2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Evolution of Efficient Neural Controllers for Robot Multiple Task Performance – A Multiobjective Approach

Genci Capi

University of Toyama
Gofuku Campus, 3190 Gofuku, Toyama, 930-8555, Japan
e-mail: capi@eng.u-toyama.ac.jp

Abstract— **While learning and evolution have been effectively applied to single task performance, multiple task performance still lacks methods that demonstrated to be both reliable and efficient. This paper introduces a new method for multiple task performance based on multiobjective evolutionary algorithms, where each task is considered as a separate objective function. In order to verify the effectiveness, the proposed method is applied to evolve neural controllers for the Cyber Rodent robot that has to switch properly between two distinctly different tasks: (1) protecting another moving robot by following it closely and (2) collecting objects scattered in the environment. Furthermore, the tasks and neural complexity are analyzed by including the neural structure as a separate objective function. The simulation and experimental results using the Cyber Rodent robot show that the multiobjective-based evolutionary method can be applied effectively for generating neural networks which enable the robot to perform multiple tasks, simultaneously.**

I. Introduction

Research on intelligent agents has mainly focused on evolution or learning of individual perceptual-motor and cognitive tasks. Nevertheless, intelligent agents operating in everyday life environments often are required to perform multiple tasks simultaneously or in rapid alternation, which can be a challenge even for humans and primates.

Several approaches have been proposed to address the problem of multiple task robot performance. The standard methodology in machine learning has been to break large problems into small, independent sub-problems, learn the sub-problems separately, and then recombine the learned pieces [1]. In [2], a different methodology has been used in which all the tasks are learned at the same time. Thrun et al. [3] presented a task-clustering algorithm that clustered the learning tasks into classes of mutually related tasks. In this approach, when the agent faced a new learning task, it first determined the most related task cluster then exploited information from this task cluster only. However, in all these approaches the tasks considered were similar with each other. Learning sequences of multiple decision tasks [4] or changing the agent behavior based on the environmental conditions [5] have also been undertaken.

In addition to learning, the evolution of neural controllers is well known for providing a family of naturally-inspired algorithms which can successfully address a wide range of robot behavior learning problems [6], [7]. In evolutionary robotics, different constraints and objectives are typically handled as weighted components of the fitness function [8], [9], thus applying a Single Objective Evolutionary Algorithm (SOEA). For example, Floreano evolved neural controllers for Khepera robots to perform straight movement while avoiding obstacles [8]. The average rotation speed, absolute difference between the right and left wheels, and proximity sensor readings are all included in a single fitness function. Cliff co-evolved pursuer and evader in a simulated environment [9]. While the fitness score of the evader was simply the length of time it lasted before being hit by the pursuer, the fitness score for the pursuer was more complicated. The pursuer received fitness points for approaching the evader and bonus for hitting the evader. The bonus was dependant on the timing of the collisions. As the authors noted, they had to try many weight coefficients before they arrived at a successful combination.

This article presents a novel approach for multiple task robot performance based on Multiobjective Evolutionary Algorithms (MOEAs) [10]. Unlike previous methods, in the experiments presented here, each task is considered as a separate objective function. The Nondominated Sorting Genetic Algorithm (NSGA) is used to generate the Pareto set of neural networks that tradeoff between the separate task performance. MOEAs have been successfully applied to evolve neural networks in which the architectural complexity and performance are co-optimized [11].

In this work, for the first time a MOEA is applied to evolve neural controllers for multiple task robot performance. The specific questions we ask in this study are whether: 1) MOEAs can successfully generate neural controllers for multiple task performance. 2) The evolved neural controllers optimized by MOEA in a simulated environment perform well in the step of real hardware implementation. 3) MOEAs can generate efficient neural controllers and provide information about the complexity of the tasks and environment. In order to answer these questions, in the experiments reported here, we consider the evolution of neural controllers for the Cyber Rodent (CR) robot that has to perform two different tasks: 1) protecting another robot by following it closely while it moves and 2) collecting objects scattered in the environment.

