International Journal of Robotics and Automation (IJRA)
Vol. 1, No. 1, March 2012, pp. 31~48
ISSN: 2089-4856

Towards Behavior Switch Control for an Evolutionary Robot Based on RL with ENN

Jingan Yanga[1], Yanbin Zhuang[2], Chunguang Li[3]
1Changhzou Key Laboratory of Software Technology & Applications, Jiangsu Prov., P. R. China and School of Computer & Information Sciences, Hefei University of Technology, Hefei 230009, Anhui Province, China.
2.3. School of Computer & Infomation Engineering, Changzhou Institute of Technology, Changzhou, Jiangsu Province, China.

Article Info

Article history:

Received Feb 6, 2012
Revised Mar 12, 2012
Accepted Mar 21, 2012

Keyword:

Behavior switching
Evolutionary robotics
Evolutionary neural network
Reinforcement learning
Robotic adaptability
Simulated binary crossover
Simulated robotic agents

ABSTRACT

This paper proposes a behavior-switching control strategy of an evolutionary robotics based on Artificial Neural Network (ANN) and Genetic Algorithms (GA). This method is able not only to construct the reinforcement learning models for autonomous robots and evolutionary robot modules that control behaviors and reinforcement learning environments, and but also to perform the behavior-switching control and obstacle avoidance of an evolutionary robotics (ER) in time-varying environments with static and moving obstacles by combining ANN and GA. The experimental results on the basic behaviors and behavior-switching control have demonstrated that our method can perform the decision-making strategy and parameters set optimization of FNN and GA by learning and can escape successfully from the trap of a local minima and avoid "motion deadlock" status of humanoid soccer robotics agents, and reduce the oscillation of the planned trajectory between the multiple obstacles by crossover and mutation. Some results of the proposed algorithm have been successfully applied to our simulation humanoid robotics soccer team CIT3D which won the 1st prize of RoboCup Championship and China Open 2010 (July 2010) and the 2nd place of the official Robo Cup World Championship (2011) on 5-11 July, 2011 in Istanbul, Turkey. As compared with the conventional behavior network and the adaptive behavior method, the genetic encoding complexity of our algorithm is simplified, and the network performance and the convergence *rate* ρ have been greatly improved.

Corresponding Author:

Jingan Yanga
Changhzou Key Laboratory of Software Technology & Applications, Jiangsu Prov., P. R. China and School of Computer & Information Sciences, Hefei University of Technology, Hefei 230009, Anhui Province, China.
Email: jayang@mail.hf.ah,cn

1. INTRODUCTION

1.1. Evolutionary Robotics

Evolutionary robotics has been an active research area that applies artificial evolution to construct control systems for autonomous robots. This is also a promising methodology for the autonomous development of the robots, in which their behaviors are obtained as a consequence of the structural coupling between a robot and environment [37][26]. In evolutionary robotics, a robot's perception-action loop is determined by a control mechanism that is artificially evolved [31]. Many works have proposed evolutionary robotics control systems using evolutionary adaptations of artificial neural network [25][8][21][1], genetic programming [32][14], and a learning classifier system [5]. Artificial neural networks are commonly employed to evolve a robot controller [11][29], and an evolutionary algorithm is used to design and/or train ANN for solving a given task. Many attempts for robot control and an evolutionary algorithm have focused

on developing autonomous robots inspired by animals and humans that have robust adaptation and stable behavior in changing environments [16][33]. Recent researches have put even more emphasis on the fitness functions used in evolutionary robotics [24][27][4]. In order to enable the robots to operate normally in unforeseen and dynamic circumstances, the robots must have the ability to evolve their behaviors. In this paper we establish the modules for obstacle avoidance behavior, goal approach behavior, and behavior-switching control based on combination of artificial neural network (ANN) and genetic algorithm (GA) which allow the robots itself to sense their environments and have self-adaptable function.

1.2. Evolutionary Reinforcement Learning and Adaptation of Behavior Control

In a natural system, animals do not only adapt to environmental changes, but they can also accumulate adaptations. They can store "knowledge" about a previously encountered environments and use it to alter their behaviors when faced with a specific environments again. This process is called learning when it occurs in a lifetime and evolution when it occurs in a lineage [7]. In other words, the agent adapts itself appropriate for different environments by evolving different behaviors. Robot learning is a dynamic continuous mechanism. ER is one of a host of machine learning methods that rely on interaction with, and feedback from, a complex dynamic environment to drive synthesis of controllers for autonomous agents [24]. Only by constant study can the robot improve better its own adaptability, and2 acquire knowledge by relying on constant interaction with the dynamic environments. Finally, the robot can learn to move in the unknown environment by repeatedly adjusting the environmental module and its own module. This is what Brooks called behaviorism thought [36][34]. He thought that the effective way to design an intelligent robot should not be like traditional artificial intelligence, i.e., completely based on symbol inference "top-down", but should be like the evolutionary mechanisms of the biological organisms who use "bottom-up" approach based on perception-action, and learning by interacting with the environments. In this work, we designed a simulation environment of robot navigation and an evolutionary robot module. The strategies for obstacle avoidance and the approach of the robot's navigation [15] in the unknown environments are regarded as the robot's basic behavior. The ability of the robotics to jointly apply many basic behaviors to its environments is called combined behavior. The process that the robot activates some basic behavior at some moment is called behavior switching. The robot's behavior and its behavior switching are controlled by using FFNN (Feed Forward Neural Network) and PNN (Probabilistic Neural Network) [10][30][6][13][22]. ANN-based controllers consist of many basic network modules. Each module is a subnetwork with the standard unified structure. This method may effectively reduce the network computing complexity and the encoding complexity, and also speed up the evolutionary rate. We then directly carry on the encoding evolution to various modules based on genetic algorithm. This evolutionary process may be regarded as reinforcement learning process of the robots.

Finally, we demonstrate by the results of the simulated experiments that these novel modules do not require the complete environmental knowledge, and their structures are simple and can be easily extended to more complex structures, because the robots have a self-adaptive ability to learn from their experiences and the environments.

1.3. Main Challenges in ER and Contributions

One of the main challenges in ER is to discover and to model different adaptation mechanisms. Most of the works in ER consider the artificial evolution of neuro controllers as one of these adaptation mechanisms [26].

It is considered a feasible methodology to develop autonomous agents that can reveal conscious abilities. Artificial evolution differs from other learning schemes because it works on a population of different individuals based on a selectionist approach, rather than a goal directed one.

Thus, major contributions of this paper are summarized as follows: (1) proposed a behavior-switching control strategy of an evolutionary robotics based on ANN and GA for implementing robotics navigation in the unknown environments; (2) designed and built behavior-switching learning modules of an evolutionary robot; (3) the proposed algorithm can escape successfully from the trap of a local minima, jumps over the region containing a local minimum point by crossover and mutation, and perform behavior switching and obstacle avoidance effectively by evolutionary reinforcement learning; (4) the genetic encoding complexity is simplified using the neural network modules; (5) the proposed algorithm has better performance in convergence speed, solution variation, dynamic convergence behavior, and computational efficiency than the path planning method based on the real-coded genetic algorithm with elitist model. This behavior-switching controller can easily be extended to other applications by adjusting control parameters of ANN and GA and physical constraints.

2. RL-ENN AND EVOLUTIONARY ROBOTICS

2.1. Reinforcement Learning witn ENN

The objective of reinforcement learning is to solve decision-making tasks through trial and error interactions with the environment. The reinforcement learning [20] is referred to learning for agent to map the environment state to action in order to maximize the total reward and obtain the the maximum cumulative reward value of state-action pair. Formally, this is defined as a Markov decision process using the state space S, action A, and the rewards R.

Definition 2.1 A Markov decision process is denoted as ⟨S,A, r, γ, p⟩, where S=(s1, s2, · · · , sm) is the state space, A=(a1, a2, · · · , an) is the action space, r:S×A→R is the reward function of the agent, γ ∈ [0, 1) is the discount factor which reflects the notion that rewards depreciate by factor γ<1, and p: S×A→Δ is the transition function, where Δ is the set of probability distributions over state space S.

The closer the discount factor γ is to 1 the greater the weight is given to future reinforcements.

Definition 2.2 A fitness function is a particular type of objective function that prescribes the optimality of a solution (for example, a chromosome in a genetic algorithm) so that the particular chromosome may be ranked against all the other chromosomes. Optimal chromosomes, or at least chromosomes which are more optimal, are allowed to breed and mix their datasets for producing a new generation that will hopefully be even better.

