

EVOLVING GOALKEEPER BEHAVIOURS FOR SIMULATED SOCCER COMPETITION

Christopher Lazarus and Huosheng Hu

Department of Computer Science, University of Essex, Wivenhoe Park,
Colchester CO4 3SQ, United Kingdom
Email: clazar@essex.ac.uk, hhu@essex.ac.uk

Abstract

This paper describes our approach on evolving a robot controller to generate a set of goalkeeper behaviours in the Simulation League of the RoboCup competition. The goalkeeper agent is a software construct that performs in a simulated environment provided by the Soccer Server. The soccer server is a noisy environment provided as a testing ground for various artificial intelligence techniques used by different research teams around the world. Our experiments were set out to determine the conditions that need to be met for an evolved goalkeeper agent to perform adequately in a defensive situation. A framework was developed to test the agent's performance in the simulator. The results show that our approach was able to produce a robust robot controller.

Keywords: Simulated Soccer, Evolutionary Robotics, Genetic Programming.

1. Introduction

Robot controller development is an area that traditionally lies under the domain of human programming. This area is an important part of robotic research and therefore has been the most active area in robotics. Traditional robotic research has been to break down the required behaviours of a robot into smaller manageable parts and architectures are built to make them work. On the other hand, this is often accomplished as a result of a self-organizing process in the Evolutionary Robotics (ER) approach [3]. It is well known that natural selection combined with genetics has the power of finding out ingenious solutions to problems that are similar to the one facing a robot controller designer. Researchers hand code solutions to robotic control and navigation problem. Successful solutions tend to be complex, rigid and highly customised. During recent years, researchers began to use techniques derived from Evolutionary Computation (EC) for developing robot controllers. The major benefits in using this technique are automatic programming and adaptability of solutions developed.

Programming a robot to perform various tasks and ultimately relate to a single goal is not trivial. Our efforts focus on evolving behaviours for a goalkeeper agent by adopting the Genetic Programming (GP) technique and the focus of the research was on the design of a set of specifications for the desired behaviours. These behaviours emerge due to evolutionary pressure set out by the designer [4]. The work presented here is based on our previous work on evolving navigation and obstacle avoidance behaviours for the wall-following problem. In that work, the evolution of a robot's behaviours involves the optimisation of a single objective function [5]. However, our current work involves the optimisation of multiple objectives [6].

There have been many attempts by researchers to develop and improve ways on robot evolutions. In many ways they are able to show that their solutions can be similar if not better than hand-coded solution, i.e. the traditional robot navigation solution. These works include evolution of both simulated and real robots. The RoboCup domain **Error! Reference source not found.** in particular is a rich domain for the exploration of these techniques. It gives researchers a platform where they can compare the performance of their techniques in a like environment¹ [7]. Given that more than one robot is involved in a soccer match, RoboCup also gives researchers a platform on which to explore multi-robot cooperation algorithms. Accordingly, several teams have used GP in evolving their soccer team [8] and [4]. Lundberg did a comparison of these works [9].

This paper is organised as follows. We describe our experiments in more detail in section 2 including the methodology that we chose and shows in a conceptual manner the architecture of the system that we have developed. In section 3, the implementation of our architecture is described. The experimental data are presented and analysed in section 4. Finally, we show what our experiments have uncovered and what further work could be done.

¹ In the Simulation League, the soccer server was designed to be a noisy environment. Therefore any solution posed by researchers needs to be able to handle the uncertainty of sensory data.

2. Methodology

The focus of our experiments is to find out the effects of using multiple objective fitness functions in our GP implementation. To accomplish this, we have devised a *training ground* situation to test the evolved behaviours of our goalkeeper. The architecture consists of a soccer server, a coach trainer agent, a hand-coded attacker agent and an evolving goalkeeper agent. The soccer server **Error! Reference source not found.** acts as a server to all of our agent clients. It accepts action commands and sends back sensory information. In other words, the soccer server runs the simulation. The coach agent performs the synchronisation between all of the agents. It does this by relaying messages to let the agents know when a trial begins and ends. It also resets the positions of the agents and the ball. The hand-coded attacker agent is a simple agent that kicks the ball towards the centre of the goal. The goalkeeper agent is the agent that has the capability to evolve due to the GP system that generates its control program.

