevent the robots from accomplishing their task by fluke. Fitness is defined as the maximum distance in all fitness cases and the distance is between the "hand" and the cylinder after the "hand" runs out of its steps.

We used function set F={IFLTE, PROGN2, Data_req, IFVhand, IFVobj} and terminal set T={D_{hand}, D_{obj}, Scan, MFe, MBe, MLe, MRe, MFh, MBh, MLh, MRh}, where the function set is very similar to level1; in the terminal set D_{hand} and D_{obj} are the output of level1, the MFe, MBe, MLe, MRe are the motor instructions of "eye", the others are the motor instructions of "hand". The other parameters are shown in Table 1.

4.2 Result

Figure 5 shows result of level2, which plots the averaged fitness values over generations for 10 runs. Note that the horizontal line at 0.1 indicates whether the "hand" can accomplish task or not. Since the fitness function is defined as the final distance between the "hand" and the cylinder, if the fitness is below that

Table 1. Parameters of Genetic Programming

Population size	2000
Crossover rate	0.85
Mutation rate	0.1
Elite rate	0.1
Maximum depth	15

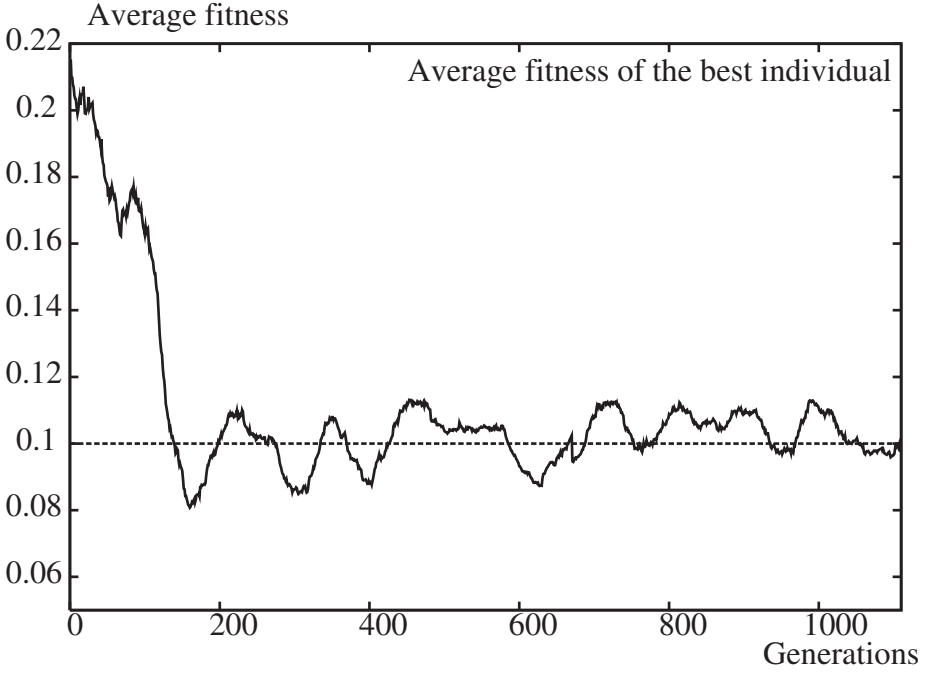

Fig. 5. Average fitness (10 runs) of the best individuals of the system

line it means finally the "hand" approached within 10 cm of the cylinder and the "hand" can detect the position of cylinder or goal, with its infrared sensors, and pick up the cylinder with its gripper, or put the cylinder down on the goal. Along with the increasing of generations the fitness cases increased from 1 to 10. This means that finally the fitness is the maximum value of 10 fitness cases. If we take this into account we can find that in this system the cooperative behaviors have been established by GP. See Fig. 8 for the emergence of cooperative behavior.

