Evolving Robot Behavior via Interactive Evolutionary Computation: From Real-World to Simulation

Gerry Dozier
Department of Computer Science & Software Engineering
Auburn University
Auburn, AL 36849-5347 USA
gvdozier@eng.auburn.edu

Keywords

Interactive Evolutionary Computation, Khepera, GRNN

ABSTRACT

In evolutionary robotics (ER), candidate behaviors are typically represented as fully connected feed-forward neural networks. The evaluation functions used for determining the "goodness" of a candidate behavior are usually written as closed-form equations which are approximations of what the behavior engineer considers good and/or bad behavior. However, the more complex the desired behavior, the less accurate a closed-form evaluation function may be.

For problems where the evaluation function is difficult to express, interactive evolutionary computations (IECs) are welcomed alternatives. However, the use of IECs has not been widely used because of: (1) the large amount of time the behavior engineer would need to spend interacting with the robot and (2) the human fatigue factor associated with long interactive sessions. In this paper, we present an interactive evolutionary system for behavior design. This system allows a behavior engineer to interactively teach a robot obstacle avoidance behavior in minutes rather than in hours, thus reducing the amount of fatigue experienced by the behavior engineer.

1. INTRODUCTION

Recently, there has been an increased interest in the application of ECs to the area of robotics and in particular the sub-areas of robot shaping and behavior engineering [1]. In many of these applications, candidate behaviors are represented as feed-forward neural networks. A population of candidate behaviors is evolved by an evolutionary computation (EC) in hopes of discovering a behavior that optimizes a pre-specified evaluation function. Although this method of behavior design is effective, the development of a precise evaluation function for complex behaviors may be difficult if not impossible [10, 11]. This realization has led researchers towards the consideration of interactive evolutionary computations (IECs) [2, 4, 5].

Presently, there exist two difficulties with using IECs for behavior engineering: (1) the slow convergence rates of IECs due to the large and complex neural network representation of candidate behaviors, and (2) the human fatigue that comes as a result of the behavior engineer having to spend long training periods with the robot.

In this paper, we present an IEC which is a preliminary attempt towards alleviating the problems previously mentioned. Our IEC achieves this by (1) reducing the size of the neural networks that represent candidate behaviors so that the IEC can evolve faster and by (2) using a model of the behavior engineer's preference in an effort to further reduce the amount of interaction time required to teach a behavior. We demonstrate the effectiveness of this approach by using it to interactively develop obstacle avoidance behavior.

SAC 2001, Las Vegas, NV.

2. BACKGROUND

In this section, we provide a brief overview of interactive evolutionary robotics, the Khepera miniature robot, and general regression neural networks (GRNNs). We use GRNNs as an alternative to large feed-forward neural networks.

2.1 Interactive Evolutionary Robotics

In interactive Evolutionary Robotics (IER), IECs are used to develop and shape robot behaviors. The first published application of IER was by Lewis, Fagg, and Solidum [2, 4]. In [4], Lewis, Fagg, and Solidum use an IEC to evolve gait behavior for a six legged robot named Rodney. In interactively evolving gait behavior, the authors applied a concept that they developed called *staged evolution*.

In staged evolution, a number of intermediate fitness scores are first defined by the behavior engineer. Each intermediate fitness score corresponds to a distinct evolutionary stage (or milestone in the evolutionary process). Each time a candidate behavior is executed, the behavior engineer assigns it a fitness based on its stage in the evolutionary process. The results of using staged evolution were impressive; however, in using this approach the behavior engineer is still left with the task of defining the criteria for each evolutionary stage.

The authors of [2] also use the concept of staged evolution along with syntactical constraints [1] to evolve gaits for

[1] [2] and [4] both make use of decomposing the gait behavior

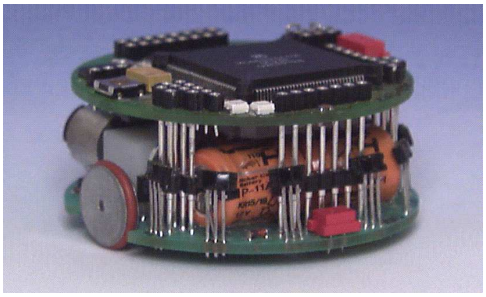

Figure 1: Khepera: Manufactured by K-Team (http://www.k-team.com)

an eight legged robot. The syntactic constraints were used to limit the type of genetic programs that could be grown. In contrast to [4], Gruau et. al. [2] assign fitness subjectively meaning that a candidate behavior's relative goodness may fluctuate each time it is re-evaluated. Using subjective fitness assignment provides the behavior engineer with the ability to shape the population (by regulating selection pressure) as well as the behaviors making up the population.