Definition 2.3 For any policy π and any state s, the value of policy π in state s is denoted as V (s) which is the expected discounted cumulative reward the agent gets and is formulated as follows:

$$V^{\pi}(s) = \sum_{k=0}^{M} \gamma^{k} E[r_{t+k} / \pi, s_t = s] \quad (1)$$

Definition 2.4 For any policy π and any state s, the expected return of taking action a in state s and following policy π afterward is denoted as Q(s, a) which is the expected discounted cumulative reward the agent gets and is defined as follows:

Definition 2.5 The immediate reward, for all states in the state space is defined as:
1) The reward for non-terminal states: r = 0.
2) The reward for terminal states:
   a) The reward for the goal state: r = 1.
   b) The reward for invalid states: r = 0.
   c) The reward for obstacle states: r = −1.

In state s∈S, an action a∈A is executed. The agent experiences a state transition st→st+1 and obtain a reward rt+1 ∈ R. The objective of the agent is to maximize its reward Rt defined by [28]

$$R_t = \sum_{i=t+1}^{T} r_i \quad (2)$$

where T is the time reaching the final state. The states s0, s1, · · · , sT is called an episode.
The decision of which action to take in a certain state is determined by the policy

$$\pi(s, a) = p(a|s) \ \forall s \in S \quad (3)$$

which denotes the probability of taking action a in state s. Q-function for a policy π is defined as

$$Q_{\pi}(s, a) = E_{\pi}\{R_t / s_t = s, a_t = a\} \quad (4)$$

which denotes the expected reward of taking action a in state s and following policy π afterward.
The task of a reinforcement learning robot is to learn a policy π : S→A for selecting actions based on current observed states π(st) = at. Learning the optimal policy π∗ for producing the greatest cumulative reward over time is denoted as follows:

$$\pi^{*}(s) = \arg\max_{a \in A}[r(s, a) + \gamma V^{*}(\delta(s_t, a_t))] \quad (5)$$

where the optimal action in state s is a that maximizes the sum of the immediate reward r(s, a) and the value (discounted by γ) of following the optimal policy. The δ(st, at)(next state given st and at) is defined as a state

transition function. V ∗ is the maximum discounted cu mulative reward that the agent can obtain starting from state δ(s, a).

When an agent is at some environmental state, the reinforcement learning method does not inform the agent to adopt the corresponding correct actions, but the selected action's quality is evaluated by using the reinforcement signal provided by the environment, and the optimal strategy is obtained through trial-and-error unceasingly to get best mapping from state to action. The reinforcement learning structure and the state-action pair are shown in Figure 1.

The environment for an agent is described as the state set: S = {si|si∈S}, the action set that an agent performs is expressed as: A = {ai|ai∈A}. Under the current state si, an agent selects action ai and perform it. At the moment the state st transfers to st+1, and obtains the reinforcement signal rt from the environment. Task of agent's reinforcement learning is to obtain the maximum cumulative reward by using a control policy π.

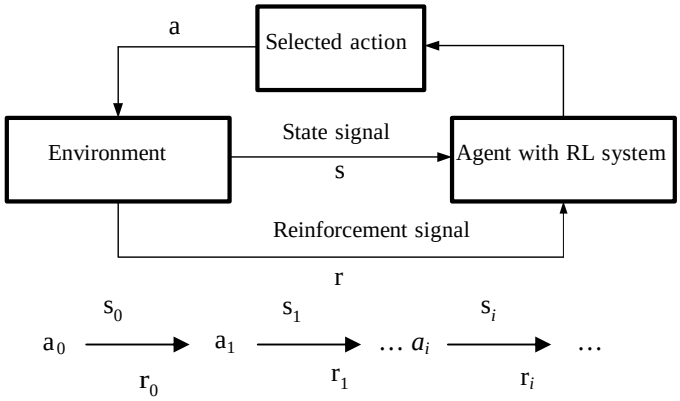


Fig. 1. The block diagram of reinforcement learning structure and state-action pair for robot behavior control.

Assume that an agent is a point robot with simplified motor actions: forward, left, right, and backward. All actions can be tried in all states. The robot world and its state of transitions are regarded as a function of the present state and action taken. The task of the robots is to reach a given goal state via the shortest path. For reinforcement learning robots, a reward function given any current state, next state, and action, st, st+1, and a, is given by equation (1).

$$r_{s_t,s_{t+1}} = \begin{cases} 0 & \text{if st+1 = st} \\ 1 & \text{if st+1 = given goal state} \\ -1 & \text{if st+1 = st (obstacle state)} \end{cases} \quad (6)$$

The negative numerical reward in equation (1) discourages agents attempting an action against the world boundary. This action does not change the state of the environment.

The basic idea of reinforcement learning is on1ine learning which combining together the control process and the learning process.

2.2. Evolutionary Neural Network

The hierarchical network architecture of ENN is designed based on the biological evolutionary genetic algorithm. The network connection weights, the network topology, the excitation function of each node in the network, and learning rules can be evolved for performing the global search widely in all solution space for network weight training until the optimal solution of the network architecture with least mean square error between the network output and goal output of the given training set are found and for avoiding the problem to being trapped in a local minimum caused by gradient descent learning. The evolution of learning rules can be regarded as a process of "learning how to learn" in ANN where the adaptation of learning rules is achieved through evolution [37].

2.3. Evolutionary Robotics Based on Reinforcement

Learning with ENN Evolutionary robotics (ER) is a new technique for the automatic creation of autonomous robots and a subfield of behavioral robotics. It is concerned with the application of evolutionary computation methods to the area of autonomous robotics control systems. One of the central goals of ER is to develop automated methods that can be used to evolve complex behavior-based control strategies.

In the design process for an intelligent robot, some key questions are: 1) how to realize the "behaviorism" idea; 2) how to learn the behavior and actions [9] by interacting with the environments; 3) how to acquire knowledge from the environments and their experiences; 4) how to control the topological evolution trend of network modeling and complexity in network model efficiently.

Among the learning methods, reinforcement learning seems to be most suited for agent-based applications for supporting the self-improvement of a population of solutions [28]. The reinforcement learning is to learn what to do, how to map situations to actions so as to maximize a numerical reward signal and is usually based on the idea that the robot receives rewards (feedback signal), which are positive or negative scalar values, based on its performance [3][2][38][19]. Depending on the reward, the agent reinforces (positive reward) or decreases (negative reward) its confidence on the correctness of its current behavior. The rewards are given by an external coach or by an evaluation function (reinforcement function) internal to the robot controller. The robot chooses one act in the environment. The condition and state are changed after the environments accept this action. One reinforced signal reward (or penalty) is created simultaneously and is fed back to the robot. The robot chooses next action, again according to the reinforcement signals and the current environmental state. The principle of the choice is to make the probability of receiving a positive reward increase. The action chosen not only affects the current reward value, but also affects the environmental conditions and states for the next choice, as well as the final reward value [23][17]. A meaningful combination of the principles of neural networks, reinforcement learning, and evolutionary computation is useful for designing agents that learn how to solve a complex task and adapt to their environment through interaction with the environment.

Here, we consider a mobile robot actuated by using left and right two caterpillar bands. Each caterpillar band has four options: forward, left, right, and backward. Their codes are: 0=forward, 1=left, 2=right, 3=backward. The behaviors and actions of a robot are implemented by changing speeds of the left and right wheels based on reinforcement learning neural network. Figure 2 shows neural network architecture including the 8 infrared sensors data inputs, 5 hidden neurons with sigmoid activation function pi, and two output neurons representing wheels commands. Encoding of the states and the actions are illustrated in Table 1.

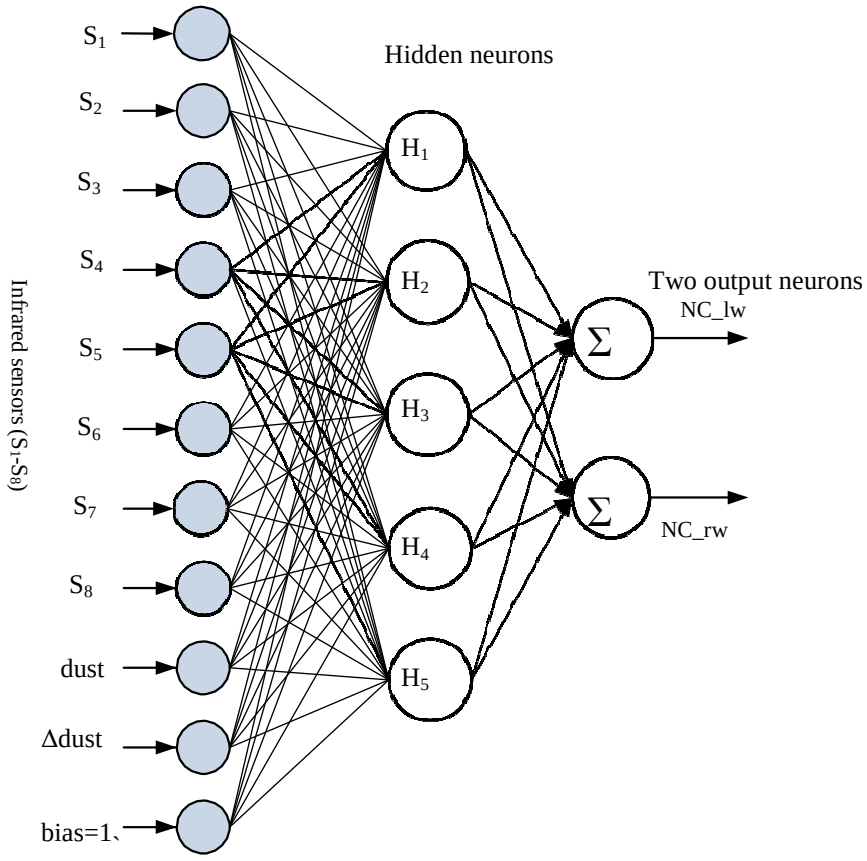


Fig. 2. Neural network architecture for the obstacle avoidance using neural modelling of the robot behaviors.