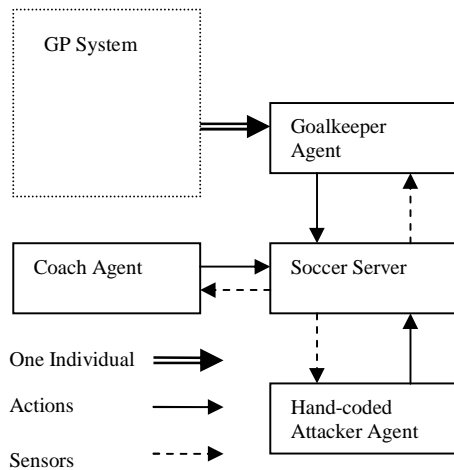


Figure 1 System Configuration

Our control experiment uses a *random* goalkeeper agent that has exactly the same capability as our GP goalkeeper. The only difference is that the *random* goalkeeper agent sends randomly generated commands to the soccer server instead of being generated by GP. Nevertheless, its fitness is still evaluated during the experiment. We run both random and evolving goalkeeper experiments to collect the data. We compared both of the goalkeeper's performance to measure the effectiveness of the GP implementation.

In designing our functions and terminals, we noted that GP may give surprising results even with a simple set of functions and terminals although it is generally accepted that GP is a robust technique [10]. Therefore, if the performance of the GP algorithm is not quite what we expect then more complex functions and terminals may be added to the gene pool. Nevertheless, one of the attractions of using GP is that detailed knowledge of the

problem domain is not essential. The focus of the problem shifts from the careful design of the behaviours to the designing of specifications that encourages emergence of the desired behaviours [4]. In this case, specifications can be equated as the fitness criterion of each individual within the GP population.

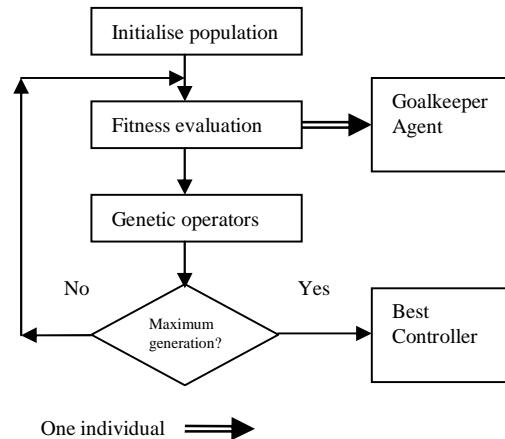


Figure 2 GP System

Therefore, we see the need for correctly representing the problem domain in the fitness functions. In our previous work we have used a single fitness function for evaluating a robot's navigation solution [5]. In our current work, we use multiple fitness functions. A goalkeeper has a single overriding objective, which is to stop any goal from being scored into its own net. Nevertheless, to achieve this, there is a set of other objectives that need to be met. The first objective concerns navigation, another is localisation (position must be kept near the centre of the goal), another is blocking the ball trajectory, the ability to kick the ball out of the goal area and lastly to catch the ball when possible. Therefore the aim of our optimisation is to transform the vector objectives to a scalar version and then proceed with the assignment of fitness values to each individual.

3. Implementation

The GP system we used in our experiments is lil-gp [11]. We also used the client networking library libclient [12]. We developed a coach agent to provide synchronisation and monitoring during the experiment run. The coach agent resets the experiment setting before the start of each trial case. This involves placing the goalkeeper, ball and attacker in their respective starting positions. Once in place, the ball is kicked by the attacker in a straight line directly towards the goal. There are 10 different ball placements for each evaluation (see Figure 3). In short each individual in the GP population representing the goalkeeper is tested on its performance in defending the goal from 10 different ball-starting positions. The attacking area in front of the goal is divided equally into 10 sectors. At the start of each

experiment, the 10 starting positions for the attacker is randomly generated within each of the 10 attacking sectors. Each attacker starting position is then tested sequentially in a clockwise order for each individual in the population. This variation in the attacker's starting position should enable a robust controller to be evolved. The evolved behaviours are not biased towards a particular attacking direction only. We evaluated each individual by parsing the expression trees and sending a command every 10ms to the soccer server. Normally this would be set to 100ms but we have reduced it to 10ms to speed up the simulation². Only one command is allowed per 10ms.

