A Comparison of a Hand-Coded Strategy versus an Evolved Strategy for Keepaway Soccer

Ciarán Mullett

School of Computer Science & Informatics
University College Dublin

Abstract

Much has been said in the past about the effectiveness of Evolutionary Algorithms (EA's) and their ability to evolve solutions to various optimization problems[1,4,]. A lot has also been written as to the effectiveness of evolved strategies for the *Keepaway* Soccer problem[2,3]. This paper is an attempt to place this performance in context and quantify the relative strengths and weaknesses of a hand-coded strategy versus that of an evolved strategy for keeping the ball. A good measure or this relative performance would be the amount of time the ball is in the possession of the Keepers or the amount of passes completed by the Keepers during the course of an individual *Keepaway* game or session. What was observed during the course of experiments was that the evolved strategy did similarly in comparison to the hand-coded strategy at keeping the ball. Therefore we can say that no major advantage accrues from using an evolutionary approach over the hand-coded approach.

1. Introduction

Much of the literature of Evolutionary Algorithms and Evolutionary Computation waxes lyrical[1,4] about the effectiveness of the evolutionary technique for finding good solutions to problems but little concrete evidence of this is presented, sometimes it is said that EA's produce human-competitive results or better[1,6]. This paper is an attempt to concretize either way some of these views using the *Keepaway* domain as a test bed.

EA's were designed to prove or disprove the thesis that simulating evolution would provide a tool for solving common optimization problems. EA's based on simple models of natural evolution attempt in some way to capture the essence of what makes natural evolutionary processes successful. There are many different EA approaches available but they all share some commonality. EA's are usually based on populations of solutions which allow for parallel search processes. EA's commonly have innovation operators such as mutation which allow problems to be looked at from different perspectives. EA's also combine useful partial solutions to problems using recombination operators and grade solutions based on a fitness function. A fitness function is a measure of the effectiveness of a solution to a given problem. Selection is another important element in EA's. Selection works by selecting the fittest individuals from a set of solutions based on their individual fitness's. The EA used in my experiment was a real-valued EA whose aim was to modify two real parameter values and provide an effective solution to the Keeper problem.

In *Keepaway* Soccer or just *Keepaway*, one team of Keepers tries to keep possession of a ball for as long as possible given the attempts of an opposing Attacker to get the ball. The game is played on a fixed size playing area , the number of keepers is greater than those of attackers but the Attackers are much faster than the Keepers. In the particular implementation studied in this paper there were three Keepers playing opposite a single Attacker. The *Keepaway* simulator used for the experiments was that developed by Daniel Kuebrich for the MASON[7] environment. Daniel Kuebrich's implementation had extremely limited intelligence with both Keepers and Attackers simply following the ball and then randomly kicking it away anywhere. Invariably the faster Attacker would be first to the ball, kick it away randomly and only on the off chance that the attacker randomly kicked it into the path of a Keeper would any Keeper ever get a touch of the ball. What was needed was an effective hand-coded strategy for the keepers to act in a more structured manner and keep the ball using some level of intelligence, this strategy would then be used as a comparison against a competing evolved

strategy developed using George Mason University's ECJ[11] evolutionary toolkit. Over a number of runs it should then become clear as to which is the more effective strategy and hence some measure of how well evolved strategies operate compared to hand coded strategies can be derived.

2. Strategies For *Keepaway*

The question here is what constitutes a good strategy for *Keepaway*. Most of the best hand-coded strategies currently in operation are based on two basic methods;

```
HoldBall()
Pass()
```

Both these methods were implemented in slightly different ways in the systems of Gustafson[3,8] and Stone et al[9]. With `HoldBall()` the keeper simply holds the ball in the one position for as long as possible given the attackers attempts to get the ball. When `HoldBall()` is combined with a knowledge as to the relative position of teammates and the location of the attacker a keeper can make an informed decision as to how long to hold the ball before releasing it with a call to the pass method. The pass method in many of the well established hand-coded strategies is also based on knowledge acquired from the environment about the relative position of teammates and the location of the Attacker. A good pass method such as that proposed by Stone, Sutton and Kuhlmann[5,9] has a weighting scheme whereby potential passes by Keepers are weighted based on the angles between Keepers and Attackers and the relative distances between Keepers. Gustafson also proposed some interesting additions to theses strategies including, the additions of a method for keepers not currently in possession to get into an open position[3] to make it easier for teammates to pass to them. Another interesting strategy, and one which is used extensively in my hand-coded *Keepaway* strategy is what I call an *aggressive* strategy whereby the Keepers strategy is not simply to focus on waiting to receive the ball but to actively get the ball themselves at all times. This *aggressive* strategy has the result that when the keepers have the ball they form a tight circle around the ball passing among themselves thus not allowing the Attacker to easily win the ball without having to force itself through the tight formation of the Keepers. Strategies such as the ones outlined can hold the ball for a few seconds on average on a 100 x 100 playing field which is about as long as can be achieved at the present time. An evolved strategy draws its individual genes and genome from some of the parameters associated with these hand-coded strategies and then potentially evolves values for these parameters which would allow the keepers to hold the ball for a longer time than a hand-coded strategy with it's statically defined parameters.