In [5], Lund et. al. develop an interesting method that allows behavior engineers (in their case the engineers were small children) to design robot behaviors for a LEGO jeep-robot. Each candidate behavior was represented as a neural network with 6 input neurons (4 bumper sensors, 2 light sensors), 6 output neurons, 6 biases, and a total of 42 connections (or weights). Their IEC was used as follows. Children are allowed to view 9 simulated robots moving through an environment. The child is then allowed to select three behaviors that are particularly attractive (or the child can select the same behavior 3 times). These three behaviors are allowed to make two mutants of themselves by mutating any of the 42 neural network weights with a mutation rate of 10%. A weight is mutated by randomly adding to it a value within the interval [-10..10]. Once the child has interactively evolved a satisfactory behavior, the preferred behavior is downloaded into the robot.

2.2 Khepera

For the experiments reported in this paper, we used the Khepera miniature robot. Khepera has a diameter of 55mm, a height of 30mm, and sports a Motorola 68331 processor, eight infra-red proximity and light sensors, and two DC motors. The speed of Khepera is measured in motor pulses, mp, where $1mp = 1/10ms$. The maximal speed of Khepera is $127mp$ which is approximately 1m/s. Figure 1 shows a picture of Khepera.

2.3 GRNNs

The GRNN was developed by Specht [9] and is very similar in principle to the zero-order Sugeno fuzzy inference system [3] and Shepard's method [6]. The GRNN can be defined as follows. Given a set of n training elements, $\{(t_1, d_1), (t_2, d_2), \ldots, (t_n, d_n)\}$ where d_i is the desired output of the i^{th} training instance, t_i. Let the classification of some input instance, x, be $f(x) = \frac{\Sigma_{i=1}^{n} h_i(x,t_i)d_i}{\Sigma_{i=1}^{n} h_i(x,t_i)}$ where $h_i(x,t_i) = exp(-\frac{||x-t_i||^2}{2\sigma^2})$. The function h_i is referred to as the i^{th} hidden function. The width of the hidden functions is denoted as σ.

into two sub-behaviors: leg movement and locomotion.

3. AN INTERACTIVE EVOLUTIONARY SYSTEM FOR BEHAVIOR DESIGN

Our interactive evolutionary system for behavior design (IESBD) attempts to reduce user fatigue in two ways: (1) by reducing the amount of interactive evolution time through evolving simpler forms of neural networks[2] and (2) by adding a non-interactive reflection phase that uses a model of the behavior engineer's preference.

The IESBD uses two phases to allow the behavior engineer (user) to interactively evolve obstacle avoidance behavior. The first phase uses an IEC that allows the user to interactively evolve a population of candidate behaviors represented as GRNNs. The IEC gathers information about the user's behavior preference and builds a training set for the second (reflection) phase.

During the reflection phase, the robot remains stationary and a model-based evolutionary computation (MEC) is used to continue to evolve the population by *simulating* the preference of the user. The MEC uses the information gathered by the IEC to construct a GRNN that can be used to model the user's preference. The GRNN is then used by the binary tournament selection mechanism that takes as input two candidate behaviors and returns the behavior more likely to be preferred by the user.

3.1 The IEC

During the interactive phase, the IEC evolves candidate behaviors as follows. Initially a random population of candidate behaviors is generated but not evaluated. After the initial population has been randomly generated, two candidate behaviors are randomly selected to be parents. Each parent is allowed to control Khepera and interact with the user for a specified amount of time ($\approx$ 4 sec.). After both parents have had an opportunity to interact with the user, the parent behavior with the lower fitness is selected to be the winner[3]. Once a winner has been determined, if the difference between the winner and the loser is greater than zero, a difference rule is constructed and added to a list of difference rules (difference rules will be discussed in later). Next, the parents are used to create an offspring that replaces the losing parent. This process of selecting parents, evaluating parents and creating offspring is repeated for a user-specified number of iterations.