In the proposed method, we evolve one single neural controller for multiple task performance, considering information relevant to each task as the sensory input of the neural controller. As the number of tasks increases, additional sensory information related to each task must be considered, resulting in complex neural networks. This makes the evolution process difficult. In addition, the hardware implementation of evolved neural controllers may result in poor performance due to the increased error in the sensory data. In order to further investigate if the MOEA can also generate efficient neural controllers for multiple task performance, the structure of the neural network is included as a separate objective function.

II. Multiobjective Evolutionary Algorithm

A. Multiobjective Optimization Problem

In multiobjective optimization problems there are many (possibly conflicting) objectives to be optimized, simultaneously. Therefore, there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality. Consider without loss of generality the following multiobjective maximization problem with m decision variables, x parameters and n objectives:

$$y = f(x) = (f_1(x_1,..... x_m),..., f_n(x_1,..... x_m)) \quad (1)$$

where $x=(x_1,.....x_m)\in X$, $y=(y_1,.....y_n)\in Y$ and where x is called decision parameter vector, X parameter space, y objective vector and Y objective space. A decision vector $a\in X$ is said to dominate a decision vector $b\in X$ if:

$$\forall i\in\{1,...., n\}: f_i(a)\geq f_i(b)\wedge \exists j\in\{1,...., n\}: f_j(a)> f_j(b) \quad (2)$$

The decision vector a is called Pareto-optimal if and only if a is nondominated regarding the whole parameter space X. Pareto-optimal parameter vectors cannot be improved in any objective without causing degradation in at least one of the other objectives. They represent in that sense globally optimal solutions. Note that a Pareto-optimal set does not necessarily contain all Pareto optimal solutions in X. The set of objective vectors corresponding to a set of Pareto-optimal parameter vectors is called "Pareto-optimal front".

In extending the ideas of SOEAs to multiobjective cases, two major problems must be addressed: how to accomplish fitness assignment and selection in order to guide the search towards the Pareto-optimal set; how to maintain a diverse population in order to prevent premature convergence and achieve a well distributed, wide spread trade-off front. Different approaches to relate the fitness function to the objective function can be classified with regard to the first issue. The second problem is usually solved by introducing elitism and intermediate recombination.

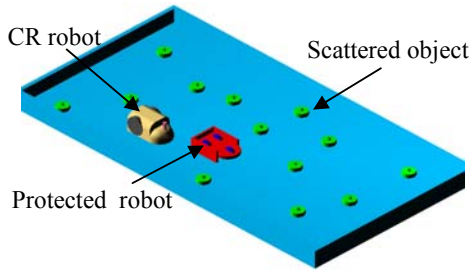


Fig. 1. Environment.

III. Multiobjective Evolution of Neural Controllers

A. Tasks and Environment

The CR robot has to learn to perform two different tasks: protecting another moving robot by following it closely; and collecting objects scattered in the environment (Fig. 1). The entire environment is a rectangle of 4m x 3.5m surrounded by walls. There are 15 green colored objects scattered in the environment. The individual life time of each agent is 700 time steps, where each time step lasts 0.1s. During this time the red color protected robot follows a rectangular trajectory with a constant velocity of 0.1m/s; at the end the protected robot returns to its initial position.

B. Neural Architecture

We implemented a feed-forward neural controller with 11, 4 and 2 units in the input, hidden and output layers, respectively. The inputs of the neural controller are the angle (A_{obj}), distance (D_{obj}) and color (C_{obj}) of the nearest object, the angle (A_{rob}) and color (C_{rob}) of the protected robot, the readings of five proximity sensors (PS_i) and the distance sensor (DS) in the front of the CR robot. The egocentric angle to the protected robot or nearest object varies from 0 to 1 where 0 corresponds to 45° to the right and 1 is 45° to the left. The value of these neurons becomes -1 when the protected robot becomes invisible or there is no object in the visual field. The proximity sensors can measure up to 0.25m, while the distance sensor varies from 0.1m to 0.8m. The proximity and distance sensor reading varies from 0 to 1, where 0 means no obstacle and 1 means touching the obstacle.

Random noise, uniformly distributed in the range of +/- 5% of sensor readings, has been added to the angle of the nearest object, the angle of the moving robot, the distance sensor, and the five proximity sensors. Because the distance to the nearest object during the experiments is determined based on the number of pixels, the random noise in simulations is considered in the range of +/- 10%. Based on the characteristics of the CR robot visual sensor, in simulations, the visual distance to the nearest object is limited to 1.2m.