Table 1. Encoding of the states and the actions

The states	0	…	14	15
Encoding of the states	0000	…	1110	1111
The actions	forward	left	right	backforward
Encoding of the actions	00	01	10	11

2.4. Relationship Between GA and Simulator

The relationship between the genetic algorithm (GA) and the simulator is illustrated in Figure 3. Obviously, the simulator and the GA have the different functions. The GA is responsible for transferring genotype to the simulator, and decoding genotype as a neural network and/or input sensor information (phenotype). The phenotype mainly describes an organism's traits and the organism's genotype describes its genetic makeup. Two organisms can have the same phenotype but have different genotype if one is homozygous dominant and the other is heterozygous.

Suppose the fitness values of individuals 1, 2, · · · , n be f1, f2, · · · , fn respectively, then the probability of an individual selected for reproduction based on roulette wheel selection (see Fig.3) is calculated by following equation[12]:

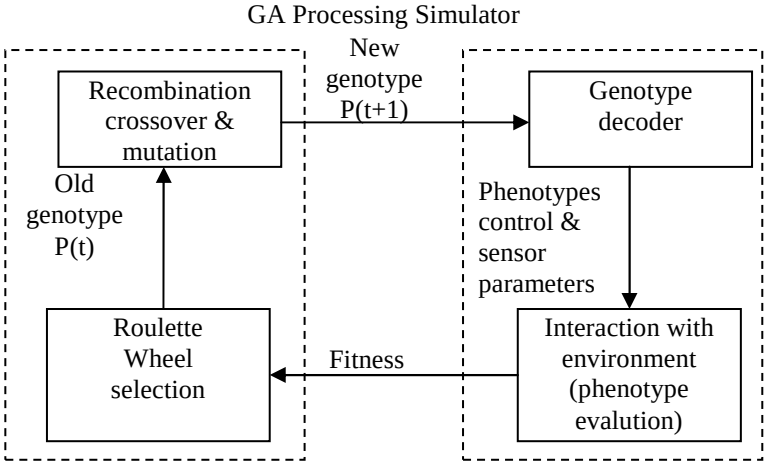


Fig. 3. Relationship between genetic algorithm and simulator

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{7}$$

The simulator uses this information to execute simula tion process, and transfers the simulated results, namely, the robot fitness values, to the GA processing. The interactive process between the robot and the environment is detailed as follows:

```
Fitness = 0
t = 0
while (t ≤ Nul) do
reads the sensor inputs
t = t + 1
compute and return to Fitness
```
(8)

where t is the length of a time step, and Nul is the upper limit of the simulation running time steps. The fitness function computation has many different options according to the different experiments. Those above are the basic steps of only one simulated operation. For each generation of the GA, each individual in the population has to pass the simulated process above. In the interactive process with the environments, reading the sensor inputs is for computing the output of control network, then for moving and updating positions of a robot according to the output of control network.

3. OBSTACLE AVOIDANCE BEHAVIORS

We have designed the behaviors for avoiding movable obstacles which get nearer from various positions and directions using the behavior network and the proposed method.

3.1. Fitness Function for Obstacle Avoidance

The fitness function is the heart of an evolutionary computing application [24]. The fitness function is responsible for determining which individuals (controllers in the case of ER) are selected for reproduction. If there is no fitness function, the individual is randomly selected. Successful evolution of the autonomous robot controllers is ultimately dependent on the formulation of suitable fitness functions that are capable of selecting successful behaviors. The performance for each evolved controller is evaluated by using the different fitness functions.

Therefore, for the autonomous robots interacting with the environment, they are given an external reward or penalty (r = rt) in some state st. For obstacle avoidance behavior r = −1 upon collision with an object and r = 0 otherwise. In order to get a optimal policy, the reward function should be fixed. For example, a robot that cannot avoid an object soon become immobilized when its path is blocked, the robot controller would obtain a negative fitness value.

The internal reinforcement signal r∗ is derived that represents the immediate reward assigned to the robotics system in terms of correctness of the actions executed so far and also is the prediction error of the total reward sum between two successive steps:

$$r*(t) = r + \gamma V(\vec{x}t+1) - V(\vec{x}t) \quad (9)$$

where r represents the direct effect of action on the transition, $\gamma V(\vec{x}t+1) - V(\vec{x}t)$ is the estimation of the improvement of states. The r∗ is used as the error to train the neural networks mentioned above.
If the desirability of a state is associated with a certain action $Q(\vec{x}t, at)$, the equation (5) for the prediction error becomes:

$$r*(t) = r + \gamma Q(\vec{x}t+1, at+1) - Q(\vec{x}t, at) \quad (10)$$

where the parameter γ ($0 \leq \gamma \leq 1$) determines the present value of the future rewards. Main differences between two reinforcement learning approaches are to calculate 1) state evaluations $V(\vec{x}t+1)$ and 2) state-action evaluations $Q(\vec{x}t+1, at+1)$.

3.2. Network Architecture and Encoding

An autonomous robotics can acquire signals from the goal sources using approach sensor for locating the goal sources, and then should approach and reach the goal sources as soon as possible. This behavior is just opposed to obstacle avoidance behavior. The network architecture design of the approach behavior is also relative to the network architecture design of obstacle avoidance behavior. Behavior control of the robots is performed by using a module neural network. The whole network consists of multiple neural network modules (g0-g7).

All modules have a similar topological architecture including two input nodes, one hidden layer with two nodes, and one output node. Reasonable selection of the node number in the hidden layer should synthetically take complex degree of the network architecture and permissible errors into account. If the number of nodes in the hidden layer is too few, the network may not be trained at all or the network performance gets very poor.

If the number of the nodes in the hidden layer is too many, although this can reduce the errors of the network system, but may lengthen training time of the network at the same time. On the other hand, the training of neural network is easy to fall into the local minimum not to be able to obtain the optimum solution which also is one of main reasons for over-fitting in neural network training and learning.

Our preliminary simulation experiments have also proved that the performance of the crossover mapping method was not satisfied. When a robot goes directly forward in free space, in order to turn to examine some goal source, it will reduce the speed of the motor on the side of the sensor approaching. Therefore, our algorithm should directly map the left sensor input into the left motor, and the right sensor input into the right motor, i.e., only connection weights (w01 between g0 and g1, w03 between g0 and g3, w21 between g2 and g1, w23 between g2 and g3, w45 between g4 and g5, w47 between g4 and g7, w65 between g6 and g5, and w67 between g6 and g7) are remained. The whole network structure of the approach sensor is similar to that of the touch sensor. But the main difference is that the input of the approach sensor makes use of the difference between the left sensor signal and right sensor signal (See Figure 3.4.1). In the experiments, we not only use a chromosome with 56 bits (6×8+8=56 bits), but also add 6 bits to evolve the

related sensor parameters: 3 bits are used for encoding the exploratory ranges of the sensors, and the other 3 bits are used for encoding the intervals between the sensor ends. A ”ci” in Figure 4 represents the encoding of wij (j=1,3,5,7).

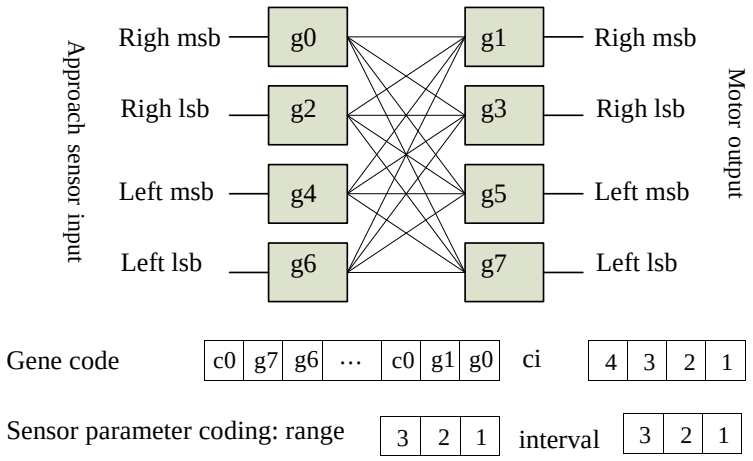


Fig. 4. Network architecture and encoding of approach behavior control of the robot.