In the sections to follow, we present some of the more salient parts of the IESBD in greater detail. These are as follows: (1) the representation of candidate behaviors, (2) the representation of difference rules, and (3) the evolutionary operators used by the IEC to create offspring.

3.1.1 *The Representation of Candidate Behaviors*

Figure 2 shows how a GRNN is used by Khepera. Notice that the left sensors (sensors 0, 1, and 2) are connected to the right wheel through a GRNN. Similarly the right sensors

[2]By evolving GRNNs the IESBD evolves obstacle avoidance behavior in minutes rather than in tens of hours as reported by [7, 8].

[3]The first parent is chosen to be the winner if and only if its fitness assignment is lower than the fitness assignment of the second parent.

(sensors 3, 4, and 5) are connected to the left wheel through a GRNN. Using this architecture, when the right sensors detect an obstacle, Khepera can move to left by slowing down the (or reversing the direction) of the left wheel. The same is true for the left sensors and the right wheel. This two-hemisphere architecture also allows one to use a single GRNN. This will reduced the number of weights to be evolved by 50%.

The GRNN used by our IESBD has only 3 hidden functions. The centers for the hidden functions are t_0=(1023, 0, 0), t_1=(0, 1023, 0), and t_2=(0, 0, 1023). Notice that each t_i is a vector containing 3 components. The first hidden function will fire with a value of 1.0 when the front-most sensors (sensors 2 or 3) have a sensor reading of 1023. The value, 1023, was chosen because it is the highest sensor reading available. This reading means that the sensor in question (thus the robot) is dangerously close to an obstacle. The second hidden function has a center at t_1 and will fire with a value of 1.0 when the middle-front sensors (sensors 1 or 4) are dangerously close to an obstacle. Finally, the third hidden function with, its center at t_2, will fire when the middle-back sensors of a hemisphere (sensors 0 or 5) are dangerously close to an obstacle.

Using the above architecture, a candidate behavior can be represented as a vector of 4 integer values. The first three values correspond to the three weights of the hemisphere GRNN while the fourth value corresponds to, σ, the widths of the hidden functions. The first three values represent the action (change in motor speed which is measured in *mp*) to be taken when the corresponding hidden function fires with the maximum value of 1.0. An optimal action would be to respond with a negative speed value with the largest magnitude. This will cause Khepera to turn quickly away from a detected obstacle.

Each candidate behavior also has a fitness associated with it. This is the number of times the sensors detect light from a flashlight while the candidate behavior is controlling Khepera. The user interacts with a candidate behavior by shining light from a flashlight onto the back sensors (sensors 6 and 7) whenever the candidate behavior causes Khepera to behave in a undesirable manner (i.e. running or bumping into obstacles/walls, lunging forward/backwards repetitively, etc.). We refer to this form of interaction as a "flashlight tap". Some candidate behaviors are more dangerous than others, therefore an abort condition was established that immediately halts the robot and assigns the destructive behavior a fitness equal to the maximal penalty available[4].

3.1.2 The Representation of User Preference

The IEC builds a list of rules that can be used by the MEC during the reflection phase to model user preference. Each rule contains 4 integer values taken from the set, {-1,0,1}, and one integer value taken from the set, {-1,1}. The first four values correspond to whether the difference between a component of two parent behaviors is negative (-1), equal (0), or positive (1). The fifth component represents the classification based on the fitnesses of the candidate behaviors being compared. If p_1 was the better parent then the fifth component of the difference vector will be -1. Otherwise the fifth component will be 1. For example let

[4]The maximum penalty for our experiments the total number of sense-act cycles that candidate behaviors were allowed to execute while in Khepera.

