A Layered Control Architecture for Humanoid Robot

Hongwei Liu † ‡ Hitoshi Iba †
† Graduate School of Frontier Science,
The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan
‡ School of Computer and Information,
Hefei University of Technology, Hefei 230009 China
$\{lhw, iba\}$@*iba.k.u-tokyo.ac.jp*

Abstract

In this paper, we propose a layered control architecture called "CBR augmented GP" to evolve robust control programs for humanoid robots. The key idea in our approach is to represent target task with *abstract behaviors* by Genetic Programming in simplified simulation and get a prototype of the control program then interpret it with Case-Based Reasoning (CBR) in the real world environments. Accordingly, our proposed approach consists of two stages: the evolution stage and the adaptation stage. In the first stage, the prototype of the control program is evolved based on abstract behaviors in a highly simplified simulation. In the second stage, the best control program is applied to a physical robot thereby adapting it to the real world environments by using CBR. Experimental results show that this approach can generate robust control programs that can easily overcome gaps between simplified simulation and real world.

Keywords: humanoid robot, genetic programming, case-based reasoning, navigation, abstract behavior

1 Introduction

Humanoid robots are designed as companions for human beings to operate autonomously in an environment with people, so they need to adapt to noisy, cluttered environments. In order to behave like human beings the humanoid robot needs not only to perform primitive behaviors, such as walking, holding etc., but also to plan complex temporal sequences of behaviors. Therefore, how to generate temporal sequences of behaviors from primitive behaviors and accomplish complicated tasks in more challenge environments is an important issue in humanoid robotics.

From a robotic point of view, a program is the most intuitive way to control a robot, Genetic Programming is usually used to generate robot control programs automatically and has proven successful in designing robots capable of performing a variety of non-trivial tasks [1, 2, 3]. However, most researches involved simple robots, e.g., insect-like mobile robots which performed simple tasks such as obstacle avoidance, navigation or wall following, etc. Moreover, when designing a robot's controller with GP approaches, a huge number of individuals need to be tested using actual robots. Unfortunately this process can require quite a long time, which is sometimes impracticable; therefore, most of the results were implemented in simulation. But it is very hard to simulate the actual dynamic of the real world [4], namely there is reality gap between simulation and real world. So the significance of the above simulated results for physical robots still needs future study.

When a person plans to accomplish a task, he always makes high-level decisions, such as how to get from point A to point B; usually, however, he is not concerned with the details of each behavior, such as how to pass through the door or how many degrees he should turn at the corner of a corridor. The low-level decisions will be made on the spot according to the current situations of the environment.

Inspired by this cognitive insight, we propose a hierarchical approach called "*CBR augmented GP*", which divides the control system into high-level planning layer and low-level reactive layer. In this approach, we use Genetic Programming (GP) to generate program in a simplified simulation to do the high-level decisions, and then employ Case-Based Reasoning (CBR) [5] as the online adaptation mechanism in the real world environments to make the low-level decisions. Accordingly, this approach has two stages: the evolution stage and the adaptation stage corresponding to high-level planning layer and low-level reactive layer respectively and involve transference of simulated results to the real world environments.

Experimental results show that this approach can efficiently generate robust control programs; although we use a highly simplified simulation, which is rather crude, the control program can easily overcome the gaps between simulation and real world environments. Furthermore, the robot can develop new strategies according to the properties of new environments which it never encountered in simulation.

The remainder of this paper is organized as follows: in section 2 we present the framework of our approach in details, and in section 3 experiments and results for simulation and real world environments will be presented. Section 4 will conclude this paper.

2 Layered Control Architecture

Humanoid robots which have adequate degrees of freedom (DOF) can possibly perform a large variety of human-like behaviors. Namely, the human-like form of robots allows for various types of behavior, and each kind of behavior contains a number of similar behaviors to achieve the same or similar effects. For instance, we can define hundreds of "left-forward" behaviors, according to the trajectories of the humanoid robot, and define hundreds of "pick-up" behaviors, according to the size of the object and the relative position between the robot and the object. We propose a concept of "*abstract behavior*", which is a symbol and represents a set of similar concrete behaviors.

