Robot multiple tasks performance and neural complexity

Genci Capi[1] and Koco Bode[2]

[1]Faculty of Information Engineering
Fukuoka Institute of Technology
3-30-1 Wajiro-Higashi, Higashi-ku, Fukuoka, 811-0195, Japan

[2] Faculty of Mechanical Engineering
Polytechnic University of Tirana
Sheshi Nene Tereza, Tirana, Albania

capi@fit.ac.jp

Abstract
This paper presents a new method for multiple tasks performance based on multiobjective evolutionary algorithm. 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 different tasks. Furthermore, the tasks and neural complexity are analyzed by including the neural structure as a separate objective function. Results using Cyber Rodent robot show that multiobjective-based evolutionary method can be applied effectively for generating neural networks controlling the robot to perform multiple tasks, simultaneously.

KEY WORDS
Evolutionary robotics, neural controller, task complexity.

1. Introduction

Traditionally, the 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 is a challenge even for humans and primates.

Several approaches in the literature have been proposed to address robot multiple tasks performance problem. Up to now the standard methodology in machine learning has been to break large problems into small, independent subproblems, learn the subproblems separately, and then recombine the learned pieces [1]. In addition to learning, 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 ([2], [3]). In evolutionary robotics, different constraints and objectives are handled as weighted components of the fitness function ([4], [5]), applying Single Objective Evolutionary Algorithm (SOEA).

This article presents a novel approach for robot multiple tasks performance based on Multiobjective Evolutionary Algorithm (MOEA) ([6]). Unlike previous methods, in the experiments presented here, each task is considered as a separate objective function. Nondominated Sorting Genetic Algorithm (NSGA) ([7]) is used to generate the Pareto set of neural networks.

In this method, we evolved one single neural controller for multiple tasks performance, considering relevant information of each task as sensory inputs. Therefore, as the number of tasks increases, the neural controllers become more complex. 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 sensory data. In order to further investigate if the MOEA can also generates efficient neural controllers for multiple tasks performance; the structure of neural network is added as a separate objective function.

Simulation and experimental results show a good performance of the proposed method. The non-dominated optimal Pareto neural controllers have a good distribution and CR robot behavior varies from completing each of both considered tasks to flexibly switching between them. Therefore, as a specific contribution of proposed method is that in a single run of MOEA are generated agents with completely different behaviors, making it possible to select the appropriate neural controller based on our preferences. Moreover, efficient neural controllers with appropriate sensory inputs are selected through the course of evolution.

2. NSGA

A real number NSGA was employed to evolve the neural controller. In [8], the authors compared the NSGA with four others multiobjective evolutionary algorithms using two test problems. The NSGA performed better than the others did, showing that it can be successfully used to find multiple Pareto-optimal solutions. In NSGA, before selection is performed, the population is ranked on the basis of domination using Pareto ranking. All nondominated individuals are classified in one category with a dummy fitness value, which is proportional to the population size [7]. After this, the selection, crossover, and mutation usual operators are performed.

In the ranking procedure, the nondominated individuals in the current population are first identified. Then, these individuals are assumed to constitute the first nondominated front with a large dummy fitness value [7]. The same fitness value is assigned to all of them. In order to maintain diversity in the population, a sharing method is then applied. Afterwards, the individuals of the first front are ignored temporarily and the rest of population is processed in the same way to identify individuals for the second nondominated front. A dummy fitness value that is kept smaller than the minimum shared dummy fitness of the previous front is assigned to all individuals belonging to the new front. This process continues until the whole population is classified into nondominated fronts. Since the nondominated fronts are defined, the population is then reproduced according to the dummy fitness values. As the individuals in the first front have higher fitness value, they always get more copies than the rest of the population.

3. Multiobjective Evolution of Neural Controllers

3.1 Tasks and Environment

The CR robot has to learn to perform two different tasks: protecting another moving robot by following it closely; and keeping a high level of energy by capturing the battery packs distributed in the environment (Fig. 1). The entire environment is a rectangular of 4m x 3.5m surrounded by walls. There are 15 battery packs in green color distributed in the environment. The CR robot starts in the same initial position and orientation. The individual life time of each agent is 700 time steps, where each time step lasted 0.1s. During this time the red color protected robot follows a rectangular trajectory with a constant velocity of 0.1m/s.