Although the concrete behaviors are various, they always display a clear strategy. The robots usually demonstrate a limited set of strategies, by analyzing the emerging behaviors we roughly classify the strategies into 3 types:

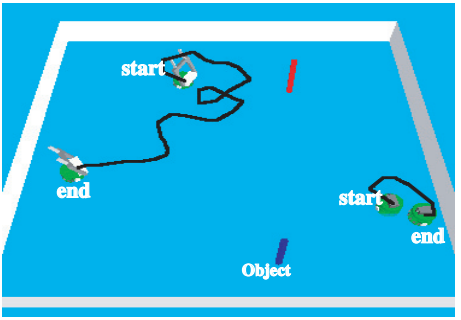

Fig. 6. At the beginning of evolution the two robots show poor coordination. Usually they move separately and aimlessly

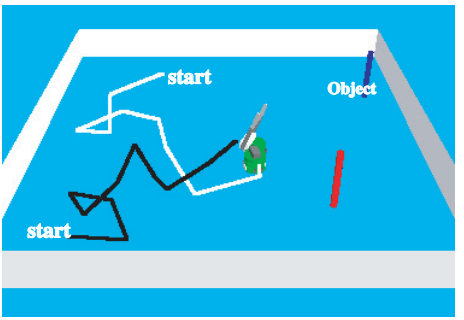

Fig. 7. They learned cooperation along with the evolution and the "observation"–"action" rhythm emerged

1. The "eye" tries to find the "hand" and then maintains its position to it; meanwhile, searching for the cylinder and finally directing the "hand" to close to the cylinder.
2. The "eye" finds the cylinder at first, moves close to it, stays near it, and then navigates the "hand" close to the cylinder.
3. The "eye" finds a suitable position neither near to the "hand" nor near to the cylinder, then it navigates the "hand" to move. After the "hand" has moved several steps the "eye" adjusts its position in response to the new situation.

   All of these strategies have the same effect, namely reduce the level of difficulty. By using such strategies the relative position of the "eye" and the "hand" becomes roughly fixed; therefore, the difficulty level is reduced from 3 to 2 (refer to Sect. 2).

Figures 6, 7, and 8 show the course of emergence of the rational strategy. At the beginning of evolution the two robots show poor coordination. Usually the "eye" and the "hand" move separately; the "hand" moves aimlessly before the "eye" surveys the environment and soon it runs out of its steps unnecessarily. Even in generation 0, there are some individuals better than others, they approach the cylinder more closely (Fig. 6).

Along with the evolution the two robots gain more skill in cooperation, they show clear rhythm of "observation"–"action"–"observation"... The "hand" never moves before the "eye" because it must save its limited steps (Fig. 7).

Finally, the two robots are more skillful. They have averaged more than 60% probability to accomplish the task. Figure 8 shows their trajectory. As shown in Fig. 8, the two robots show favorable coordination. At first the "eye" observes the environment and directs the "hand" to move and then the "eye" observes again adjusting its position and directing the "hand" to move again... We can also observe that the trajectory of "hand" is getting more and more smooth along with their interaction. These phenomena indicate that the rational strategy has emerged.

