Pareto-dominance Based MOGP for Evolving Soccer Agents

Christopher Lazarus
Department of Computer Science and Mathematics
Tunku Abdul Rahman University College
Kuala Lumpur, Malaysia
Email: christopherl@acd.tarc.edu.my

***Abstract*—Robot behaviour generation is an attractive option to automatically produce robot controllers. Most high-level robot behaviours comprise multiple objectives that may be conflicting with each other. This research describes experiments using two Pareto-dominance based algorithms together with a Multi-objective Genetic Programming (MOGP) framework to evolve high-level robot behaviours using only primitive commands. The performance of hand-coded controllers are compared against controllers evolved using the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) algorithms. An additional comparison is also performed against controllers evolved using the weighted sum fitness function. The experiment results show that the Pareto-dominance based MOGP performed better than the hand-coded and the weighted sum evolved controllers.**

I. Introduction

In this work, Genetic Programming (GP) was used to evolve a set of goalkeeping behaviours for a goalkeeper agent. The use of canonical GP coupled with weighted sum fitness measure allows the generation of control programs that are acceptable but falls short of the hand-coded versions. Pareto-dominance based Multi-objective Genetic Programming (MOGP) is constructed to improve the performance of the GP system. Although the use of weighted sum fitness measure has had some success on some particular problems, it is widely known that non-dominated solutions located in the concave regions of the tradeoff surface cannot be obtained. Messac et al. [1], describes why some objective functions fail to capture Pareto solutions and further highlighted conditions to determine the ability of certain objective functions to capture a Pareto point. Recently, researchers in the Evolutionary Multi-objective Optimisation Algorithm (EMOO) field have produced various alternate and complementary techniques that also attempt to ameliorate this issue.

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) are chosen as a basis for the MOGP schemes. Both algorithms are selected because of their efficiency, their ability to outperform non-elitist approaches [2], often used in performance comparison using standard metrics [3], [4] and the removal of the need for weighting parameters in the weighted sum fitness measure technique. Nevertheless, due to the output of this class of multi-objective optimisation approaches, there is a need for the decision maker to select a preference on the optimal solution based on tradeoffs or constraints. In this research, a post-evolution competition is held on members of the Pareto set to help select an optimal control program.

II. Multi-Objective Optimisation

Multi-objective Optimisation (MOO) has mostly been used in the engineering and operation research field [5], [6], [7]. This field arises due to a pressing need when researchers attempt to solve real world problems. In the research laboratory, this algorithm is typically used to solve simpler well-known problems. There are many encouraging results prompting researchers to use it to tackle much harder problems. Traditional MOO involves optimising an objective from a set of objectives one at a time iteratively. This is generally recognised to be time consuming.

Researchers acknowledge that in using Evolutionary Algorithm (EA), many solutions could be explored in one iteration. This is because EA typically utilise a set of individuals where each individual represents a potential solution. At each generation during the evolutionary process, all the individuals in the population are evaluated. That is multiple potential solutions can be explored simultaneously in one run. Therefore, the combination of MOO and EA is seen as the natural next step in multi-objective optimisation.

During the last few decades, due in part to the rise in computing power and the improvements in EA techniques, Genetic Algorithm (GA) has become a popular and demonstrably successful EA strategy. This factor has arguably encouraged more researchers to use GA as the evolutionary technique of choice in combination with the MOO approach. Examples of more widely known EMOO approaches are the non Pareto-based approaches such as Vector Evaluated Genetic Algorithm (VEGA) [8], Pareto-based approaches such as Multiple Objective Genetic Algorithm (MOGA) [9], Non-dominated Sorting Genetic Algorithm (NSGA) [10], Niched Pareto Genetic Algorithm (NPGA) [11], Pareto Archive Evolution Strategy (PAES) [12], Strength Pareto Evolutionary Algorithm (SPEA) [4].

Zitzler et al. [3] did a comprehensive discourse and comparison on various multi-objective optimisation approaches. Similarly, a tutorial on evolutionary multi-objective optimi-

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.49

IEEE computer society

sation was published by Coello [13]. For a comprehensive mathematical discussion on multi-objective optimisation, the reader is guided to an excellent monograph by Miettinen [7].

III. Pareto-dominance Based Approach

Both the NSGA-II and SPEA2 algorithms are popular amongst EMOO researchers due to their robustness in solving fragmented search space problems. In EMOO terminology, this class of search space is also known as non-convex. In the case of the weighted sum approaches, the global optimum of the objective fitness for any set of positive weights is always a non-dominated solution for a multi-objective problem [14]. However, in cases where the non-dominated solutions are located in concave regions of the tradeoff surface or the Pareto-optimal set, weighted sum approaches cannot obtain these non-dominated solutions. The reason is because for non-dominated solutions that are located in the concave regions, their corresponding global optimum is suboptimal [15].