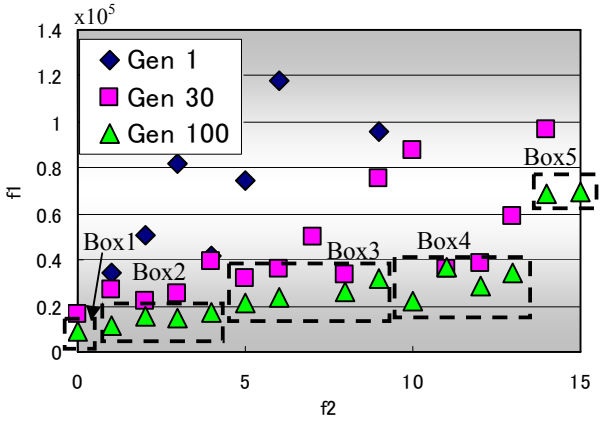


Fig. 2. Nondominated optimal solutions of different generations.

The hidden and output units use sigmoid activation function:

$$y_i = \frac{1}{1+e^{-x_i}} \tag{3}$$

where the incoming activation for node *i* is:

$$x_i = \sum_j w_{ji} y_j \tag{4}$$

and *j* ranges over nodes with weights into node *i*.

The output units directly control the right and left wheel angular velocities where 0 corresponds to no motion and 1 corresponds to full-speed forward rotation. The left and right wheel angular velocities, ω_{right} and ω_{left}, are calculated as:

$$\begin{aligned} \omega_{right} &= \omega_{\max} * y_{right} \\ \omega_{left} &= \omega_{\max} * y_{left} \end{aligned} \tag{5}$$

where $\omega_{\max}$ is the maximum angular velocity and y_{right} and y_{left} are the neuron outputs. The maximum forward velocity is considered to be 0.5 m/s.

C. Evolution

For any evolutionary computation technique, a chromosome representation is needed to describe each individual in the population. The genome of every individual of the population encodes the weight connections of the neural controller. The genome length is 52 and the connection weights range from -10 to 10. For the protecting task, the target distance d_t between the CR robot and the protected robot is considered 0.3m. In order to minimize the difference between the target and real distance, d_r, the fitness, f_1, is considered as follows:

$$f_1 = \sum_{i=1}^{max\_st} \left| d_t^i - d_r^i \right| \tag{6}$$

where *max_st* is the maximum number of steps.

The fitness of the object collecting task, f_2, is simply the number of objects collected during its individual lifetime. If an individual happens to hit the protected agent or the wall, the trial is terminated and a low fitness is assigned. Therefore, such individuals will have a low probability to survive. The following genetic parameters are used: N_{ger}=100, N_{pop}=50, σ_{shared}=0.4.

IV. Results

A. MOEA Results

In this section, we first discuss the best solutions obtained from the MOEA in terms of multiple task performance. All the simulations were performed on a Pentium 4 3.2GHz computer.

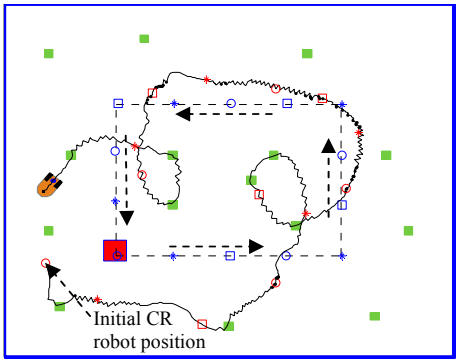
Fig. 2 shows the nondominated optimal front for generations 1, 30, and 100, averaged for five different runs of MOEA. During the first 30 generations there is a great improvement of the quality and distribution of nondominated optimal solutions. The nondominated optimal front of 100 generation has a clear tradeoff between the two objective functions. Therefore, we can choose whether to select a neural network that controls the CR robot to perform as follows: only the robot protecting task (Box 1); only the object collecting task (Box 5); or both the robot protecting and object collecting tasks by flexibly switching between them (Box 2, Box 3, Box 4).