3.2 Neural Architecture

We implemented a feed-forward neural controller with 11, 4 and 2 units in input, hidden and output layers, respectively. The inputs of neural controller are the angle (A_{bat}), distance (D_{bat}) and color (C_{bat}) of the nearest battery pack, the angle (A_{rob}) and color (C_{rob}) of the protected robot, the sensor readings of five proximity sensors (PS_i) and the distance sensor (DS) in the front of CR robot. The five proximity sensors are angled as shown in Fig. 3. The egocentric angle to the protected robot or nearest battery pack varies from 0 to 1 where 0 corresponds to 45° to the right and 1 to 45° to the left. The value of these neurons becomes -1 when the protected robot becomes invisible or there is no battery pack in the visual field. The proximity sensors can measure up to 0.25m, while the distance sensor from 0.1m to 0.8m. The proximity and distance sensor reading varies from 0 to 1, where 0 means no obstacle and 1 touching the obstacle. Random noise, uniformly distributed in the range of +/- 5% of sensor readings, has been added to the angle of the nearest battery pack, angle of the moving robot, distance sensor and five proximity sensors. Because the distance to the nearest battery pack 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 CR visual sensor, in simulations, the visual distance to the nearest battery pack is limited up to 1.2m.

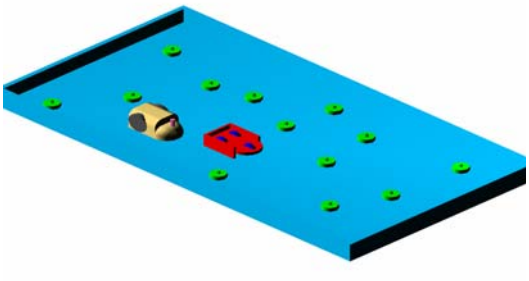

Fig. 1. Environment.

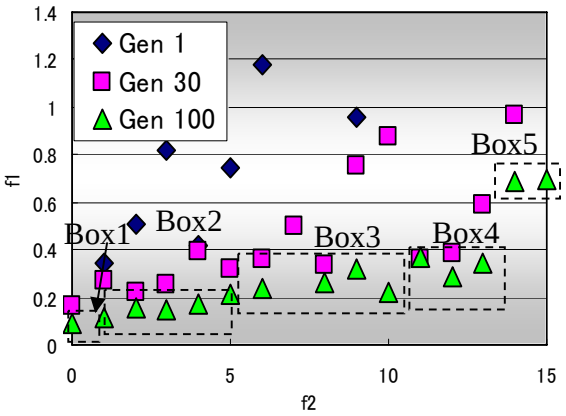

Fig. 2. Pareto optimal solutions of different generations.

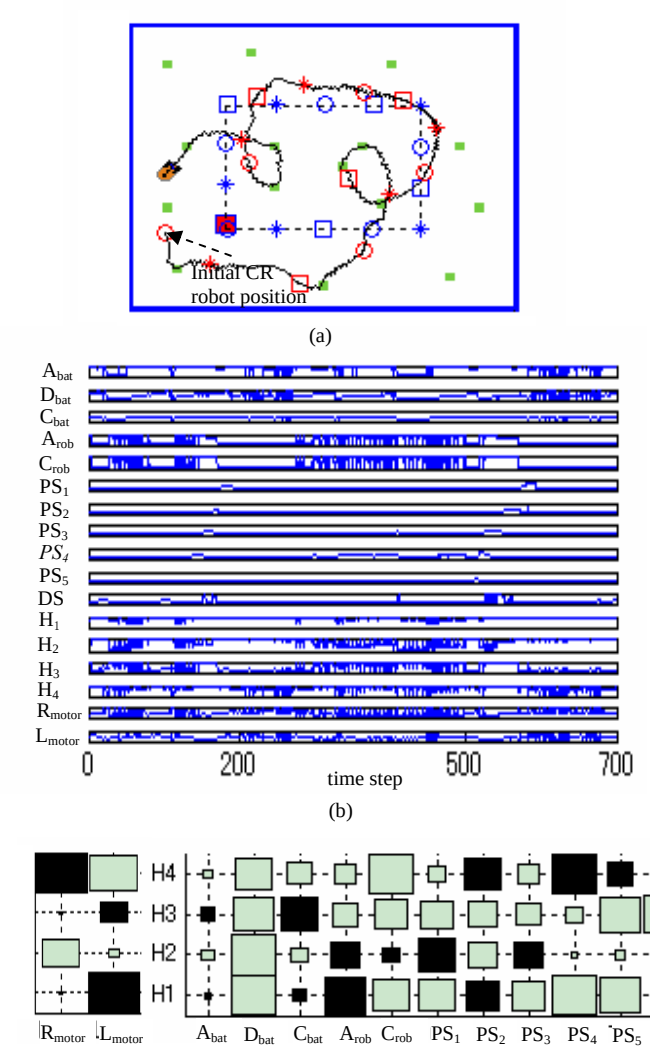

Fig. 3. CR multiple task performance (Box3). (a) CR trajectory. (b) Unit activation. (c) Hinton diagram of connection weights.