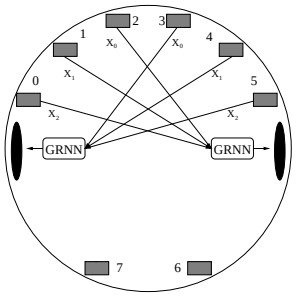

Figure 2: Khepera using a GRNN Controller

$p_1 = (-14, -25, -17, 136)$ and $p_2 = (-20, -25, -22, 388)$ be two parent behaviors, let the fitnesses of p_1 and p_2 be $f_1 = 50$ and $f_2 = 21$, and let the difference threshold be zero. Thus, $p2$ would be selected to be the winner and the resulting difference rule, v, would be $v = (1, 0, 1, -1, 1)$. In difference rule v, the first value is the value 1 because the difference between the first components of p_1 and p_2 is positive. This is also the case for the difference between the third components of the parents. Notice that the second component of the difference rule is zero. This only occurs when corresponding components of parents are equal.

3.1.3 The Evolutionary Operators

The IESBD uses three evolutionary operators: (1) sign mutation, (2) uniform crossover, and (2) uniform-bounded mutation. Each of these operators were specifically designed for the different cases that may arise during the evolutionary process.

Before a candidate behavior is allowed to control Khepera, its fitness is checked to see if has been aborted in the past. If this is the case then it is useless to allow the behavior to control Khepera again. Instead the sign of one of the first three components of the aborted candidate behavior is randomly flipped. This is an important operator because candidate behaviors present in the population at early stages of the evolutionary process tend to be very dangerous.

Uniform crossover is used exclusively when no feasible candidate behaviors exist. In uniform-crossover, the offspring randomly receives a value from either parent with probability 0.5. The new offspring replaces the losing parent and is selected to be, p_1, the first parent. The second parent is randomly selected from the population. By making sure that each offspring is selected to be p_1 the IEC minimizes the loss of genetic material. In fact, by using this approach crossover allows more than two parents to crossed indirectly. Therefore crossover is used exclusively until a feasible behavior has been evolved then uniform-bounded mutation is used to refine all feasible behaviors.

Uniform-bounded mutation is used only when at least one feasible candidate behavior has been evolved. We define a feasible candidate behavior as one that that has recorded at least two consecutive wins against other candidate behaviors and has not been aborted. In uniform-bounded mutation, a feasible parent,p_f, creates one offspring,p_c, as follows: $\forall_{i<3} \quad p_{c,i} = p_{f,i} + rnd(-\mu_1, \mu_1) \quad and \quad p_{c,3} = p_{f,3} + rnd(-\mu_2, \mu_2)$.

If it is the case that both parent behaviors are infeasible (meaning that they have both been aborted) then one

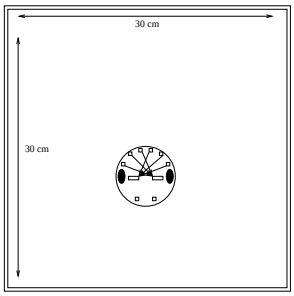

Figure 3: The Environment

feasible candidate is randomly selected and mutated twice in order to create two offspring to replace the two parents. Both offspring are then selected to be parents and are evaluated. If the case arises where the winning parent is feasible and the losing parent is infeasible then the winning parent creates an offspring using uniform-bounded mutation that that replaces the losing parent. The offspring is then selected to be one parent while the second parent is randomly selected from the population.

3.2 The MEC

The MEC used during the reflection phase works as follows. Two individuals are randomly selected from the population to become parents. Once the two parents are selected, they are used to create a difference vector. The difference vector is then given along with the set of difference rules to the preference GRNN. The preference GRNN returns a negative value if the first parent is to be chosen as the winner and a nonnegative value if the second parent is to be chosen. The winning parent uses the uniform-bounded mutation operator discussed earlier to create an offspring to replace the losing parent. After the losing parent is replaced, two more parents are randomly selected from the population and the evolutionary process is repeated for a user-specified number of iterations.

4. EXPERIMENT SETUP

The experiments reported in this paper take place in a 30cm by 30cm arena made with white cushioned poster board (see Figure 3). If no walls are detected, both motors are set at a speed of 15mp. Candidate behaviors take integer values for their first three components from the interval, [-30..30] and integer values for their fourth component from the interval, [1..512]. Therefore the size of the behavior space (search space) for this problem is 116,214,272.