In our approach, we use GP to evolve programs with abstract behaviors in highly simplified simulation to represent the solution of target task in a high level, and then use Case-Based Reasoning [5] as the online adaptation mechanism to interpret abstract behaviors into concrete motor instructions in the real world environments. Accordingly, our proposed approach has two stages: the evolution stage and the adaptation stage corresponding to the evolution and online adaptation mechanisms respectively.

2.1 Evolution stage

Una-May et al. pointed out GP suffers from the problems of non-convergence because GP is not completely hierarchical [6]. Thus in robotics GP cannot hierarchically construct high level behaviors from very basic behaviors. GP is most often successful if it uses "high level" primitive behaviors. H. Liu et al. employed high level behaviors as the terminals of GP and has proven success in evolving complex behaviors [7]. They reported that the behaviors are robust enough to overcome the gaps between simulation and real world environments [8].

In CBR augmented GP, we employed "abstract behaviors" as the terminals of GP. Each abstract behavior is a notation, which denotes one of the similar behaviors that belong to the same kind of primitive behaviors. For instance, we defined all behaviors, which move forward while turn left as "left-forward" in spite of their trajectories and degrees (see figure 7). Similarly, we can define other abstract behaviors such as "right-forward", "forward", "side left" and "side right", etc. By defining the abstract behaviors, the variety of behaviors of humanoid robots can be reduced to a few kinds of primitive behaviors, which will be employed as terminals in GP and, consequently, reduced the complexity of GP.

By using abstract behaviors as the terminals of GP in simulation, we do not evolve all the details of the controller; instead, we use a highly simplified simulation to evolve the prototype of control program, which in fact presents the rules of control. Thus the evolution stage is also called the "rule extraction stage". The simulation does not need to accurately model the robot and its environments, only those characteristics that are relevant for the target task. As a consequence, the simulation can be greatly simplified. For instance in this paper, since the task of the robot is object carrying, we developed a very simple 2D mobile robot simulator, where the humanoid robot is treated only as a point, and the abundant behaviors of humanoid robot are reduced to nine abstract behaviors.

Since the result acquired from this stage presents only the rules of the control program, in the adaptation stage these abstract behaviors can be interpreted by CBR while running in real world environments. Thus these terminals are intelligent high level behaviors; in particular the primitives are intelligent simple CBR systems and have the ability to reason suitable behaviors. This allows adaptation to real world environments. Consequently, we call them "intelligent terminals".

2.2 Adaptation stage

2.2.1 Structure of case base

A case base is a database of cases; in this paper, the case base is organized into two layers: the first layer is an abstract layer, which contains the same abstract behaviors that are employed in the evolution stage. The second layer is a concrete layer, in which a subset of concrete behaviors for each abstract behavior and their respective specific motor instructions are defined (figure 1). Each element of the concrete layer consists of three parts: a condition, which defines the situations when this case should be retrieved; an abstract, which indicates the abstract behavior to which this case belongs; and motor instructions, which define the specifications of the concrete behavior (figure 1).

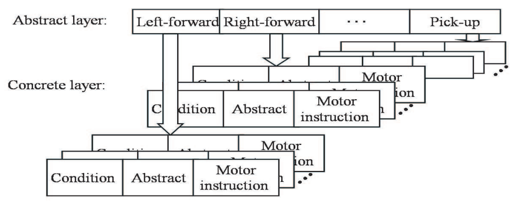


Figure 1: **The definition of case base**. The case base is organized into 2 layers. The abstract layer corresponds to abstract behaviors, which are employed in the evolution stage. The concrete layer then defines a subset of behaviors for each abstract behavior. Each case consists of three parts: the condition, under which the concrete behavior will be retrieved; the abstract, the abstract behavior to which this case belongs; and its specific motor instructions.