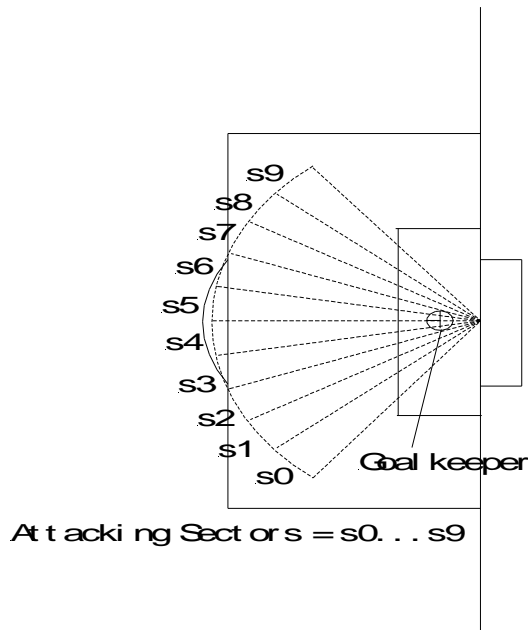


Figure 3 A segment of the soccer field showing a goalkeeper at location 50, 0 with the attacking area divided equally into 10 sectors. Figure is not according to scale.

Evaluation time

The length of timesteps allowed for an attacker to attack and score a goal has a direct impact on the length of time each evaluation takes. For the purpose of the experiments each evaluation takes 40 timesteps. Each timestep corresponds to a set period of 10ms. We allow the attacker 40 timesteps per trial to give it enough time to try to score a goal.

Fitness measure

The fitness evaluation we adopted is called the *Weighted Sum Approach*

² Only one command is allowed per 100ms. The rules of the competition stipulate that no flooding of the soccer server is allowed. If this rule is violated, the soccer server may run only one action command arbitrarily in a single time step and ignore the rest.

$$f = \sum_{i=1}^n w_i f_i$$

where $0.0 \leq f \leq 1.0$. f denotes the overall fitness of an individual, n denotes the number of fitness measures, w_i denotes the weights and f_i denotes the objectives to be optimised [13]. There are 5 fitness cases that influence the evolution of desired behaviours.

Fitness cases

- f_1 = Ball saving
- f_2 = Localisation
- f_3 = Ball locate
- f_4 = Movements
- f_5 = Positioning

Max Generation	50
Population	
Type	Generational
Size	100
Initialisation	Half and Half
Selection Type	Tournament with size 7
Genetic Operators	
Crossover rate	0.9
Mutation Rate	0.1
Function Set	iflte, add, sub, mult, div, sin, cos, catch, dash, kick, turn, put, ADF0
Terminal Set	play_dir, play_pos, dist_to_ball, dir_to_ball, dir_to_ogool, dist_to_ogool, dist_to_opp, dist_to_opp, rand, get0, get1, ARG0, ARG1

Table 1 GP Parameter Settings

The rationales for the choice of the fitness measures above are that we want the evolved goalkeeper to be mobile but moving towards the ball and intercepting it. The movements and positioning fitness cases reflect this desire. Fitness f_4 and f_5 might seem to overlap in functionality but without fitness f_4 , the movement of the ball instead of the goalkeeper might be the cause for the proximity of the goalkeeper to the ball.

Therefore, by encouraging selection of individuals that are mobile ensures that there are cases where the proximity of a goalkeeper to a ball is caused by the goalkeeper's own movements. We gave fitness f_4 a higher weight because of similar reasons. Nevertheless, fitness f_4 and f_5 are ineffective if they don't result in a ball saving event. So we give a higher weight for behaviours that resulted in a ball saving event.

Fitness case 1 is given by the equation

$$f_1 = 1 - \frac{g}{t_{\max}}$$

where g is the goal scored by an attacker and t_{\max} is the maximum trial case per evaluation

Fitness f_2 is needed to measure the goalkeeper's ability to estimate its own location within the playing field. If both initial and ending position of the goalkeeper within each trial is estimated then localisation is deemed as successful and fitness f_2 is allocated.

Fitness f_3 measures the goalkeeper's ability to locate the ball. This is done by checking whether the goalkeeper is able to estimate the distance between itself and the ball at the start and end of a trial. The goalkeeper is only able to do this if the ball is in its field of vision. Therefore this fitness can also be equated with the ability of the goalkeeper in turning towards the ball.