3.3. Parameter Setup for Obstacle Avoidance

Here, the GA algorithms apply a single-point crossover to artificial chromosomes with 56 bits. The algorithm uses a special bit mutation pattern so that only one bit in the chromosome turns over. First we choose one chromosome to execute mutation based on the mutation rate, and then stochastically choose one bit to turn over in the selected chromosome. Therefore, mutation rate that must be divided by the chromosome bits can be equal to the mutation rate normally used in a GA algorithm.

Elitism was implemented by finding the fittest individuals n in the current generation and copying them across into the new generation before the breeding process has begun. Elitist selection always preserves the fittest individuals from the population to the next generation. Therefore, elitist selection increases a convergence rate of GA, but elitist selection may lead to premature convergence in the GA search process being trapped in local optima since the inherited best individuals might not be close to the global optima. For example, if too many individuals are preserved from one generation to the next or if a ’super’ individual (an individual with a far better fitness than the rest of the population) occurs, then elitism can seriously increase the chances of the best individual being rapidly replicated within a few generations and filling the whole population with identical ’siblings’. Once this happens, the population will tend to stagnate to a local minima. Our studies have shown that one way to overcome this is to take a higher mutation rate. The effect of a low mutation rate on a population reflects few variations available to respond to sudden environmental changes. This means the species is slower to adapt. A higher mutation rate may damage more individuals, but by increasing variation in the population could increase the speed at which the population can adapt to changing circumstances. The Elitist selection here has been embodied in the following aspects:

1) if the robot does not do anything, accumulate 0 score;
2) if the robot goes directly ahead, but does not attempt to avoid any obstacle, and finally crosses the environment boundary, then the robot would accumulate one negative score which takes 0 score;
3) if the robot can correctly perform obstacle avoidance by learning, it can accumulate one positive score;
4) if the robot can not only complete obstacle avoidance, but also go forward at full speed in the free space, then it has a higher positive score.

The simulation steps is setup 1500 so that the robot can accumulate one high fitness value in its evolutionary process, and also learn obstacle avoidance behavior correctly. The weighted factor of collision punishment takes a value of 10 (determined after many experiments). If this factor value is too low because walking distance is far greater than the collision punishment value, some individuals which give a poor performance may be regarded as the best individuals, and may be involved in the heredity operation in the next generation. When this factor value is too high, the pressure on the robot to execute obstacle avoidance behavior is rapidly increased, and so the useful heredity material in the evolutionary process may be lost.

In all experiments to improve the efficiency of the simulation, the size of the population was 40. These simulation experiments have demonstrated that the use of a large population does not enhance the

performance of the system. The upper limit of the simulation evolutionary operation takes 40 generations. Another reason is to save time. We observed from our experiments that a community's average performance enhancement is limited after that community has evolved about 30∼40 generations.

3.4. Simulated Results for Obstacle Avoidance

The simulated experimental results for obstacle avoidance in the different environments and analysis of the results are is illustrated in Figure 5. As well as changing the crossover rate and the mutation rate, the simulation experiments also contained two different environments: one contained 4 obstacles and the other contained 10 obstacles. The experimental results are illustrated in Figure 5(a) and Figure 5(b), respectively.

Evolutionary time for an environment containing many obstacles should be longer. The more obstacles, the more less free-motion opportunity and the collision times will increase. This leads the robot to increase its accumulated negative score possibilities. In the environment containing 10 obstacles, the production of the best individual will take 10 generations. However, in the environment containing 4 obstacles, this evolutionary process needs only 4 generations. The robot's evolutionary performance become worse in the environments containing more obstacles, because the straight lines of highest score values that obtained by the robot have less opportunity to move forward. Slightly increasing the crossover rate and mutation rate may make the convergence rate slow down. But 40 generations' evolution later, the average experimental performance may be accepted. Too high a crossover rate can reduce the GA performance, since its main aim is to have a higher destruction rate to gain the highest fitness value pattern. Too high mutation rate has a remarkable effect on GA performance [38]. In some situations, the system continues to operate for only 10 generations more, and the system performance may get worse. However, the convergence rate and the quality of the solutions are greatly improved by combining parallel GA and crossover mapping.

4. IMPLEMENTATION OF BEHAVIOR SWITCHING CONTROL STRATEGY

4.1. Neural Network Encoding Based on GA

Neural networks are well suited for training with evolutionary computing-based methods because they can be represented by a concise set of tunable parameters [30]. A wide variety of neural network structures have been used. The most common of these are layered feedforward or recurrent network architectures. The behavior-switching controllers of ER based on the network modules proposed in this paper is accomplished by combining ANN and GA.

The basic idea of behavior switching control strategy is that behavior-switch controller should be activated immediately and switched to obstacle avoidance behavior and follow wall behavior once the touch sensor finds the obstacles, and the ERL systems have to change robots' moving direction rapidly based on the basic behaviors and jump over the region with the local minimum point, and escape from the trap of local minimum and plan their paths to goal again. Furthermore, the genetic algorithms are used to evolve network connection weights for providing a global search method for network weight training and for avoiding the problem to being trapped in a local minimum caused by gradient descent learning.

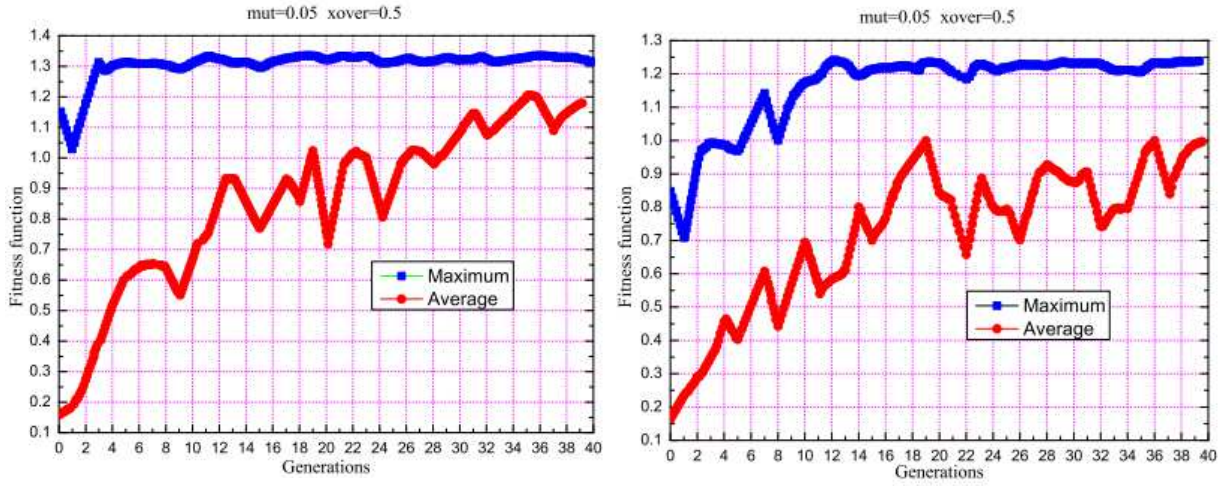
This is very important to multi-robot system and the simulated and real robotics soccer games. The approach behavior should also be activated when the robot is in the free space. For the basic approach behavior and approach sensor parameters are effective, we can use the GA to evolve the switching network of the basic behaviors directly.

This experimental environment contained only 10 obstacles and 10 goal sources. This is because the basic behavior and the sensing parameters embedded in the robot evolve on the basis of this environment. In our experiment, positions, sizes, and shapes of the obstacles, and the positions of the goal sources are variable.

The numbers of obstacles and the goal sources are fixed at 10. The robot's design is basically the same as the one described above, but the approach sensor parameters are fixed, and no longer evolve by the GA encoding. This new network structure is illustrated in Figure 6. The chromosome includes 11 module networks (g0–g10)(each module network has 6-bit encoding; every weight has 1-bit encoding). Therefore, the module networks and their connection weights need 6×11+8×2+4×2+4=94-bit encoding in al1. The first 8 module networks are almost the same as the module networks in the several sections above. The main difference is that the output is not directly connected to the motor outputs, but is the input of the second cell network composed by the other two module networks (See Figure 7). The output of the module network 11(g10) determines the basic behavior that will be activated. The output of 0 represents obstacle avoidance, and the output of 1 means approach behavior.