3. Tools

The two tools used in the experiments were MASON and ECJ.

3.1 MASON

MASON was designed to be a foundation on which to run large Java simulations. MASON was built using Java and allows for the visualization of experiments in the Evolutionary Computation domain. In the case of *Keepaway* it acts as a foundation on which the *Keepaway* simulation runs. One of the great advantages of MASON as a simulation environment is that it "delineates between model and visualization"[7], this separation of concerns allows you to focus primarily on the model and then think about the visualization issues later. The *Keepaway* simulator already in MASON was that developed by Daniel Kuebrich. Kuebrichs' version of *Keepaway* was very basic indeed with only simple go-to functionality for the keepers, they made no attempt to pass the ball and simply randomly kicked it away when they could. The *Keepaway* interface can be seen in Figure 1 with the individual keepers chasing the ball and a lone attacker doing likewise.

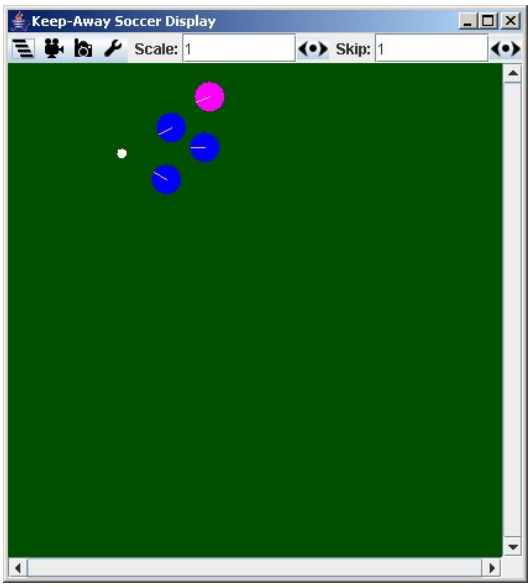

Figure 1 ***A View of the MASON Keepaway interface.***

3.2 ECJ

ECJ is an Evolutionary Computation system written in Java. It is a highly flexible system which allows all the various settings for an evolutionary experiment to be defined in a parameter file. These individual parameter files can then be used in conjunction with the toolkit to do experiments. The parameter file format is very flexible and allows you to easily modify parameters such as the probability of crossover, type of selection mechanism, type of representation and any other important parameters for individual experimental setups.

4. Experimental Setup

The experimental setup for the following experiments was quite simple. Both the hand-coded and the strategy whose parameters had been evolved would be run 50 times with the total number of passes recorded for both the evolved and hand-coded strategies for a single run. Each run would be terminated as soon as an attacker gained possession of the ball. Statistical information could then be drawn from the number of passes achieved in individual runs. These numbers could then be compared and it should become clear which strategy was more effective.

The hand-coded policy as mentioned earlier was based on an *aggressive* strategy. This policy was seen to be quite effective with the Keepers able to pass the ball successfully for a few seconds on average. This policy was run 50 times and the statistical information taken from it.

The evolved strategy was based on a Evolutionary Algorithm(EA) representation of the parameters to be evolved. It used a fixed size genome which encoded the two parameters that were identified as important for the Keepers control mechanism. These two parameters were floating point values which encoded the “sight” and kicking range parameters of the individual keepers respectively. These two values were statically set to 100.0 and 1.25 respectively in the hand-coded strategy. These values were seen to work effectively; the goal then was to evolve values for the parameters which would allow the Keepers to keep the ball for as long as and perhaps longer than the statically defined hand-coded policy.

The setup for the evolved strategy was also quite simple, using ECJ and its parameter file specification of an evolutionary process. The representation of an individual was made up of two float values, these two values or genes together constituted a genome. This genome would then be manipulated via the process of single point crossover and mutation along with the raw fitness function (number of passes) over a maximum of 100 generations with a population size of 75. The genome would be represented internally in ECJ as a *FloatVectorIndividual*. This genome would then be modified via crossover and mutation. The selection mechanism was a simple pipeline of tournament selection. In the case of *FloatVectorIndividual* mutation would be done using the default mutation method for *FloatVectorsIndividuals.* This default method works by randomizing genes within a predefined range.

The probability of crossover would be set arbitrarily at .7 in the initial runs and would be modified over future runs for further analysis. Similarly mutation probability was set at .05 initially.
Once the values for the parameters had been modified using ECJ the values for these parameters were fed back into the actual Keeper code the relative fitness of these new parameters would then be evaluated over 50 runs. This process would be continued until an ideal individual was found, in this case an ideal individual had to be able to pass consistently. The result of these experiments would be a comparison of the fittest evolved-strategy versus the hand-coded strategy.

5. Results

The total number of passes were recorded for each run up to 50 runs, the number of passes which corresponded directly with the fitness of an individual in the evolved case turned out to be very similar to that of the hand-coded strategy. It didn't take long for the evolved strategy to arrive at optimal value's for the parameter's, in fact by generation 38 ECJ had evolved similar parameters for the two characteristics. So by generation 38 the evolved strategy could potentially equal that of the hand-coded policy.