Fitness f_4 is as follows

f_4 if p_s is equal to p_e then

no movement is detected so $f_2 = 0$

else

movement detected so $f_2 = 1$

where p_s is the position of goalkeeper at the start of an evaluation, and p_e is the position of goalkeeper at the end of an evaluation

Fitness f_5 is given below. This fitness measure uses a logarithmic scale function of the distance between the goalkeeper and the ball at the start of an evaluation and also at the end of an evaluation. This has the effect of ensuring that the relationship between the distance between the goalkeeper and the ball is logarithmic. This ensures that the fitness decreases following a logarithmic curve as the distance between the ball and the goalkeeper becomes larger.

f_5 if $(1 - \frac{\log(d_e)}{\log(d_s)})$ is less than 0 then

set f_3 to 0

else

set f_3 to $1 - \frac{\log(d_e)}{\log(d_s)}$

where d_s is the distance between goalkeeper and ball at the start of an evaluation greater than 1. d_e is the distance between goalkeeper and ball at the end of an evaluation greater than 1

Termination-criteria

Termination of the GP run is set to the number of maximum generation predetermined at the beginning of the GP run.

4. Experimental Results and Analysis

Table 2 presents all experiments that have been carried out in this research and the results are collected and summarised in Figure 5.

Experiment	Behaviour generated by	Memo ry	ADF	Weights
1	Random	N/A	N/A	Equal
2	Random	N/A	N/A	Lexicographic
3	GP	Yes	No	Equal
4	GP	Yes	No	Lexicographic
5	GP	No	Yes	Equal
6	GP	No	Yes	Lexicographic
7	GP	Yes	Yes	Equal
8	GP	Yes	Yes	Lexicographic
9	GP	Yes	Yes	SQRT (Lex.)

Table 2 Experiment runs

The first two experiments, where the goalkeeper's behaviour was generated randomly, are used as a benchmark for subsequent experiments. The starting position for the goalkeeper is set at location 50, 0 on the playing field. Three sets of weights were used during the experiments.

Weights Fitness	Equal	Lexicographic	Square root (Lexicographic)
1	0.2	0.9	0.685942
2	0.2	0.09	0.216914
3	0.2	0.009	0.068594
4	0.2	0.0009	0.021691
5	0.2	0.00009	0.006859