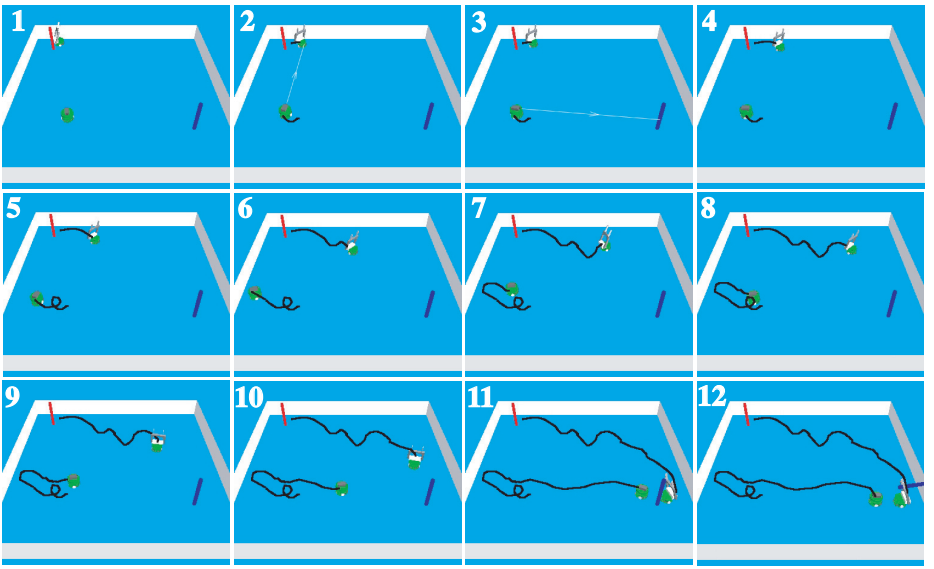

Fig. 8. Finally the skillful cooperative behaviors emerged

In our target task, due to the complexity, the two robots could not ensure accomplishment of the task in all cases. Although, along with the evolution the success rate increased. We test the success rate by the following method: for each generation we select out the best individual to test its success rate as the success rate of the generation. In the test stage we generate n maps in advance, giving the position of the "eye", "hand", cylinder, and goal as different from the training environment; but, keeping them constant to provide a fair condition to all the individuals, which are taken to test the success rate. If the "hand" can approach the cylinder within 10cm and then approach the goal also within 10 cm, then we mark the individual as able to accomplish the task on this map. The success rate is defined as the ratio of the number of accomplished maps to total maps n. In Fig. 9 the upper curve shows the trend of the average success rate along with the evolutionary process.

4.3 Comparison with Direct GP and ANN

For comparison, we also employed direct GP approaches to solve the problem. In the direct GP approach, in order to keep things simple, we did not use the input image directly; instead, we kept the level0 fixed and just used GP to generate programs for level1 and level2. The definition of fitness and the other parameters are the same as in the evolutionary subsumption approach. In direct GP approach, although the "eye" and the "hand" can find the rational strategies and produce cooperative behaviors, they take almost 3 times the number of generations of the evolutionary subsumption approach and often failed to converge

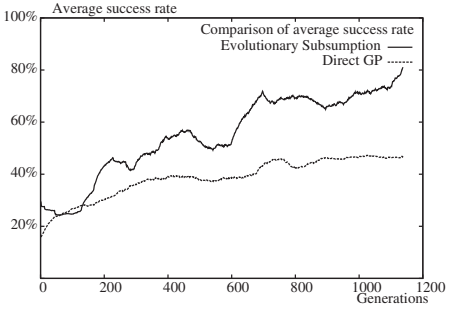

Fig. 9. Comparison of averaged success rates (10 runs) of evolutionary subsumption and direct GP

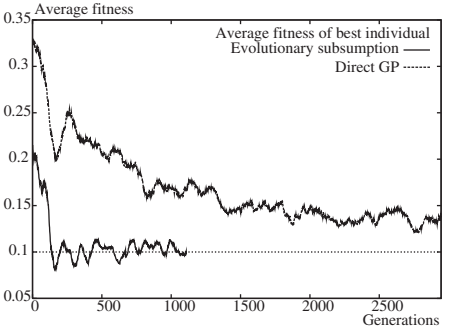

Fig. 10. Comparison of the best individual's averaged fitness (10 runs) of the direct GP approach and the evolutionary subsumption approach

due to premature convergence. Furthermore, the final fitness of the evolutionary subsumption approach is much better than that of the direct GP. Figure 10 shows the comparison of the best individual's fitness over the generations and in Fig. 9, we can find the success rate of the evolutionary subsumption approach superior to the direct GP approach. This seems due to the reasonable subsumption architecture, we have designed the suitable framework for the whole system and the GP need only search the optimal solution for each layer in a relatively small search space. On the other hand, the direct GP approach has to search in a large search space and often times it can not, or it must take a number of generations' evolution to decompose the problem into rational components. Therefore, the final results are inferior to the subsumption architecture.

In the ANN approach we used a 3 layer feed-forward neural network and used a generic BP algorithm to train the neural network. Since the BP algorithm is a supervised approach, when using the BP algorithm we have to provide a set of correct input-output pairs as training samples. However for "eye" there are too many possible strategies. It is impossible to foresee the action of "eye"; therefore, we can not provide standard input and output and use an error value to train the "eye". In other words, the reasonable strategies can not emerge automatically, they must be specified by the designer. So as an alternative we restricted the movement of the "eye": fixed the position of the "eye" and ensured that the "hand" and the cylinder both appear in its view-field. We used "batch training mode", i.e., weights are changed only after the "hand" ran out of its steps and stopped. We used the distance of this moment as the error to train the weight. Still the cooperative behaviors could not emerge within reasonable training times. Occasionally they succeeded in one fitness case, but it could not be generalized to other cases. For this approach the success rate approximates to 0%.