2.2.2 Interpreting

The result acquired from the evolution stage is presented by abstract behaviors, where every abstract behavior indicates a set of concrete behaviors that belong to the same kind. The abstract behaviors need to be interpreted into concrete behaviors when the robot is operating in real world environments. Our proposed approach employed Case-Based Reasoning to interpret the abstract behavior into concrete behavior. For each abstract behavior, the CBR system retrieves the most similar behavior from the corresponding subset of concrete behaviors based on the current situation and reuses this behavior. Thus the prototype of the control program is interpreted into a sequence of concrete behaviors, according to the real environment, which the humanoid robot encountered (figure 2).

By using CBR to interpret the prototype of the control program into a sequence of concrete behaviors, the robot can produce flexible behaviors and adapt to all the various environments it encountered.

3 Experiments and results

We employed humanoid robot HOAP-1 of Fujitsu Inc. The HOAP-1 robot was designed for a wide application in research and the development of robotic technologies. It is 48cm high, 6kg, has 20 DOFs, and is a light and compact humanoid robot (figure 6).

The CBR augmented GP approach is evaluated in an object carrying task. In this task, the humanoid robot is placed in a cluttered environment with obstacles; it must search the environment, locate and approach the object, pick it up, and then carry the object to the goal.

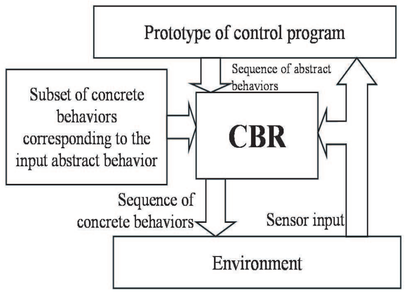
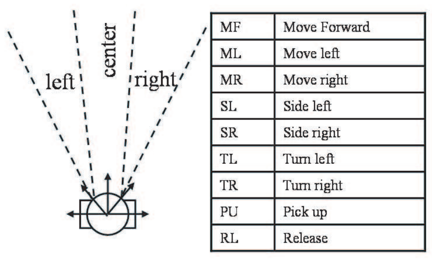

Prototype of control program
Sequence of abstract behaviors
Subset of concrete behaviors corresponding to the input abstract behavior
CBR
Sequence of concrete behaviors
Sensor input
Environment

Figure 2: **The interpreting process of CBR**. The CBR system reasons the most suitable concrete behavior from the subset of concrete behaviors which correspond to the input abstract behavior according to the current situation. Thus the output sequence of behaviors is flexible for the currently encountered environments.

Although this problem has been widely studied, a number of researchers have implemented this task successfully with mobile robots [9, 2], but, to our knowledge, there are few researches involving humanoid robots. However, as companions for human beings, carrying an object from one place to another is the most important, common task. Yet with a humanoid robot, the solution to this problem is very difficult due to the redundancy of a humanoid robot's behaviors. Therefore, generating human-like behaviors in the context of the object-carrying problem, is always an important issue in humanoid robotics.

3.1 Results of the evolution stage

As aforementioned, the task of the evolution stage is rule extraction, namely, in our target task, to generate strategies of navigation. We defined 9 abstract behaviors (figure 3), and utilized vision as the only sensor, the field-of-view of a simulated robot is shown in figure 3). The settings and main parameters of GP are shown in table 1.



MF	Move Forward
ML	Move left
MR	Move right
SL	Side left
SR	Side right
TL	Turn left
TR	Turn right
PU	Pick up
RL	Release

Figure 3: The definition of abstract behaviors. In this paper, according to our target task we defined 9 abstract behaviors

The robot is allowed to move 100 steps to find the object and then carry it to the goal. The fitness function measures the individual's performance in three aspects: whether the robot can accomplish the task (or the final distance between the robot and the object if failed), the total steps the robot takes to achieve the task, and the number of collisions.