The Box 3 neural networks induce the CR robot to simultaneously perform both tasks with the same priority, as shown in Fig. 3(a). The CR robot, while follows the protected robot, captured eight of the objects scattered in the environment. Fig. 3(b) shows that all sensory units are activated during the CR movement. The proximity and distance sensors help the CR robot to not hit the protected robot, even while it moves very close and perpendicular to the moving direction of the protected robot (around 150 and 575 steps). The Hinton diagram of the Box 3 neural controller shows that D_{obj} has strong weight connections with hidden units [Fig. 3(c)]. This leads us to the conclusion that the CR robot switches between two tasks based on the activation of the D_{obj} unit.

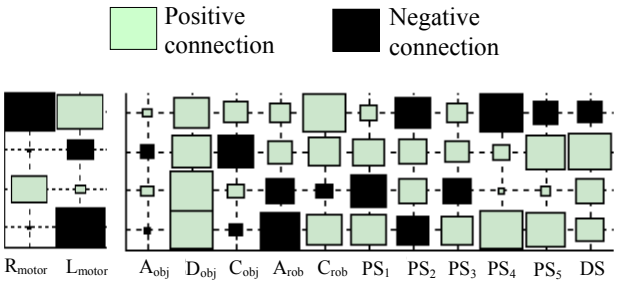
A. Hardware Experiments

We implemented the evolved optimal neural controllers on the real hardware of the CR robot, which is a two-wheel driven mobile robot. A video capture of the CR robot controlled by Box 3 neural network is shown in Fig. 4. The protected robot has a red cover with a rectangular shape in order to be detected by the visual and proximity sensors. Fig. 4 shows that the CR robot protected the moving robot, while also switching to the object collection task.

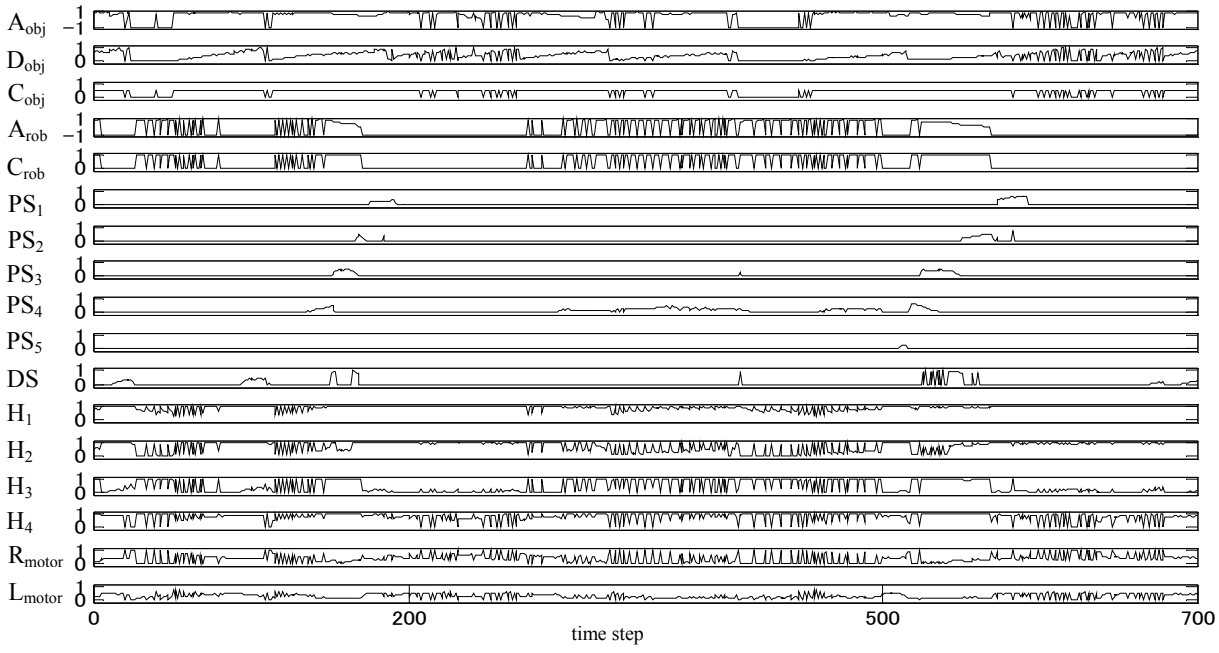
However, there are two main differences between the simulated and real robot performance. First, the distance to the nearest object utilized by the CR robot to switch from the protecting to the object collecting task was longer in the real environment.