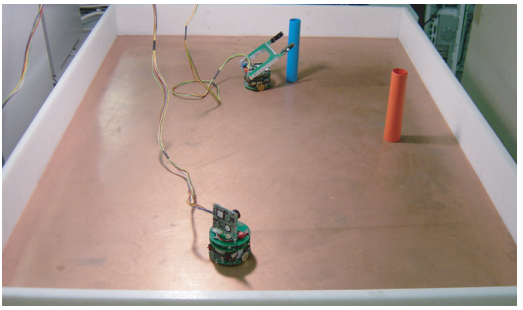

Fig. 11. The preliminary environment of the real robots

4.4 From Simulation to Real Robots

Usually the simulators are too ideal and abstract compared to the real world and gaps exist when moving from the simulator to real robots. In our ongoing experiment (Fig. 11) we employ two Khepera robots, they are connected to a desktop workstation though aerial cable. The aerial cable provides the robots with data communication from the desktop.
Since our approach employed subsumption architecture, in which the layers can be redesigned and added incrementally, when moving from the simulator to real robots we just need to redesign level0 and part of level1. Moreover, we did not use any extra information that only can be obtained via the simulator. This alleviates the complexity of moving to real robots.

5 Discussion

When designing an intelligent robot it is impossible to foresee all the potential situations a robot will encounter and specify all behaviors in advance. Especially in multi-robot systems, the situations that each robot encounters are much more complex and unpredictable. Therefore, the basic issue is how to design a control program for such systems. In this paper we studied such an issue and evaluated our evolutionary approach in an "eye"-"hand" cooperative problem. The experimental result shows that by applying GP to subsumption, our approach, efficient emergence of a rational strategy for multi-robot systems is possible.

Subsumption architecture decomposes the control system into a set of layers, each layer implements an asynchronous task achieving behavior [1]. Though it can produce a robust and flexible robot control system, the design of each layer is still a burdensome task, especially the high level layers, for instance path planning and reasoning etc. Furthermore, the robot's behavior is specified manually by adding corresponding layers, namely the reasonable behaviors can not emerge automatically just like in neural network architecture (see Sect. 4.3). In a multi-robot system it is impossible to specify the strategy of each robot in advance, the rational strategy of a multi-robot system can only emerge as a result of interaction of robots in the system.

Artificial evolutionary approaches can establish rational strategy automatically, especially in multi-robot systems. Dario Floreano et al. applied a co-evolutionary neural network in a predator-prey problem and proposed that the chase and evasion strategy would emerge automatically by the interaction of the two robots [3]. H. Iba et al. discussed the emergence of cooperative behavior for a multi-robot/agent system [4,8]. Their research showed that sophisticated strategies can be produced efficiently by the GP approach.

However, it is also intractable to search for a direct solution using evolutionary approaches for complex multistage behaviors, such as our target system. In this paper we proposed an *Evolutionary Subsumption* method, which combines the above two approaches. Compared with direct GP and classical subsumption architecture, our experimental result shows that it has been endowed with the advantages of the two approaches. By generating task achieving behaviors automatically, it alleviates the burden of design of subsumption. Furthermore, by employing an evolutionary approach, rational behaviors can emerge along with the evolution. On the other hand, when the search space is too large, subsumption architecture can restrict the search space and make evolutionary approaches converge within a practicable time.

In summary, comparing subsumption architecture, direct GP and ANN approaches:

1. It is difficult for subsumption architecture to search for a rational strategy, the behaviors should be specified in advance. Also subsumption architecture claimed that complex behaviors are simply the reflection of a complex environment, but the behaviors are specified by the designer, manually with corresponding layers.
2. Direct GP converges several times slower than the evolutionary subsumption approach.
3. BP algorithm ANN approach, due to its supervised nature, has the same problem as the subsumption architecture; that is, the cooperative behaviors can not emerge automatically.