Candidate behaviors were identified as destructive if any of the following cases were true, where RWMS and LWMS denote the right wheel motor speed and the left wheel motor speed:

1. sensor2 and sensor3 were equal to 1023 and both RWMS and LWMS were greater than zero;
2. sensor1 = 1023 and $RWMS \geq LWMS \geq 0$;
3. sensor4 = 1023 and $LWMS \geq RWMS \geq 0$;
4. sensor0 = 1023 and $RWMS > LWMS \geq 0$;
5. sensor5 = 1023 and $LWMS > RWMS \geq 0$.

These rules were used to protect Khepera from destructive behaviors that might cause it to run into a wall at a high speed even when a wall is detected. For example these rules will abort a behavior that says, "when the front sensor detects an obstacle increase the wheel speed".

5. EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of our IESBD, we conducted two experiments. For both of the experiments we used the following parameter settings: the population size was set to 15, μ_1 and μ_2 were set to 12 and 102, the total number of iterations (interactive phase + reflective phase) was held constant at 400, and each candidate behavior was allowed to control Khepera for 4 seconds (100 sense-act cycles) during the interactive phase. We ran our IESBD a total of 5 times for each experiment.

The results are organized in a table where each row represents the results for one individual run of the IESBD. The columns (from left to right) are as follows. The first column (T) corresponds to the number of the run (trial), the second column represents the time (in seconds) the IESBD took in evolving for the specified number of interactive iterations[5], and the third column contains the number of difference rules that were created by the IEC to form a neural model for the MEC. In the fourth column, the best behavior evolved by the IEC alone is presented along with the number of sense-act cycles one or more sensors had a reading of 1023 (which we refer to as a **hit** shown in brackets) when the behavior was allowed to control Khepera for a total of 40s (1000 sense-act cycles). If a behavior was not feasible in that it was aborted then it is denoted by using NF. Finally, the fifth column contains the best behavior evolved by the MEC[6] along with the recorded number of hits (again shown in brackets).

For the experiments we were trying to train the robot to (1) avoid running into walls and (2) to move as slowly as possible. To accomplish this, we used the following training strategy. As long as Khepera was in contact with a wall or exhibiting erratic behavior (i.e. repeatedly lunging forward and backwards), we would shine the flashlight on sensors 6 and 7. For stable behaviors that were able to avoid the walls, we would give a quick flashlight tap each time Khepera had to turn away from a wall. Using this strategy, the slower the speed the fewer flashlight taps the behavior experienced.

5.1 Results of Experiment I

Table 1 shows the results of our interactive evolutionary system on five runs. For each of the five runs (trials) the total number of interactive and model-based iterations were 150 and 250. The results in Table 1 show that the IEC is effective in evolving feasible behaviors. This is the case for all 5 trials. However, the model developed during the interactive phase correctly modeled user preference 80% of the time. Notice that the MEC is able to develop better behaviors on Trials 1-3. On trial 4, a non-feasible behavior was evolved and on trail 5 a worse performing behavior was evolved.

The best overall behavior evolved by the MEC was on the first and second trials. This behavior causes the robot to move as slowly as possible and to turn away from a detected obstacle as quickly as possible given the constraints

[5]The MEC took less than a second to evolve its iterations.

[6]For the MEC, we selected the behavior that had the greatest number of wins. When a behavior beat another the winning behavior incremented its wins by 1 plus the number of wins the losing behavior had.

on straight-ahead speed, turning speed, and the widths of the Gaussian functions. In fact the larger the σ value is, the slower the straight ahead speed of the robot will be. However, it should be noted that this behavior would be optimal on for the 'four wall only' environment. If the objective were for Khepera to move through narrow corridors, the best behavior would be different.

T	S	R	IEC	MEC
1	735	91	(-30,-29,-22,512)[199]	(-30,-30,-30,512)[8]
2	939	108	(-20,-30,-20,449)[298]	(-30,-30,-30,512)[11]
3	790	98	(-30,-15,-25,277)[328]	(-30,-30,-30,401)[64]
4	590	72	(-30,-30,-30,139)[216]	(-30,-20,-25,36)[NF]
5	618	79	(-21,-28,-26,221)[270]	(-30,-30,-7,106)[363]

Table 1: Five Trials with 150 Interactive Iterations and 250 Model-Based Iterations

5.2 Results of Experiment II

Table 2 shows the results of 5 more trials where the number of interactive iterations was reduced by 50%. The number of model-based iterations was increased appropriately so that we could keep the total number of candidate behaviors created constant at 400. As one might expect, the total amount of time it took the IEC to run was reduced when compared with the results in Table 1.