4.2. Parameter Setup of GA Operator

The robot's duty is to reach all goals (10) within 250 time steps. Therefore, the robot which reaches all goals and do not touch any obstacle will obtain the highest fitness value of 2000. However, if this robot touches an obstacle or stops at some time step in the evolution cycle (deadlock), its fitness value should be 0. The fitness values that other robot can have lie between two fitness values of 0 and 2000. Their chromosomes may be used as intermediate mediums of the heredity overlapping operation.



(a) (b)

Fig. 5. Comparison between the experimental results in the different obstacle environments: (a) Maximum fitness and average fitness of the individuals at each generation in 4-obstacle environment; (b) Maximum fitness and average fitness of the individuals at each generation in 10-obstacle environment.

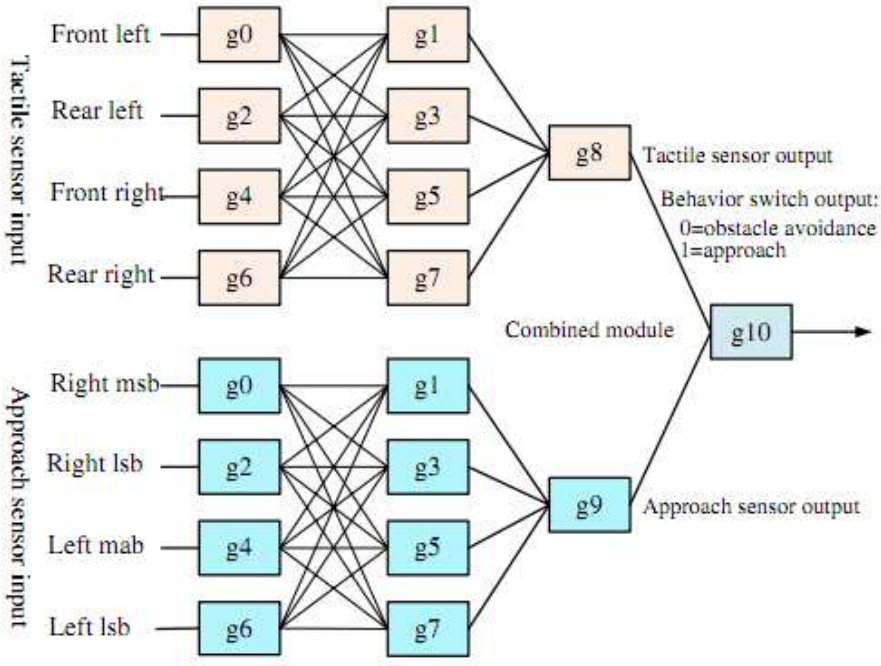


Fig. 6. The neural network topology of behavior switching control: the output of the module network (g10) has two states: 0 represents obstacle avoidance behavior, and 1 means approach behavior.

Because the simulated time steps are reduced, the computation time for each generation is suitably increased, and therefore the number of individuals is increased up to 50. The number of the evolutionary

generations is still 40, and the convergence rate ρ is kept approximately the same as the convergence rate ρ in our previous experiments. Moreover, in this experiment we use the Elitism strategy to ensure that the current best individual survives to a next generation.

Table 2. Functions and parameters of RL and genetic algorithm

Functions of RL and genetic algorithm	parameters
Learning time of each generation (s)	20-30
Number of individuals	50
Crossover probability *pc*	0.5-0.7
Mutation probability *pm*1 for increasing links	0.1
Mutation probability *pm*2 for increasing nodes	0.05
Weight mutation probability *pm*3	0.2
Mutation probability (per gene) *pm*4	0.1
Excitation response mutation probability *pm*5	0.1
Learning rate η of reinforcement learning	0.3

4.3. Experimental Results and Evaluation

Comparison curves of average performance for behavior switching after the GA operated 10 times in two different simulated environments are illustrated in Figure 7, and the initial population that operated every time all is stochastic. We see from Figure 7 that the robots walk more easily in the environments in which their initial average fitness values are higher, and can maintain this superiority in the extra simulation process. After this robot had operated 10 times in the different experiments, we took the average fitness value, and the obtained result curve is smoother. This expresses a more real learning curve than that in the earlier experimental result from one operation only.

It should be noted that the robot's learning curve in the initial phase in this experiment is steeper than the learning curve in the experiment in the previous section. The GA can improve rapidly the performance of the robot, and establish the correct mapping relations from the sensor inputs to the behavior outputs (actions). One of reasons that this phenomenon occurs is to reduce the number of the system's behavior outputs. There is only 1 bit which has two kinds of possible outputs in this experiment, but in the previous experiments, 4 bits including 16 kinds of possible outputs were mapped directly to the motor control (actions to be executed). The second reason is that the architecture and the connection weights of the module neural networks have be evolved using crossover and mutation operations. The complexity of the network architecture is simplified with less nodes and the weights preferred.

Moreover, the experimental results have demonstrated that the input of the approach sensors may be reduced to 2 bits, one of them marking goal source on the left-hand side, and the other marking goal source on the right- hand side. If the goal sources exist on both sides of the robot, the goal sources near the robot is processed at first. According to our experiments, when a touch sensor, in particular a front sensor is activated, the robot's behavior should switch immediately to the obstacle avoidance behavior.

4.4. Comparison with Adaptive Behavior Method

We evaluated our proposed behavior-switching control strategy of an evolutionary robotics by using the evolved ANN controller from the population and the hand coded knowledge-based controllers and compared the results with the adaptive behavior method proposed by Hyeun-Jeong Min in [22].

In our experiments, the 3D robot simulator and a real Khepera II mobile robot are used. The simulator environment includes: three mobile robots (Robot 1, 2, and 3) which generate behaviors using the proposed method, three goal objects (Goal 1, 2 and 3), and two static circular obstacles. Two of the robots act as the movable obstacles.

These two robot-obstacles can only detect and avoid the walls of the fence, and cannot avoid a robot when they collide with other robots. This simulator may change the angles of the view, and evolutionary RL was used for parameter optimization and algorithm verification, and for testing performance of the proposed method.

At the local minimum point P in Figure 8, the robot halted and switched to the collision-avoidance behavior immediately and performed collision avoidance in the different obstacle environments based on behavior switch control and follow wall behavior by crossover and mutation, and reduced the oscillation of the planned trajectory between the multiple obstacles. Figure 9 presents the simulated results of the obstacle avoidance in the different obstacle environments: Robot3 in Figure 9(a) halts and switches to the collision-avoidance behavior immediately when it finds two static obstacles and two movable obstacles in the same

direction at its front and avoids two movable obstacles, fence, and two static obstacles respectively, and finally reaches the Goal3 with smaller oscillation of the planned trajectory; Figure 9(b) illustrates the path trajectory of the robot which generates behaviors and two moving robots (Robot1 and Robot2) which act as the movable obstacles in the environments (from [22]).

As compared with the conventional behavior network and the adaptive behavior method, the genetic encoding complexity of our algorithm is simplified, and the network performance and the convergence rate have been greatly improved due to the application of the module's neural network. Therefore, the robots can successfully perform obstacle avoidance, goal approach, and behavior-switching control in the dynamic environments with multiple obstacles. Comparison of the collision-avoidance performances and the reward value changes of the robot in the environments with walls and obstacles in each learning epoch based on Evolutionary Learning with Neural Networks (ERLNN) and normal RL are illustrated in Figure 10(a) and Figure 10(b). From the results, it is obvious that the trajectory in the early stage is not smooth and gets smooth by evolutionary learning. At last, the robots could move freely without any collision in the environments. This work demonstrates once again the feasibility of application of the controllers based on ANN and GA to ERL and shows its potentials regarding adaptability and learning behaviors.

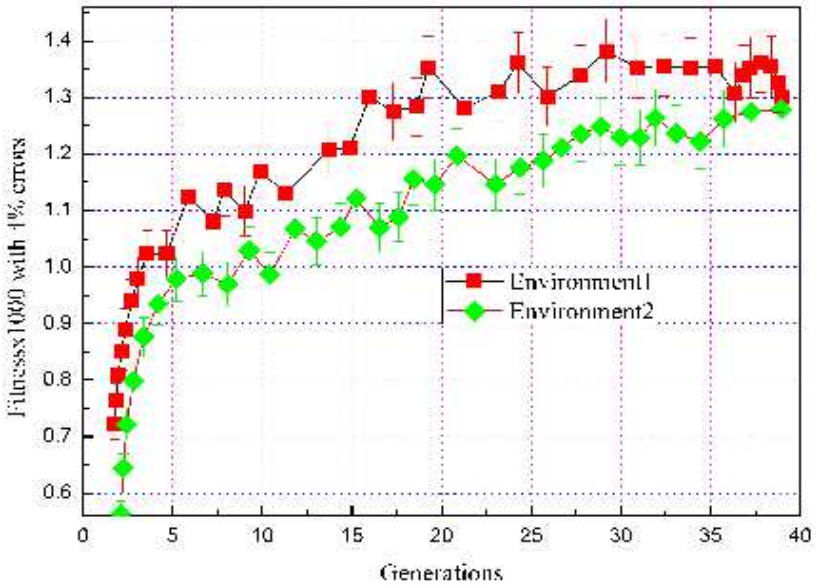


Fig. 7. Comparison curves of average performance for behavior switching control in the different obstacle environments.