3.3 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 neural controller. The genome length is 52 and the connection weights ranged 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:

$$f_1 = \sum_{i=1}^{max\_st} \left| d_t^i - d_r^i \right| \qquad (1)$$

where *max_st* is the maximum number of steps.

The fitness of battery capturing task, f_2, is simply the number of battery packs captured during the individual lifetime. If an individual happens to hit the protected agent or the wall, the trial is terminated and a low fitness is attached. Therefore, such individuals will have a low probability to survive. The set of genetic parameters used are: N_{ger}=100, N_{pop}=50, σ_{shared}=0.4.

4. 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 in a Pentium 4 3.2GHz computer.

Fig. 2 shows the Pareto 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 on the quality and distribution of Pareto optimal solutions. From this figure, it can be deduced that MOEA is equally capable of finding the best solution for each objective when two conflicting objectives are considered, simultaneously.

The behavior of CR controlled by the neural controller of Box 3 solution is shown in Fig. 3(a). The CR robot, while follows the protected agent, captured eight of the battery packs distributed in the environment. Fig. 3(b) shows that all sensory units are activated during the CR motion. The proximity and distance sensors helped the CR robot not to hit the protected robot while it moves very close and perpendicular to the moving direction of the protected robot (around 150 steps and 575 steps). The Hinton diagram of Box 3 neural controller (Fig. 3(c)) shows the D_{bat} has strong weight connections with hidden units. This leads us to the conclusion that CR robot switches between two tasks based on the activation of D_{bat} unit.

4.1 Neural and Task Complexity

In the following, the results of applying MOEA to evolve efficient neural controllers are presented. In difference from previous approaches, where the fitness function of obstacle avoidance task and the structure of 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 evolved neural structure generated by MOEA could be also used as an index to measure empirically the task complexity.

In addition of 52 genes encoding the weight connections of the neural network, the genome encodes 15 binary genes (11 for sensory input neurons and 4 for the hidden neurons), which indicate if an input or hidden unit exists in the network or not.

The objective function f_3 is considered as follows:

$$f_3 = nr_i + nr_h \qquad (2)$$

where nr_i ,nr_h are the number of input and hidden units.

Fig. 4 shows that as the neural controller complexity increases, the solutions move to the upper-left corner, which means a better performance. Not surprisingly, the most complex neural networks control the CR robot to perform both tasks by switching between them based on the environment conditions. The neural network has seven units in the input and hidden layers and the CR robot captured fourteen battery packs while trying to keep a short distance from the protected robot. In addition, the number of units to complete only the protecting task is larger than that of battery capturing task, five and four units, respectively.

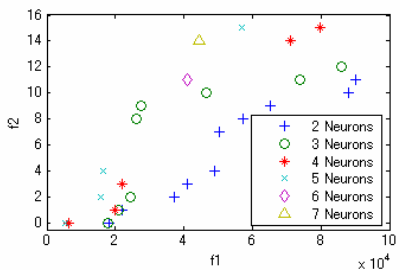

Fig. 4. Performance of neural controllers with different number of sensory and hidden units.

5. Conclusion

This paper has experimentally investigated the effectiveness of applying MOEA to address the robot multiple tasks performance problem. We considered two different tasks for the CR robot that has to protect another robot and capture the distributed battery packs. In particular, we demonstrated that in a single run of MOEA are generated robust neural controllers with completely different characteristics ranging from performing each of all considered tasks to simultaneously performing different tasks by flexibly switching between them. In addition, the MOEA generated efficient neural controllers with minimum number of sensory and hidden units for multiple tasks performance.

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] 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.

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

[4] 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.

[5] 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.

[6] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK, 2001.

[7] N. Srivinas, and K. Deb, “Multiobjective optimization using non-dominated sorting in genetic algorithms”, Evolutionary Computation, vol. 2, no. 3, pp. 279-285, 1995.

[8] A. H. F. Dias and J. A. de Vasconcelos, "Multiobjective genetic algorithms applied to solve optimization problems," IEEE Transactions on Magnetic, vol. 38, no. 2, pp. 1133-1136, 2002.