Table 3 Weights used during the experiments

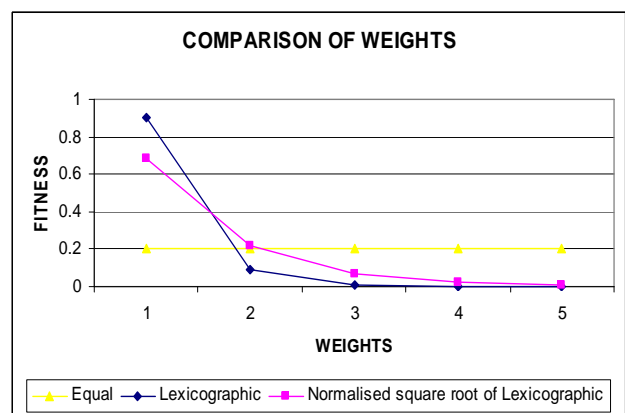


Figure 4 The fitness distribution when lexicographic weights are normalised

Run Mean Standardised Fitness

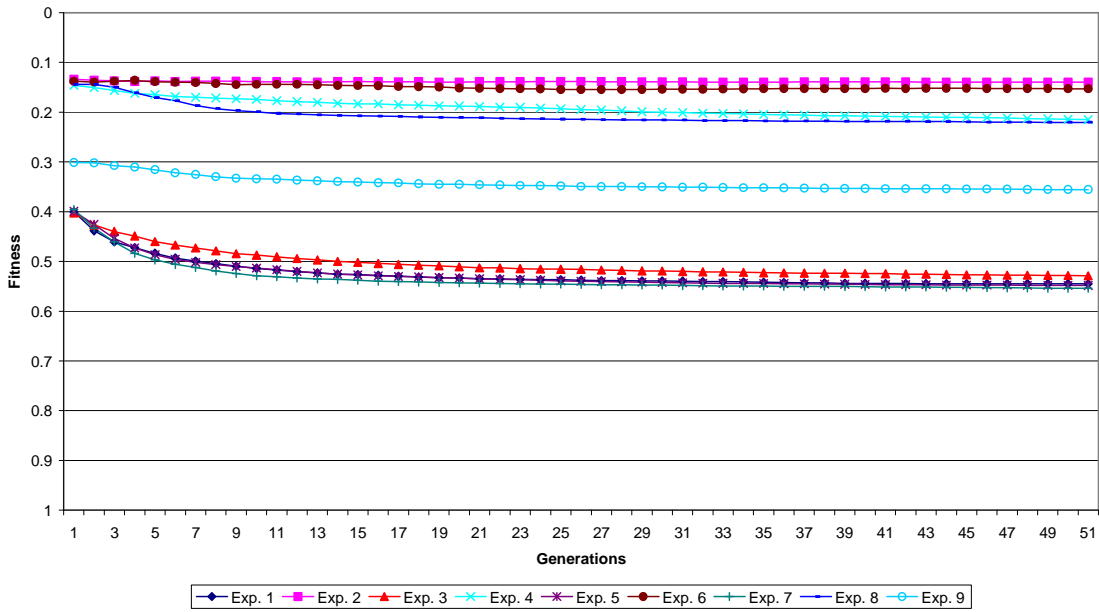


Figure 5 Comparison of mean population fitness

Best of Run Individual Standardised Fitness

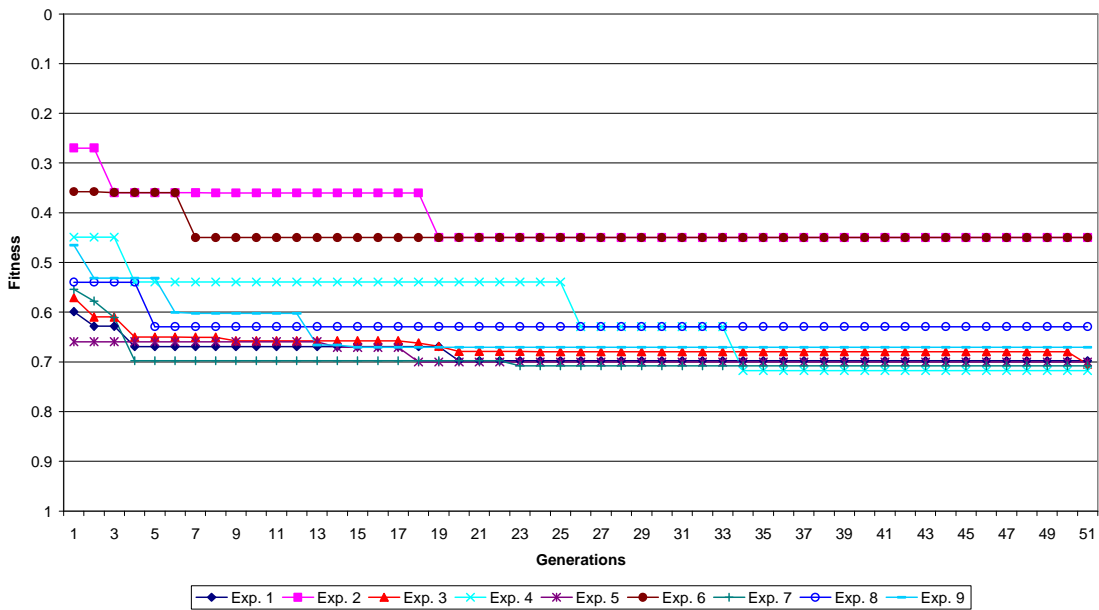


Figure 6 Comparison of individual fitness

As shown in Table 3, one set of weights is designed to give equal importance to all of the 5 fitness measures. The second set of weights is designed to bias the importance of the fitness measures in a lexicographical manner in terms of fitness. The third set of weights is derived from normalised square roots of the lexicographic weights. The third set of weights has a smoother distribution compared to the second set. This is shown in Figure 4. This allows for the fitness measures to be more uniformly distributed

across the fitness range but would still be skewed according to the fitness measures ranking.

All of the experimental runs based on lexicographic weights have lower overall fitness when compared with runs that are based on equal weights. Lexicographic weights biased the fitness measurement too much towards fitness 1 which is ball saving. This results in GP not being able to capitalise on the contributions made by the other fitness measures. This also shows that even though ball saving is

the main objective, ball saving is not enough on its own to allow GP to explore potential solutions. By specifying that all of the fitness measures have equal weights, we see that the fitness is distinctively higher compared to using lexicographic weights. We can also see this assertion by comparing both of the random runs. The random run that uses equal weights has higher fitness compared to the random run that uses lexicographic weights

Experiments 2, 3, 4 and 5 were run to observe whether the GP performance would be enhanced by the addition of either memory or ADF. The results show that the GP with ADF has higher fitness than the GP with memory when equal weights are used. In contrast, the GP with memory has higher fitness than the GP with ADF when lexicographic weights are used. Looking at both the equal weights and the lexicographic weights groups, we can see that in both groups, the highest performing GP is the one that implements both memory and ADF in their architecture. The addition of either ADF or memory in isolation does improve the performance of the GP but its performance increases further if both of them are implemented together.