(a)



(c)



(b)

Fig. 3. CR multiple tasks performance (Box3). (a) CR trajectory. (b) Unit activation. (c) Hinton diagram of connection weights.

This is because in some situations more than one object entered into the visual field of the CR robot, resulting in an increase in the pixel number. The other difference, observed during the performance of the protecting task, was the relative position between the CR and the protected robot. In the hardware implementation, the CR robot moves further ahead relative to the protected robot.

The reason is that in the simulated environment, the angle to the protected robot is calculated relative to its center. On the other hand, in the real hardware experiments, the angle is calculated based on the position of the red blob in the visual sensor. Therefore, during the experiments, even if the center of the protected robot is out of the visual field, half of the front of the robot is still visible. However, despite these differences, the CR still performed the multiple tasks well.

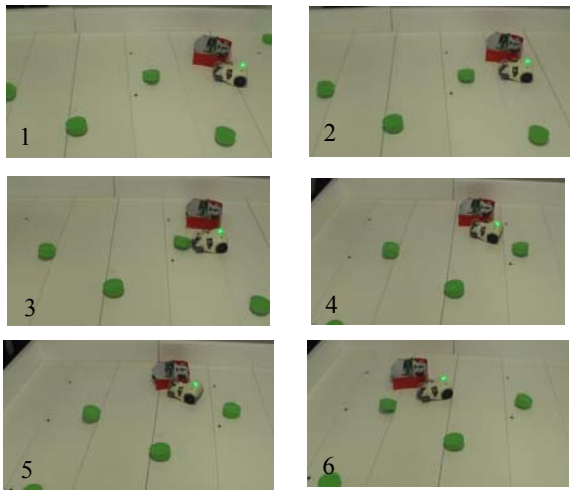


Fig. 4. Video capture of the CR robot during the experiment.

V. Neural and Task Complexity

The previous experiments demonstrated that our approach can effectively be applied to evolve neural controllers for multiple tasks execution. However, as the number of tasks increases, additional sensory units related to each task have to be considered. This results in large neural controllers and a long genome, making it difficult for the MOEA to find the Pareto optimal set. In addition, the error introduced by the sensory input units may lead to poor performance by the evolved neural controllers.

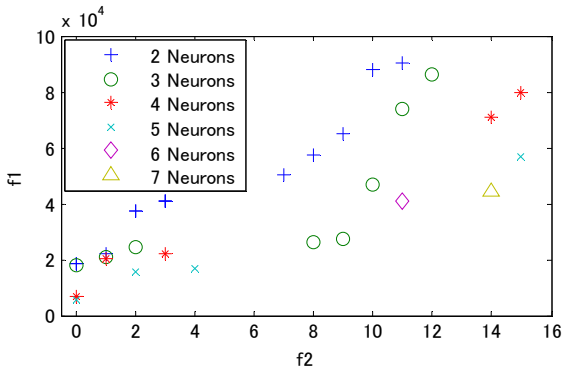


Fig. 5. Performance of neural controllers with different number of units.

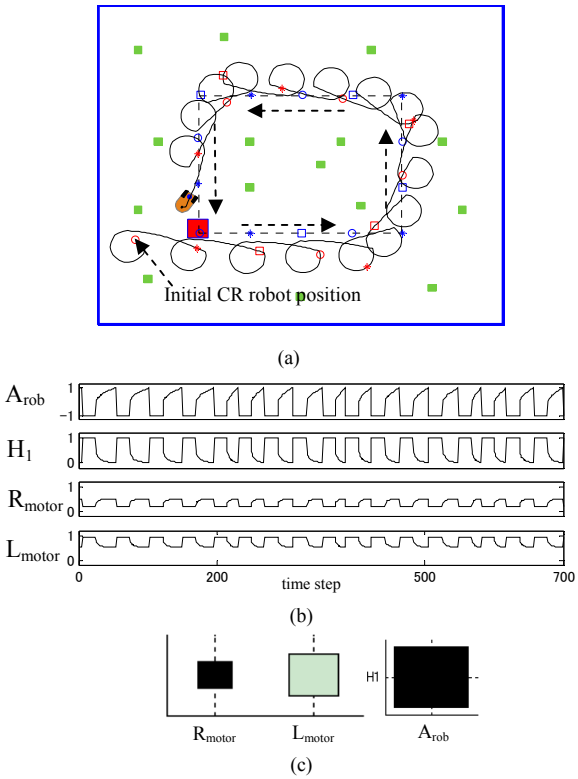


Fig. 6. Performance of neural controller with one sensory input and one hidden unit for protecting task. (a) CR trajectory. (b) Unit activation. (c) Hinton diagram of weight connections.