The average number of passes which was equal to 14.38 in the hand-coded policy and 11.23 in the evolved strategy turned out to be similar. Similarly the maximum number of passes was equal to 34 in the hand-coded and 27 in the evolved strategy respectively. This similarity is not all that surprising given that the evolved values for the parameters were very similar. The same value was evolved for the sight characteristic while a slightly smaller value was evolved for the possession characteristic. These parameters also have a limited impact on the final solution due to the limitations the *aggressive* strategy introduces. Also the parameters set in the hand-coded solution were optimal and hence the evolutionary process latched on to these same optimal values and this in turn leads to the similarity in the solutions. What perhaps would have been more interesting in terms of demonstrating the power of the evolutionary approach was when looking at earlier stages of the evolution when the parameters evolved were not as effective, but this is not was being evaluated; only an optimal result (the fittest) would be used as a comparison as like has to be compared to like.

6. Conclusions & Future Work

Hand-coding of solutions to problems is something which programmers associate with some high art, a sort of black magic which requires a lot of learning, experience and intuition. It is something which develops over time so the whole idea that a computer could somehow program itself using some sort of Darwinian evolutionary system is at first quite starting and alien. It is a revolutionary idea. But a revolution can only bring about lasting change for the better if it offers an effective alternative to what currently exists. This has already been demonstrated in some areas of design[1]. With this in mind the EA approach has to be proven or at least demonstrated to be an effective alternative to traditional hand-coded approaches. This paper was an attempt to quantify this effectiveness.
It turned out that the evolutionary approach came up with a *Keepaway* strategy which had similar attributes as the hand-coded strategy in terms of how effective it was at keeping the ball. In this regard perhaps one can conclude that yes the evolutionary technique as suggested by Koza[1,6] and others is an effective way of getting practical results to programming problems from computers. In one respect it is quite disheartening to think that the evolutionary technique didn't evolve a vastly superior solution but at least it has a similar ability and perhaps can be improved upon in the future. The limitations of the hand-coded *aggressive* strategy with it's lack of finesse in terms of how it operates made this study less valid than it could have been had more intelligent solutions been used. The very fact that only two parameters were being evolved could be said to have acted as a girdle forcing the final results to have a

certain shape. Still, even given these limitations a quantitive comparison was found which demonstrated that evolved strategies can be just as effective as hand-coded ones.
Future work might involve the use of a more effective passing strategy and a better measure of Keeper fitness which depends on time on the ball, how many passes completed successfully and how far on average the ball was kept away from the Attacker. This more realistic and powerful fitness measure may lead the evolutionary process to a better overall solution. Also an investigation into the effectiveness of some of the more refined strategies as proposed by Stone et al[9,10] and Gustafson[3,8] rather than the simple *aggressiv*e strategy could be analyzed and the effectiveness or otherwise of their strategies could be investigated.

References

1. Koza, John R. 1992. Evolutionary Algorithm - On the Programming of Computers by Means of Natural Selection. The MIT Press
2. Luke, S. 1998. Evolutionary Algorithm produced competitive soccer softbot teams for robocup97. In Genetic Programming (GP98) conference proceedings, Madison.
3. Gustafson, S. 2000 Layered learning in Evolutionary Algorithm for a co-operative robot soccer problem. Masters Thesis. Kansas State University.
4. Banzhaf, W. Nordin, P et al. 1998. Evolutionary Algorithm – An Introduction. Morgan Kaufmann Publishers Inc.
5. Kuhlmann, G.; Stone, P. 2003. Progress in Learning 3 vs. 2 *Keepaway*.Systems, Man and Cybernetics, 2003. IEEE International Conference on
   Volume 1, 5-8 Oct. 2003
6. Koza, John R, Keane, Martin A. et al. 2003 Evolutionary Algorithm IV: Routine Human-Competitive Machine Intelligence.Kluwer Academic Publishers
7. Luke, S.Cioffi-Revilla, Panait, L.,Sullivan, K. 2004. MASON: A New Multi-Agent Simulation Toolkit. *Proceedings of the 2004 SwarmFest Workshop.*
8. Gustafson, S et al. 2001. Layered Learning in Evolutionary Algorithm for a Cooperative Robot Soccer Problem. Proceedings of EuroGP'2001, v. 2038 of LNCS, pages 291--301, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.
9. Stone, P. Sutton, R S. Kuhlmann, G. 2003. Reinforcement Learning for RoboCup-Soccer *Keepaway*. Adaptive Behavior, 13(3):165-188.
10. Stone, P. Balch, T. Kreatzschmarr G. 2001. Keeping the Ball from CMUnited-99. Peter Stone and David McAllester. "RoboCup-2000: Robot Soccer World Cup IV. Springer Verlag, Berlin.
11. Luke, S. 2002 ECJ: A Java-based Evolutionary Computation and Genetic Programming Research System. http://cs.gmu.edu/~eclab/projects/ecj/