When we compared performances of best of run individuals in contrast to populations, we start to see that the distinction caused by different weights set becomes less obvious. Experiments that use the lexicographic weights, has best of run individual nearing the performances from the experiments that uses equal weights. An additional experiment (experiment 9) that uses the normalised lexicographic weights was run to observe the effects of smoothing the distribution of the fitness measures. The population fitness for this experiment shows a marked improvement. However its population fitness is still below the population fitness for experiments that uses equal weights.

5. Conclusions and Future Work

The selection of weights in the *Weighted Sum Approach* as we have shown in this paper is important to the performance of the GP implementation. In this approach most weight selection problems are solved heuristically. Considering the simulation results, each of the fitness measures is equally important. Attempts to bias the fitness measures by adjusting the weights shows lower overall population fitness. Nevertheless, the fittest individual within the population where the weights are not equally distributed still manages to be fitness-competitive to the fittest individual using equal weights. Other multi-objective optimisation methods are described by [13] and [6]. The addition of memory and ADF as part of an agent's memory and an individual's program tree respectively for the duration of the run increases the GP's performance. This corresponds with the increase in difficulty of the task that the goalkeeper agent is given.

The overall results show that the best individual reaches a plateau at approximately two thirds of the fitness range. This shows that although GP manages to evolve a good set of behaviours, they are still short of

optimum fitness. The next stage of this work will be to investigate on ways to improve the performance of the evolved goalkeeper further.

Acknowledgements: We wish to thank Prof. Edward Tsang and Dr. John Ford for their useful inputs and advices during this research.

References

- [1] The RoboCup Federation, "RoboCup Official Site", <http://www.robocup.org>, 2003
- [2] I. Noda, "Soccer Server System", <http://sserver.sourceforge.net/NEW/>, 2003
- [3] S. Nolfi, "Evolutionary Robotics: Exploiting the full power of self-organization," *Connection Science*, vol. 10, pp. 167-183, 1998.
- [4] D. Andre and A. Teller, "Evolving Team Darwin United," in *RoboCup-98: Robot Soccer World Cup II*, M. Asada, Ed.: Springer-Verlag, 1998.
- [5] C. Lazarus and H. Hu, "Using Genetic Programming to Evolve Robot Behaviours," Proceedings of the 3rd British Workshop on Towards Intelligent Mobile Robots (TIMR) '01, Manchester, 2001.
- [6] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multi-objective evolutionary algorithms: Empirical results (Revised Version) Technical Report 70," Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich 1999.
- [7] G. Adorni, S. Cagnoni, and M. Mordonini, "Genetic Programming of a Goal-Keeper Control Strategy for the RoboCup Middle Size Competition," presented at Genetic Programming: Second European Workshop, EuroGP'99, Goteberg, Sweden, 1999.
- [8] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-Evolving Soccer Softbot Team Coordination with Genetic Programming," presented at Proceedings of The First International Workshop on RoboCup, IJCAI-97, Nagoya, Japan, 1997.
- [9] J. Lundberg, "Survey over Genetic Programming Approaches to RoboCup", <http://www.dsv.su.se/~johank/courses/int4/robocup/papers/1999/int7/lundberg.pdf>, 2003
- [10] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, "Genetic Programming An Introduction," Morgan Kaufmann Publishers, Inc., 1998, pp. 112.
- [11] B. Punch and D. Zongker, "lil-gp Genetic Programming System", <http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html>, 2003
- [12] I. Noda, "libsclient", <http://www.isi.edu/soar/galk/-RoboCup/Libs/libsclient4.01.tar.gz>, 2003
- [13] C. M. Fonseca and P. J. Fleming, "Multi-objective Optimisation," in *Evolutionary Computations 2*, T. Bäck, D. Fogel, B., and Z. Michalewicz, Eds. UK: IOP Publishing Ltd., 2000.