Table 1: The parameters of GP

Population size	1000
Crossover rate	0.85
Mutation rate	0.1
Elite rate	0.01
Maximum depth	10
Initial depth	3
Function set	progn2, if_no_obs_left, if_no_obs_right, if_no_obs_center, if_obj_near, if_obj_far, if_obj_left, if_obj_right, if_obj_center
Terminal set	The abstract behaviors

We generate 100 maps, i.e., the positions of robot, object, goal and obstacles for each generation, and every individual must be evaluated in terms of these 100 maps. The maps remain fixed within the same generation, but vary between generations. Since the maps of real world environments are unpredictable, this measure can produce more generic strategies than one using fixed maps. Consequently, the fitness is defined as eq1.

$$f = \sum_i (w_1 \times dis_i + w_2 \times step_i + w_3 \times collision_i) \quad (1)$$

where dis_i means the final distance between the object and the robot, $step_i$ denotes the remained steps when the robot accomplished the task, $collision_i$ means the number of collisions and w_1, w_2 and w_3 are the weights of the three aspects of performance.

Figure 4 traces one typical trajectory of a simulated robot, in which the circle O denotes the object and the circle G denotes the goal. From the trajectory, we can see that in simulation, the robot is able to select a rational path to approach the object while avoiding obstacles. Figure 5 shows the curve of average fitness (thin line) and of best fitness (thick line).

It can be observed from these results that the navigational strategies have been established, and the simulated robot can accomplish the task by using a rational strategy with fairly high success rate.

3.2 Results of the adaptation stage

The humanoid robot is equipped with a digital camera used as the sensor. Machine vision might

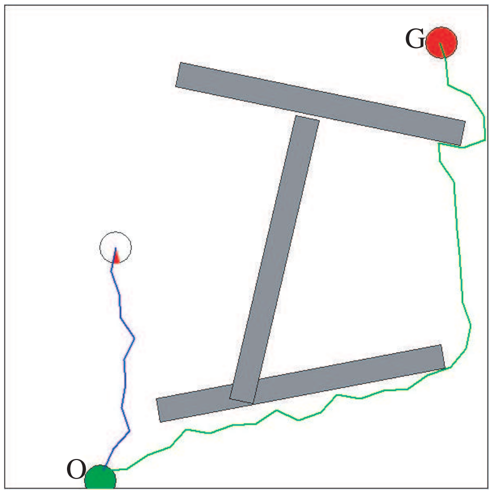


Figure 4: **The result of evolution stage**. This figure shows one example of the simulated robot's trajectory

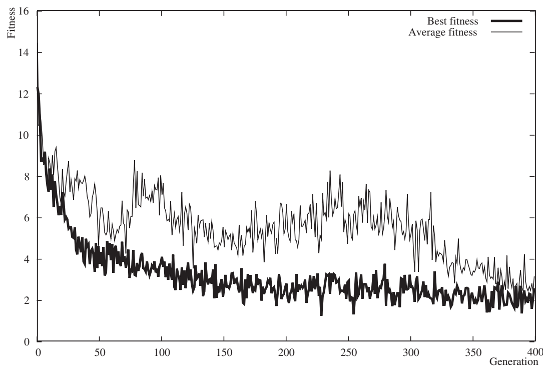


Figure 5: **The fitness of evolution stage**. This figure shows the curves of best fitness (thick line) and average fitness (thin line) over generations.

make substantial contributions to flexible adaptation, but, in order to fix our attention on the task of object carrying and not immerse ourselves in the field of machine vision and image understanding, we use particular colors to identify the object, goal and obstacles. As shown in figure 6 (upper panel), the robot hangs his head to see the area in front of his feet; thus we divide the image field into 6 sections. The horizontal 3 sections indicate left, center and right respectively; the vertical 2 sections represent distance: far or near.

We used the upper borders of obstacles to represent their orientation and position, e.g., line AC in figure 6 (lower panel), and used orientation and position of obstacles to define the conditions of cases[1]. Since, in the evolution stage, we used 9 abstract behaviors, the case base has 9 subsets. For each subset, we defined a set of concrete behaviors based on the orientations, positions of obstacles.