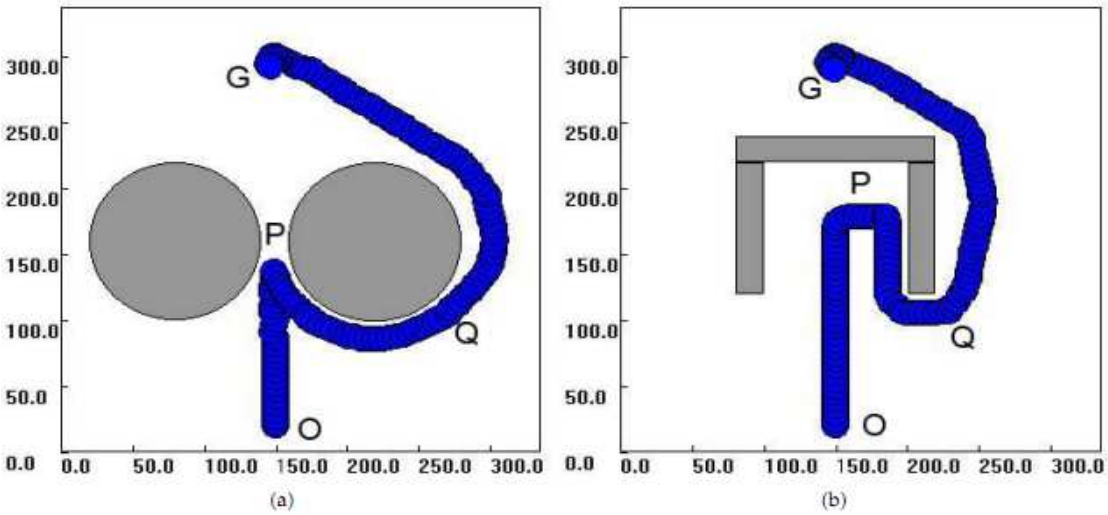


Fig. 8. At the local minimum point P, the robot halted and switched to the collision-avoidance behavior immediately and performed obstacle avoidance in the different obstacle environments based on behavior switch control and follow wall behavior by crossover and mutation.

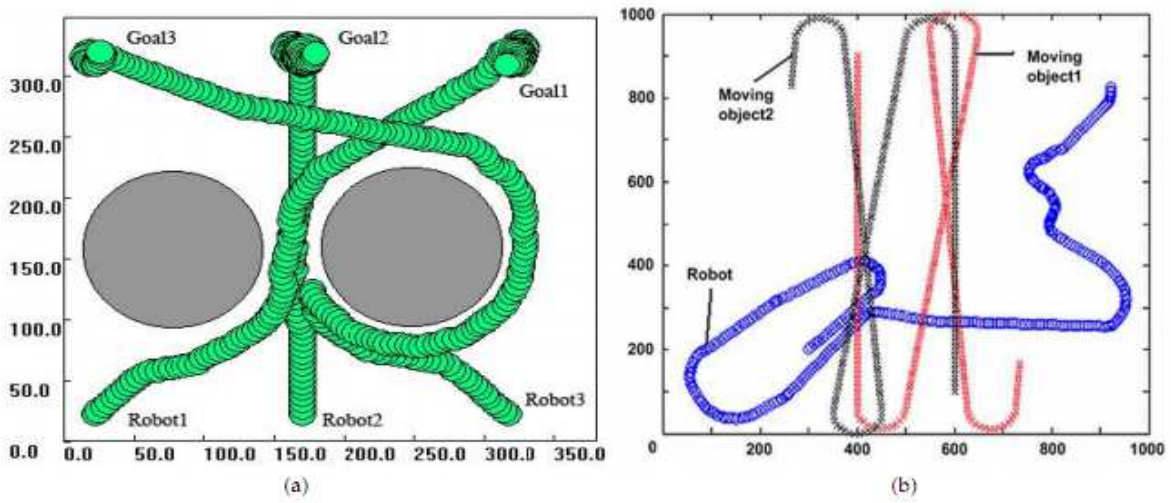


Fig. 9. Comparison of the collision-avoidance behaviors in the different obstacle environments: (a) An example of simulated movement of the robot with neuro-controller for behavior switching to collision-avoidance behavior in complex dynamic environments with two static obstacles and two moving robots which act as the movable obstacles; (b) the trajectory of the robot and two movable obstacles (moving object1 and moving object2) in the environments (from [22]).

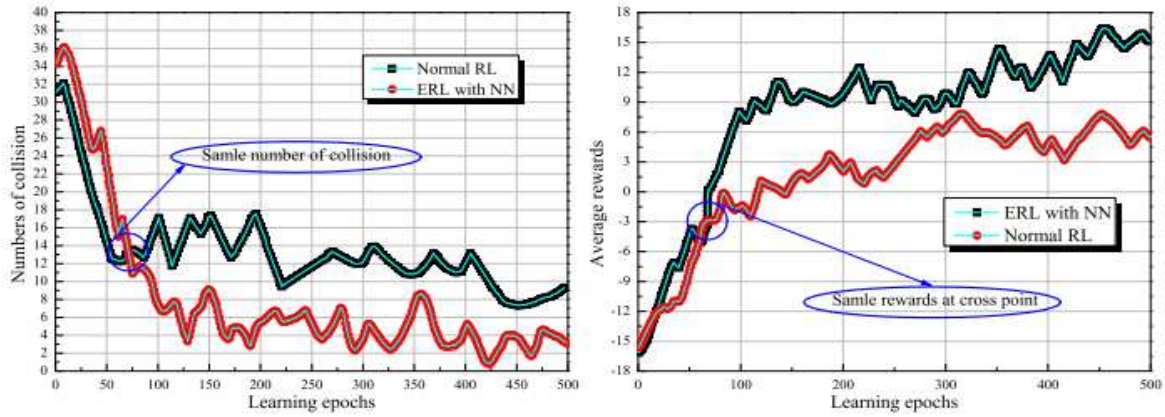


Fig. 10. Comparison of the collision-avoidance performances and the reward value changes of the robot in the environments with walls and obstacles in each learning epoch (one epoch = 100 time steps): (a) The collision times of the robot in the environment with walls and obstacles using ERL with NN decrease faster than using normal RL in each learning epoch. At last, the robots could move freely without any collision in the environments; (b) Changes of the reward values using normal RL and ERL with NN.

5. APPLICATIONS OF PROPOSED ALGORITH

The algorithm proposed in this paper has been partly applied to our simulated humanoid soccer robotics (see Figure 5.4.2). The experiments have been performed both on the Webots Pro 6.4.0 simulator and on a real NAO humanoid robot (Figure 11) using the parameters described in Table 2.

5.1. Fitness function definition

At first, we define some functions, for example, B is position of ball, *O*i represents an opponent, i = (1, 2, · · · , 5), Dist(X, Y) is referred to the dstance between two objects, Dec is to change the binary code of the receiver into algorithm, Avg() is average function. Therefore, the fitness function is defined as:

$$\text{Fit} = \frac{\text{Avg}(\sum_{i=1}^{4} \text{Dist}(B, O_i) + \text{Dist}(O_i, \text{Dec}(c_2c_1)))}{\text{Dist}(B, \text{Dec}(c_2c_1))} \tag{11}$$

where k = 1, 2, · · · , 50 which is the number of individuals. The c2c1 is the binary code of the receiver.

As the operating environments of the soccer humanoid robot system with five against five are dynamic and very complex and its decision-making strategy is most important. Therefore evolutionary reinforcement learning (ERL) for performing cooperation and coordination between the soccer multiple-robots is used for learning the decision-making strategy, evolving network architectures and connection weights (including biases) simultaneously, and adjusting parameters of FNN and GA. Furthermore, the residual algorithm is used to guarantee the convergence of the proposed algorithm to the optimal solution and can retain a high learning rate of the direct algorithms.