In the following, the results of applying MOEAs to evolve efficient neural controllers are presented. In contrast with previous approaches [12], where the fitness function of an obstacle avoidance task and the structure of the neural controller are included in a single fitness function, we considered the structure of the neural controller as a separate objective function. The complexity of the evolved neural structure generated by MOEA can also be used as an index to empirically measure task complexity.

In addition to 52 genes encoding the weight connections of the neural network, the genome encodes 15 binary genes (11 for the sensory input units and 4 for the hidden units), which indicate if an input or hidden unit exists in the network or not. Rather than using variable-length genotypes to allow for varying numbers of hidden and memory units, we use fixed-length genotypes with the maximum number of input and hidden units. This encoding method allows an input or hidden unit to evolve even if it is not active during a certain period of the evolutionary optimization process. The objective function f_3 is constructed as follows:

$$f_3 = nr_i + nr_h \tag{11}$$

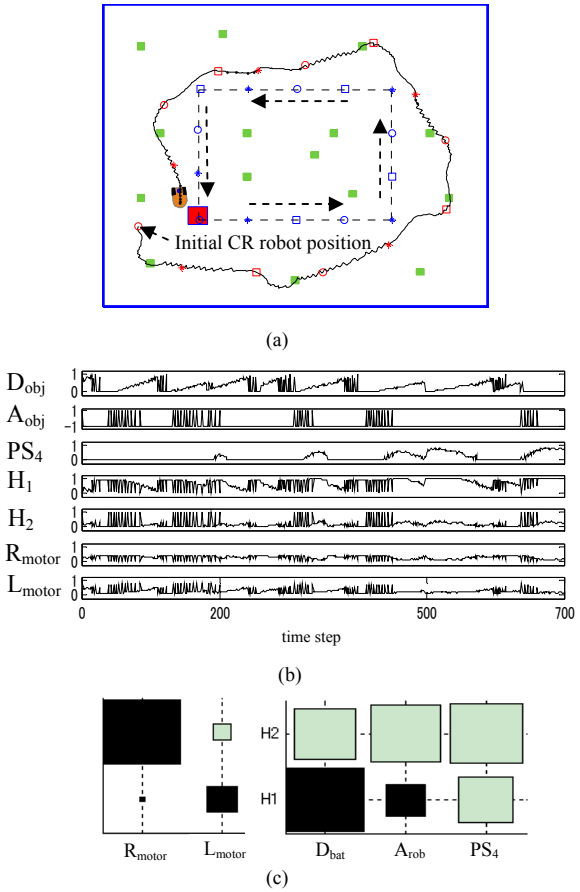


Fig. 7. Performance of neural controller with three sensory inputs and two hidden units. (a) CR trajectory. (b) Unit activation. (c) Hinton diagram.

where nr_i and nr_h are the number of input and hidden units. The minimum number of input and hidden units is considered to be two. Based on the value of binary genes, the input and hidden units are selected and the neural controller is constructed.

The graph of nondominated front solutions for different numbers of units is shown in Fig. 5. This figure shows that as the neural controller complexity increases, the solutions move to the lower-right corner, which indicates better performance. Not surprisingly, the most complex neural networks control the CR robot to perform both tasks by switching between them based on the conditions in the environment. The neural network that has seven units in the input and hidden layers controlled the CR robot to collect eight objects while working to maintain a short distance between itself and the protected robot. In addition, the number of units to complete the protection task is larger than that of the object collection task: five and four units, respectively.