[1]Only the cases of picking up and releasing involve the positions of object or goal; other cases used only obstacles' positions.

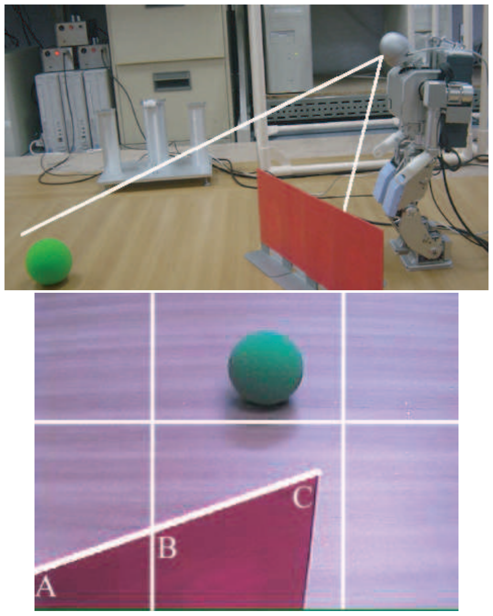


Figure 6: In our experiments the humanoid robot bows its head to observe the area in front of its feet (upper panel). We divide the image (lower panel) into 6 sections, indicating the locations of left, center and right respectively, and distance of far or near. Line AC presents the orientation and position of the obstacle, and the orientation and position of this obstacle are used to define the condition of case.

As described in section 2.2.2, in the adaptation stage we used CBR to interpret the program, which was acquired from simulation, and adapt it to the real world environments. The most important step of CBR is the retrieval process. For each input image, we calculate the orientation, position and length of obstacles in it, and then compare with the one stored in the case base to evaluate the similarities. Concretely, the orientation, position and length of line segments in each section of image are taken into account when calculating similarity. The most important one of these three elements, however, is the orientation, because it represents the relation between the robot and obstacles; thus we assign different weights to these three elements. The similarity can be calculated from the following formula (eq2):

$$s = w_1 \times \Delta l + w_2 \times dis + w_3 \times \Delta ang \qquad (2)$$

the three elements Δl, dis and Δang denote the differences of length, distance and angle between the currently input image and the one stored in case base previously.

After retrieving the most similar case from case base, we then reuse this case according to current situation; thus the abstract behaviors are interpreted into the concrete behaviors according to current environmental conditions. This mechanism is usually referred to as adaptation, which enables the robot to adapt flexibly and dynamically to real world environments.

For example, we defined several "move forward" behaviors for the abstract behavior "MF", from 0 step to 6 steps. The most suitable concrete behavior will be retrieved by Case-Based Reasoning mechanism according to the current circumstance. Similarly, we defined cases for other abstract behaviors (refer to figure 7).

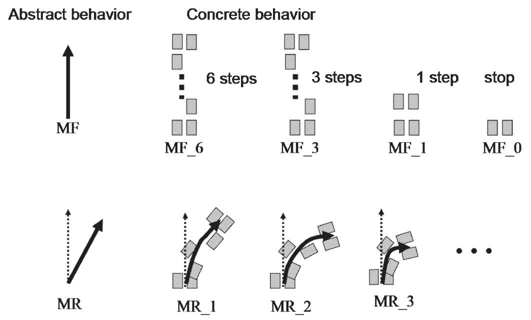


Figure 7: Examples for definition of concrete behaviors (the blocks represent footprints of robot)

In the real world, If we directly apply the best program to the physical robot without the CBR mechanism, the robot will completely fail. The robot did not appear to be able to reproduce the strategies acquired from simulation. We observed from experiments, under this situation, that the robot soon collided with an obstacle and lost its balance. This is not surprising, because the real world is entirely different from the simulation.

Figure 8 shows a typical run of physical robot with Case-Based Reasoning mechanism[2]. We can observe from these clips that the robot reproduces the strategies of the simu-lated robot. It follows the obstacles while avoids bumping to them, and approaches to the object finally picks it up. During the running process, the robot can adjust the number of steps and orientation of its primitive behaviors according to the concrete circumstance, therefore the robot demonstrates adaptive and flexible behaviors and exhibits high robustness to position of obstacles and object[3].