5.2. Reinforcement learning for Behavior switching control

The reinforcement learning tasks for robot navigation focus on learning a policy $\pi : S \rightarrow A$ for selecting actions based on current observed states $\pi(st) = at$. Learning the optimal policy $\pi*$ for producing the greatest cumulative reward over time and ϵ-greedy action selection is used. When converged to the true state-action values, then the greedy policy for selecting actions is optimal according to the following criterion:

$$a^*(x) = \underset{b \in A}{\operatorname{argmax}}\, Q(x, a') \quad (12)$$

the reinforcement learning for Behavior Switching Control (BSC) are denoted in Algorithm 1. s is state at time n, s′ is the actual state at time n + 1, a is the action been taken at time n, a′ is the action been taken at time n + 1. $0 < \gamma < 1$ is the discount factor and $\eta(s, a)$ is a learning rate parameter of the state-action pair (s, a) at time step n ($0 < \eta < 1$). A is the set of possible actions and r is the reward the agent receives when action a is taken in state s. The g(s, a, s′) is the cost as s, a, s′ are defined previously. The state-action functions for other states and actions remain unchanged.

ALGORITHM 1: RL-learning for BSC

```
/*—-Phase 1-initialization—-*/
Initialize ANN and the humanoid robot system
qX,qY,qA (quantization of (x, y,a) configurations)
N ← 40 {number of episodes for Q-learning}
Q(s, a) ←0 (∀s, a)
episode←0{actual episode}
r←{immediate reward values for all positions}
/*—-Phase 2-finding a policy—-*/
repeat
    episode←episode + 1; initialize s′
    Get current state
    Obtain Q(s, a) for each action by substituting
    current state and action into the neural network
    Robot moves and gets current state
    Choose a from s using ϵ–greedy exploration
    derived from Q(s, a).
    a∗(s) = argmaxa′∈A Q(s, a′)
    if collision occurred then
        reinforcement = −1 and back to theposition
        before collision.
        Q^target (s, a) = g(s, a, s′) + γmax_a′∈A Q(s′, a′)
        use Q^target to train ANN in Fig. 3.4.2
    end
    /*—-Phase 3-update Q(s, a)—-*/
        repeat
        Take action a, observe s′, r
        Update the state-action function Q(s, a):
        Qn(s, a)←Qn−1(s, a)+
        ηn(s, a)[r+γmax_a′∈A Qn−1(s′, a′)−Qn−1(s, a)]
        s←s′
    until s is terminal
    Repeat 10-20
until episode = N
```

Fig. 11. The experiments have been performed both on the Webots 6.4.0 simulator and on a real NAO humanoid robot with its 21 degrees of freedom using the parameters described in Table 2.

5.3. Fuzzy decision-making applications

The complex decision-making task is divided into multiple learning subtasks that include dynamic role assignment, action selection including obstacle avoidance, goal approach, and behavior-switching control, and action execution which constitute a hierarchical learning system to learn each subtask at the various layers.

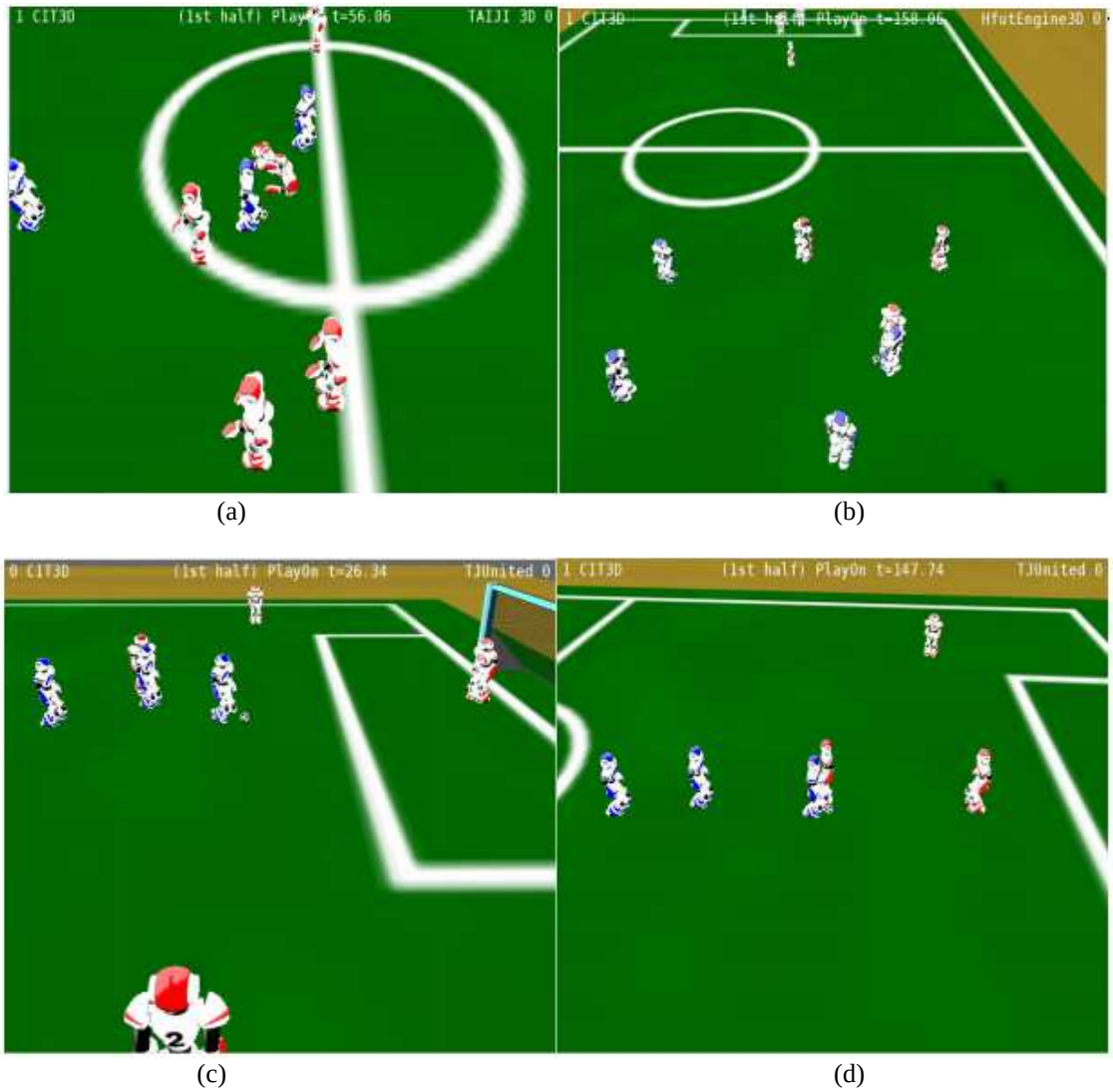


Fig. 12. Some results of the proposed algorithm have been successfully applied to our simulated humanoid robot soccer team CIT3D2010. Some scene frames for the CIT3D (blue) to participate in official RoboCup ChinaOpen2010 Competition were shown: (a) the player of the simulated humanoid robotics soccer team CIT3D launched an attack on the opponents (red) goal and won by one point at 56.06s; (b) the player of the CIT3D with ball was intercepted by an opponent player and tried to pass the ball to teammate at 158.06s; (c) the player of the CIT3D launched an attack and shot at the opponent goal at 26.34s; (d) the player of our humanoid robotics soccer team CIT3D broke through and launched an attack on the opponent goal at 147.74s.

5.4. The solved key problems

In the applications of proposed algorithm, we have focused on solving the following problems:

1) Consider the humanoid soccer robots require fine tuning, particularly, gait optimization for improving the speed of individual robots, the trajectory precision, and the gait stability;
2) The ERL techniques with ANN have been used to find optimal parameter sets for various humanoid robot behaviors;

3) A behavior-switching of the humanoid robots is sped up, for instance, from “FORWARD” action switching to ”LEFT” or ”RIGHT” action. The fast speed switching plays a critical role in the humanoid soccer robotics games;
4) The humanoid soccer robotics agents could escape successfully from the trap of a local minima and avoid motion deadlock status (st+1=st), converge rapidly to optimal solution with various learning rates, and reduce the oscillation of the planned trajectory between the multiple obstacles using various schemes including “crossover and mutation”, “simultaneous learning”, and “novel modified error function” that will be discussed in detail in another article.

Fig. 13. A player (red) of the real CIT3D2010 launched an attack and shot at the opponent goal in the real environments.

(a) (b)

Fig. 14. (a) our simulation humanoid robotics soccer team CIT3D which won the *the 2nd* place of the official RoboCup World Championship (2011) on 5-11 July, 2011 in Istanbul, Turkey; (b) The cup of *the 2nd* place for RoboCupSoceer 3D Simulation League.

These results have been successfully applied to our simulated humanoid robot soccer team CIT3D which won the 1st prize of the official RoboCup Championship and China Open 2010 (July 2010) and the 2nd place of the official RoboCup World Championship (2011) on 5-11 July, 2011 in Istanbul as shown in Figure 12, Figure 13, and Figure 14.