Despite reducing the number of interactive iterations, the IEC still performed well in that it was able to evolve feasible candidate behaviors on all trials except for the first. However, the quality of the behaviors evolved by the IEC is not as good as those evolved by the IEC in the first experiment.

The MEC is able to develop better behaviors than the IEC on all trials except for Trial 4. In this trial, a good rule was not constructed that would cause the model-based selection algorithm to prefer slower straight-ahead speeds. However, the action values evolved (-24, -30, -30) are good. Notice, in Trial 1, that although the best candidate behavior evolved by the IEC was not feasible an appropriate set of rules was discovered during the interactive phase which allow the reflection phase to produce a feasible candidate behavior.

T	S	R	IEC	MEC
1	242	30	(-18,-30,-6,64)[NF]	(-30,-30,-30,82)[206]
2	336	46	(-21,-7,-30,420)[392]	(-30,-30,-30,512)[3]
3	218	29	(-18,-20,-30,495)[285]	(-30,-30,-30,512)[4]
4	146	13	(-30,-24,-4,181)[533]	(-24,-30,-30,1)[NF]
5	142	14	(-30,-30,-3,166)[475]	(-30,-30,-30,512)[8]

Table 2: Five Trials with 75 Interactive Iterations, 325 Model-Based Iterations

6. CONCLUSIONS

Our preliminary results show that evolving GRNNs with our IESBD is a promising approach towards reducing the amount of fatigue experienced by the behavior engineer. The IEC was able to evolve feasible behaviors 9 out of the 10 trials. Furthermore, we show that by modeling user preference the interactive time may be reduced further. The MEC was able to improve on the quality of behavior evolved by the IEC 7 out of the 10 trials. Although the IESBD was not 100% successful, we believe that the approach is a step in the right direction.

In the future, we plan to develop IESBDs that will use model-based selection during the interactive phase. It is hoped that this will accelerate the convergence upon good behaviors.

7. REFERENCES

[1] Dorigo, M., and Colombetti, M. (1998). *Robot Shaping: An Experiment in Behavior Engineering*, The MIT Press.

[2] Gruau, F. and Quatramaran, K. (1996). Cellular Encoding for Interactive Evolutionary Robotics, *tech. rep. University of Sussex, School of Cognitive Sciences, EASY Group, Brighton, UK.*

[3] Jang, J.-S. R., Sun, C.-T. and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall.

[4] Lewis, M. A., Fagg, A. H., and Solidum, A. (1992). Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot, Proceedings of the 1992 IEEE International Conference on Robotics and Automation, pp. 2618-2623.

[5] Lund, H. H., Miglino, O., Pagliarini, L., Billard, A., and Ijspeert, A. (1998). Evolutionary Robotics – A Children's Game., Proceedings of the 1998 IEEE International Conferences on Evolutionary Computation, pp. 154-158.

[6] Mitchell, T., M., (1997). *Machine Learning*, WCB/McGraw-Hill.

[7] Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, pp. 190-197.

[8] Odagiri, R., Yu, W. Asai, T., Yamakawa, O., and Murase, K. (1998). Measuring the Complexity of the Real Environment with Evolutionary Robot: Evolution of a Real Mobile Robot Khepera to have a Minimal Structure, *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 348-353.

[9] Specht, D. F., (1991). A General Regression Neural Network, *IEEE Transactions on Neural Networks,* 2(6):568-576.

[10] Takagi, H. (2000). Active User Intervention in an EC Search, *Proceedings of the 5th Joint Conference on Information Sciences*, Vol. I, pp. 995-998.

[11] Takagi, H. (1998). Interactive Evolutionary Computation, *Proceedings of the 5th International Conference on Soft Computing and Information / Intelligent Systems (IISUKA'98)*, pp. 41-50.