In the following, we analyze the strategy employed by the CR robot to complete single and multiple tasks as the number of sensory units is reduced. Fig. 6(a) illustrates the performance of a minimal neural structure with only one sensory input (A_{rob}) and one hidden unit for the protecting task. When the protecting robot is visible the CR moves rapidly keeping the protecting robot on its left side. As the angle increases the angular velocity of the left wheel is reduced but still remains larger than that of the right wheel. Therefore, the protected robot escapes from the visual field. Fig. 6(b) shows that when the protected robot is not visible the H_1 is fully activated. Due to the positive connection between the H_1 with L_{motor} unit and the negative connection with the R_{motor} unit, as shown in Fig. 6(c), the CR rotates clockwise until the protected robot again comes into the visual field. However, the value of f_1 is nearly three times larger than that of the best neural controller with two sensory input and three hidden units (Fig. 5).

Fig. 7(a) shows a successful robot controller with only three sensory inputs (D_{obj}, A_{rob} and PS_4) along with two hidden units that switches between the two tasks and arrange to collect six objects while also following the protected robot. Fig. 7(b) and 7(c) further illustrate the robot controller's performance, showing the unit activation values and the Hinton diagram of the weight connections. When the PS_4 unit is active, due to the strong positive connections with the H_1 and H_2 units, the CR robot ignores the visible objects and just follows the protected robot. However, when the protected robot changes direction, the activation of the PS_4 unit becomes 0 and the CR switches to the object collecting task. Therefore, the CR utilized the activation of the PS_4 unit to switch between the two tasks.

VI. CONCLUSIONS

This paper has experimentally investigated the effectiveness of applying MOEAs to address the multiple task robot performance problem. In particular, it was demonstrated that in a single run of the MOEA, robust neural controllers are generated with distinctly different characteristics, ranging from performing each of the assigned tasks to simultaneously performing different tasks, by flexibly switching between them. Therefore, the user can select the most appropriate neural controller based on the task priority or the environmental conditions. Finally, we applied the MOEA to generate efficient neural controllers with a minimum number of sensory and hidden units for multiple task performance. The robustness of evolved neural controllers was also tested on the real hardware of the CR robot, using visual, proximity and distance sensors.

REFERENCES

[1] A. Waibel, H. Sawai, and K. Shikano, "Modularity and scaling in large phonemic neural networks," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp.1888-98, 1989.

[2] L. Y. Pratt, J. Mostow, and C. A. Kamm, "Direct transfer of learned information among neural networks," in *Proc. of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 584-589, 1991.

[3] S. Thrun and J. O'Sullivan, "Clustering learning tasks and the selective cross-task transfer of knowledge," In S. Thrun and L.Y. Pratt, editors, *Learning To Learn*. Kluwer Academic Publishers, 1998.

[4] S. P. Singh, "Transfer of learning by composing solutions of elemental sequential tasks," *Machine Learning,* vol. 8, no 3, pp. 323-339, 1992.

[5] G. Capi and K. Doya, "Application of evolutionary computation for efficient reinforcement learning," *Applied Artificial Intelligence*, vol. 20, no. 1, pp. 1-20, 2006.

[6] S. Nolfi, "Evolving robots able to self-localize in the environment: The importance of viewing cognition as the result of processes occurring at different time scales," *Connection Science,* vol. 14, no. 3, pp. 231-244, 2002.

[7] G. Capi, and K. Doya, "Evolution of neural architecture fitting environmental dynamics," *Adaptive Behavior*, vol. 13, no. 1, pp.53-66, 2005.

[8] D. Floreano, and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, pp. 396-407, 1996.

[9] D. Cliff, and G. F. Miller, "Co-evolution of pursuit and evasion II: Simulation methods and results", *From animals to animats 4*, pp. 506-515, 1996.

[10] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.

[11] H. A. Abbass "A Memetic Pareto Evolutionary Approach to Artificial Neural Networks", The Australian Joint Conference on Artificial Intelligence, Adelaide, Lecture Notes in Artificial Intelligence LNAI 2256, Springer-Verlag, 1-12, 2001.

[12] R. Odagiri, Y. Wei, T. Asai, O. Yamakawa, and K. Murase, "Measuring the complexity of the real environment with evolutionary robot: Evolution of a real mobile robot Khepera to have minimal structure," *Proc. IEEE Int. Conf. on Evolutionary Computation (ICEC98)*, pp. 348-353, 1998.