6. CONCLUSIONS AND FUTURE WORK

This work has successfully constructed behavior-based control obstacle avoidance, goal approach, and behavior-switching learning modules of an autonomous robot. The novel modules are not required to have complete environmental knowledge and have more simple architectures which can be easily extended to more complex structures. In our experiments, a neural network modelling the robot behaviors is conceived using evolutionary learning when encountering obstacle. The obstacle avoidance behavior, goal-approach behavior, and behavior-switching modules are performed using the evolved ANN controller from the population and the hand coded knowledge-based controllers which were not dependent on rules assigned previously, but depended on the knowledge acquired from the environments via autonomous evolutionary learning.

The main contributions of the proposed algorithm: 1) RL-EANN algorithm evolves network architectures and connection weights (including biases) simultaneously and emphasizes the behavioral links between parents and their offspring in evolution, such as weights training after each architectural mutation and node splitting; 2) can perform the decision-making strategy and parameters adjustment of FNN and GA by learning; 3) can escape successfully from the trap of a local minima and avoid motion deadlock status ($s_{t+1}=s_t$) of humanoid soccer robotics agents, and reduce the oscillation of the planned trajectory between the multiple obstacles by crossover and mutation; 4) can perform behavior switching and behavior-based control obstacle avoidance effectively using evolved neuro-controllers; 5) can perform closer cooperation and coordination between the teammate agents by evolutionary learning.

Our future works will focus on developing neurocontrollers with architectures like the one presented here for real mobile robots and humanoid robotics in the real environments and making further research on the effect of changes of crossover rate and mutation rate on the best performance and average performance of the multi-robot systems.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers and the editors for their helpful suggestions and comments for improving quality of this paper.

REFERENCES

[1] Baluja S. 1996. ”Evolution of an artificial neural network based autonomous vehicle controller,” IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics, 26(3):450–463.
[2] Barto A G, Sutton R S and Brouwer P S. 1981. ”Associative search network: a reinforcement learning associative memory,” Biol. Cybern, vol.40, pp.201–202.
[3] Beom H R and Cho H S. 1995. ”A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning,” IEEE Transaction on System, Man, and Cybernetics, 25(3):412–425.
[4] Brooks R A. 1991. ”Intelligence without representation,” Artificial Intelligence, 47(1-3):139–159.
[5] Dorigo M and Schnepf U. 1993. ”Genetics-based machine learning and behaviour based robotics: a new synthesis,” IEEE Transactions on Systems, Man, and Cybernetics, 23(1):141–154.
[6] Dorigo M. and Trianni V. 2004. ”Evolving self-organizing behaviors for a swarm-bot,” Autonomous Robots, 17(2-3):223–245.
[7] Fern´ andez Le´ on J A, Tosini F M, Acosta G G, and Acosta H N. 2005. ”An experimental study on evolutionary reactive behaviors for mobile robots navigations,” Journal of Computer Science & Technology, 5(4):183–188.
[8] Floreano D and Mondada F. 1998. ”Evolutionary neuro-controllers for autonomous mobile robots,” Neural Networks, 11(7-8):1461–1478.
[9] Floreano D and Urzelai J. 2000. ”Evolutionary robots with on-line self-organization and behavioral fitness,” Neural Networks, Elsevier, 13(4-5):431–443.
[10] Floreano D and Urzelai J. 1999. ”Evolution of Adaptive-Synapse Controllers.” In D. Floreano et al. (Eds.), Advances in Artificial Life. Proceedings of the 5th European Conference on Artificial Life, Berlin: Springer Verlag. (ECAL’1999).
[11] Garc´ ıa-Pedrajas N, Ortiz-Boyer D, and Herv´ as-Mart´ ınez C. 2006. ”An alternative approach for neural network evolution with genetic algorithm: crossover by combinatorial optimization,” Neural Networks, Elsevier, 19(4):514–528.
[12] Goldberg, D. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

[13] H¨ ulse M, Wischmann S, and Pasemann F. 2004. ”Structure and function of evolved neuro-controllers for autonomous robots,” Connection Science, 16(4):249–266.
[14] Kamio S. and Iba H. 2005. ”Adaptation technique for integrating genetic programming and reinforcement learning for real robot,” IEEE Transactions on Evolutionary Computation, 9(3):318–333.
[15] Kim Jong-Hwan, Kim Ye-Hoon et al. 2009. ”Evolutionary multi-objective optimization in robot soccer system for education,” IEEE Computational Intelligence Magazine, 4(1):31-41.
[16] Krcah P. 2008. ”Towards efficient evolutionary design of autonomous robots,” Springer-Verlag Berlin Heidelberg, G.S. Hornby et al.(Eds.): ICES 2008, LNCS 5216, pp.153–164.
[17] Liu H W and Iba H. 2003. ”Multiagent learning of heterogeneous robots by evolutionary subsumption,” Lecture Notes in Computer Science, LNCS 2724, pp.1715–172, Springer, Berlin, Heideberg.
[18] Lucas S M and Kendall G. 2006. ”Evolutionary Computation and Games,” IEEE Computational Intelligence Magazine, 1(1):10–18.
[19] Mahadevan S and Connell J. 1992. ”Automatic programming of behavior-based robots using reinforcement learning,” Artificial Intelligence, 55(2-3):311–365.
[20] Moriarty D E et al. 1999. Evolutionary algorithms for reinforcement learning. Journal of Artificial Intelligence Research, vol.11, pp.241–276.
[21] Meeden L A. 1996. ”An incremental approach to developing intelligent neural network controller,” IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics, 26(3):474–485.
[22] Min Hyeun-Jeong and Cho Sung-Bae. 2009. ”Adaptive behaviors of reactive mobile robot with Bayesian inference in nonstationary environments,” Applied Intelligence, Springer, 33(3):264–277.
[23] Murray A and Louis S J. 1995. ”Design strategies for evolutionary robotics,” In E. A. Yfantis, editor, Proceedings of the Third Golden West International Conference on Intelligent Systems, Kluwer Academic Press, Las Vegas, Nevada, USA, pp.609–616.
[24] Nelson A L, Barlow G J, and Doitsidis L. 2009. ”Fitness functions in evolutionary robotics: A survey and analysis.” Robotics and Autonomous Systems, 57(4):345-370.
[25] Nolfi S. 1997. ”Using emergent modularity to develop control systems for mobile robots,” Adaptive Behavior, 5(3-4):343-364.
[26] Nolfi S and Floreano D. 2000. ”Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines,” MIT Press, Cambridge, MA.
[27] Schmidt M D and Lipson H. 2008. ”Co-evolving fitness predictors,” IEEE Transactions on Evolutionary Computation, 12(6):736–749.
[28] Sutton R. S. and Barto A. G. Reinforcement Learning: An Introduction, Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 1998.
[29] Tabuse M, Kitazoe T, Shinchi T, and Todaka A. 2002. ”Evolutionary robot controllers with competitive and cooperative neural networks,” Artificial Life and Robotics, 6(1-2):52–58.
[30] Tabuse M, Shincht T, Todaka A, and Kitazoe T. 2003. ”Evolutionary robot with competitive-cooperative neural network,” Transactions of Information Processing Society of Japan, 44(10):2503–2513.
[31] Trujillo L, Lutton E, Lutton E, and de Vega F F. 2008. ”Behavior-based speciation for evolutionary robotics,” in Proc. of GECCO08, July 12-16, Atlanta, Georgia, USA.
[32] Vanneschi L, Tomassini M, Collard P, and Clergue M. 2003. ”Fitness distance correlation in structural mutation genetic programming,” Lecture Notes in Genetic Programming, LNCS 2610, pp.455–465, Springer-Verlag, Berlin Heideberg.
[33] Wang H Y. 2000. ”Research on integration of the evolutionary robot behaviors based on neural network,” Hefei: Hefei University of Technology, pp.36–57.
[34] Wang H Y, Yang J A and Jiang P. 2000. ”Research on evolutionary robot behavior by using developmental network,” Journal of Computer & Development, 37(12):1457–1465.
[35] Yang J A and Zhuang Y B. 2010. ”An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem,” Applied Soft Computing, 10(2):653–660.
[36] Yang J A and Wang H Y. 2002. ”Research on integration of the evolutionary robot behaviors based on neural network,” Journal of Shanghai Jiao Tong University, Vol.36, sup. pp.89–95.
[37] Yao X. 1999. ”Evolving Artificial Neural Networks”, in Proceedings of the IEEE, 87(9):1423-1447.
[38] Zagal J C and Ruiz-del-Solar J. 2007. ”Combining simulation and reality in evolutionary robotics,” Journal of Intelligent and Robotic Systems, 50(1):19–39.