The other essential issue for a multi-robot system is what kind of architectures should be employed, i.e., central control or distributed control. Chern H.Y. et al. employed GA to evolve a team of neural networks in a cooperative multi-robot system and studied the tradeoff between central and distributed controllers [2]. They found that co-evolutionary architecture can accelerate the convergence. Since our target system is different from the systems in literature [2, 3,8], that is the "hand" is not completely autonomous, the convergent speed of co-evolutionary architecture is slower than central-control architecture. We will report the comparison of distributed control and central control architecture in another paper.

6 Conclusion

We examined the emergence of cooperative behavior and proposed that the two robots can find several quite reasonable strategies. According to the results in

Sect. 4 these strategies are different, but have the same effect; that is by interacting with each other, the "eye" and the "hand" determine their relationship, thereby reducing the difficulty level efficiently from 3 to 2, and finally accomplishing the task. Our results lead us to deduce the following empirical conclusions:

1. Evolutionary Subsumption is efficient in emergence of a heterogeneous multi-robot system. It shows superiority to both classical subsumption architecture and GP approach. As a divide-and-conquer method this approach can be applied to a number of other robot control problems.
2. The direct GP approach can also be used to deal with some complex systems; but compared with the evolutionary subsumption approach the efficiency is much lower and easily falls into local optimum.
3. Since a standard feed-forward neural network with BP training algorithm needs samples to train the network, it is awkward in emergence of novel behavior.

However, the path of the "hand" to approach the cylinder is not optimal and the success rate is much lower for some high-reliability applications. For future research we want to extend this approach to co-evolutionary architecture and hope to improve the robustness and efficiency of our target system.

Acknowledgement. This work was partially supported by the Grants-in-Aid for Scientific Research on Priority Areas (C), "Genome Information Sciences" (No. 12208004) from the Ministry of Education, Culture, Sports, Science and Technology in Japan.

References

1. Brooks R., "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation 2(1), 14–23 (1986).
2. Chern Han Yong and Risto Miikkulainen, Cooperative Coevolution of Multi-Agent Systems, Technical Report AI01–287
3. D. Floreano, S. Nolfi, and F. Mondada, Competitive Co-Evolutionary Robotics: From Theory to Practice. In R. Pfeifer, editor, From Animals to Animats V: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior, MIT Press-Bradford Books, Cambridge, MA, 1998.
4. Hitoshi Iba, Emergent cooperation for Multiple Agents using Genetic Programming, in Parallel Problem Solving form Nature IV (PPSN96), 1996, 32–41
5. Hitoshi Iba, Evolving Multiple Agents by Genetic Programming, Genetic Programming 3, Spector, L.Langdon, W.B., O'Reilly, U.-A., and Angeline, P.A.,(eds), MIT Press, pp. 447–466, 1999.
6. M. Terao and Hitoshi Iba Controlling Effective Introns for Multi-Agent Learning by Genetic Programming, in Proc. of the Genetic and Evolutionary Computation Conference (GECCO2000), pp. 419–426, 2000
7. J.R. Koza, Evolution of subsumption using genetic programming, in F.J. Varela and P. Bourgine (Eds.) Toward a Practice of Autonomous Systems, pp. 110–119 Cambridge, MA: MIT Press. 1992.

8. K. Yanai and H. Iba Multi-agent Robot Learning by Means of Genetic Programming: Solving an Escape problem, in Evolvable Systems: From Biology to Hardware, 4th International Conference (ICES2001), pp. 192–203. 2001
9. Matt Quinn, A comparison of approaches to the evolution of homogeneous multi-robot teams, In Proceedings of the Congress on Evolutionary Computation (GECCO2001) pp. 128–135, Seoul, S. Korea. IEEE Press, 2001
10. Nolfi S. and Floreano D., Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press/Bradford Books, 2000.
11. P. Nordin and W. Banzhaf., An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming, Adaptive Behavior, 5(2):107–140, 1997
12. W.F. Punch and W.M. Rand,GP+Echo+Subsumption – Improved Problem Solving, in Proc. of the Genetic and Evolutionary Computation Conference (GECCO2000), pp. 411–418, 2000