Although we used highly simplified simulation, which is only a coarse mimic of real environments, by employing these two mechanisms, the humanoid

[2]Videos of results are available at http://www.iba.k.u-tokyo.ac.jp/~lhw/humanoid/

[3]Since the two stages of our target task–picking up and release object–are similar, we only showed first stage in figure 8

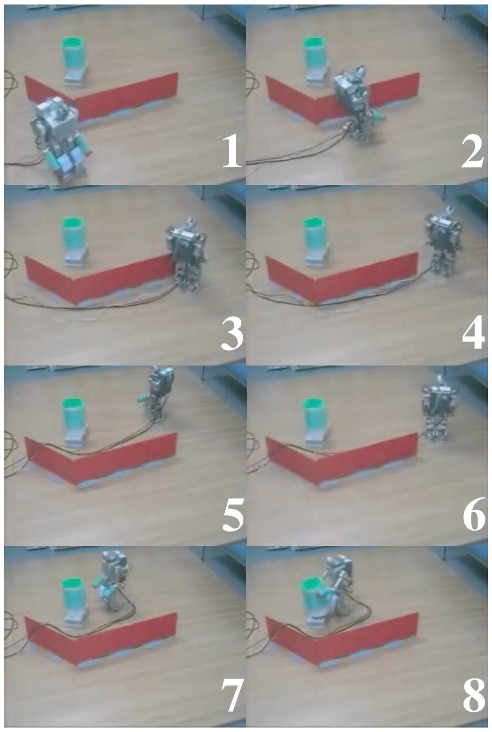


Figure 8: A typical run with Case-Based Reasoning. From this figure we can see that the robot repeated the strategies acquired from simulation.

robot can overcome the gaps between the simulation and the real world and flexibly adapt to real world environments.

4 Conclusion

In this paper, we employed CBR augmented GP to program the humanoid robot automatically. We used a simplified simulation to generate a prototype of the control program, and then the control program was interpreted by CBR to adapt it to the real world environments. Experimental results showed that this approach can generate robust control programs for humanoid robots as well as dynamically adapt to the real world environments. Our results lead us to deduce the following empirical conclusions:

1. The CBR augmented GP is efficient in developing rational control strategies and programming for humanoid robots; the resulting systems are robust in response to the variation of environments.

2. The simulation can be highly simplified, and the adaptation mechanisms can overcome the gaps between the simplified simulation and real world environments.

3. Since GP is not completely hierarchical, employing high level primitive behaviors can improve the adaptability and robustness of robot.

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] J. R. Koza. Evolution of subsumption using genetic programming. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life. Towards a Practice of Autonomous Systems*, pages 110–119, Paris, France, 11-13 Dec 1992. MIT Press.

[2] Nolfi S. and Floreano D. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press/Bradford Books, 2000.

[3] Peter Nordin and Wolfgang Banzhaf. Real time control of a khepera robot using genetic programming. *Cybernetics and Control*, 26(3):533–561, 1997.

[4] R. Brooks. Artificial life and real robots. In *European Conference on Artificial Life*, pages 3–10, 1992.

[5] Janet Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers Inc., 1993.

[6] Una-May O’Reilly and Franz Oppacher. An experimental perspective on genetic programming. In R Manner and B Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 331–340, Brussels, Belgium, September 28 - 30 1992. Elsevier Science.

[7] Hongwei Liu and Hitoshi Iba. Multi-agent learning of heterogeneous robots by evolutionary subsumption. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1715–1728, Chicago, IL, USA, 2003.

[8] H. Liu and H. Iba. Multi-agent learning by evolutionary subsumption. In *2003 Congress on Evolutionary Computation (CEC 2003)*, pages 1115–1122, Canberra, Australia, 2003. Springer-Verlag.

[9] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.