IV. Experiment Setting

The experiments listed below were performed using a hand-coded attacker with Kick Powers of 100 (KP100) and Kick Powers of 50 (KP50):

- Static Goalkeeper vs. Hand-coded Attacker
- Random Goalkeeper vs. Hand-coded Attacker
- Hand-coded Reactive Goalkeeper vs. Hand-coded Attacker
- Hand-coded Predictive goalkeeper vs. Hand-coded Attacker
- Weighted Sum Goalkeeper vs. Hand-coded Attacker
- MOGP (NSGA-II) Goalkeeper vs. Hand-coded Attacker
- MOGP (SPEA2) Goalkeeper vs. Hand-coded Attacker

The placements and facing directions of the attacker including the ball are set at the beginning of each experiment. The ball is placed at a predetermined distance from the attacker as show in Figure 1. This means that the attacker needs to approach the ball before it can kick the ball towards the goal. This is done to give the goalkeeper a minimal time to prepare for the oncoming attack and also to mimic a more general attacking scenario.

TABLE I
Attacker initial positions and facing directions for each attacking sector

Attacker	Attacker's Location (x, y)	Attacker's Facing Direction	Ball's Location (x, y)
0	(41.00, 14.48)	-5.11	(43.00, 12.10)
1	(39.00, 13.24)	17.57	(41.00, 11.07)
2	(37.00, 10.47)	-7.59	(39.00, 8.76)
3	(34.00, 5.04)	-3.25	(37.00, 4.21)
4	(34.00, 3.10)	-15.49	(37.00, 2.59)
5	(34.00, -2.44)	-15.41	(37.00, -2.04)
6	(34.00, -3.83)	0.00	(37.00, -3.20)
7	(36.00, -8.54)	-10.50	(38.00, -7.14)
8	(39.00, -12.49)	-12.64	(41.00, -10.44)
9	(42.00, -15.05)	-25.65	(43.00, -12.58)

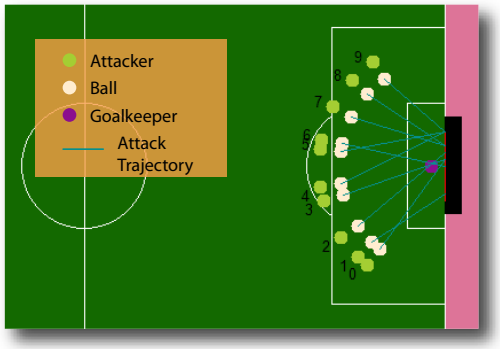

Fig. 1. A segment of the soccer field showing the goalkeeper, ball and attacker. The endpoints of the kick directions are varied initially using a predetermined seed but remains the same during the evolutionary run.

A. Parameters

The population size for the weighted sum experiment is set to 100 individuals. The NSGA-II algorithm uses elitism to ensure found solutions are not lost in later generations during its evolutionary run. To achieve this, it uses a secondary population that has the same size as the main population. Therefore, during the evolutionary run, the total population size is 200 individuals. The SPEA2 experiment also utilises elitism, which involves the explicit use of an archive that is the same size as the main population size. Therefore, the population size used for the SPEA2 experiment is also 200 individuals; 100 individuals for the main population and 100 individuals for the archive.

All of the experiments conducted rely on actual match time perceptions. The soccer server parameter *fullstate* is set to *off*. Setting this parameter to *on* indicates to the soccer server that all perceptions that are given to the agents are complete and without any uncertainty. This parameter is set to *off* because controllers should be able to work despite or with the noise switched on to emulate closely real match situations[1]. Nevertheless, the experiments are contrived scenarios but only as much as the attack distances and directions are predetermined and repeatable. However, it is more than likely that some of these positions and scenarios could occur in a real match.

Experiments involving evolution is set to terminate at generation 50. The size of the population and the length of the evolutionary run is chosen as a compromise between having enough individuals to represent potential solutions and the necessary length of evolutionary time for solutions to be evolved and the length of time to perform the evolution. Since each individual has to be evaluated by conducting 10 trials per generation, and each trial consists of 40 timesteps (400 ms), the evolutionary period is lengthy.

[1] A ball's movement can only be changed by other agents, including players and trainers. However, the soccer server adds some noise to its direction and velocity. Additional noise is also included in the visual ball information that is given to an agent by the soccer server.

B. Fitness Functions

There are five objective fitness functions used. The weighted sum experiment uses equal weights of 0.2 for each of the objective fitness functions. Both crossover (with a rate of 0.9) and mutation (with a rate of 0.1) are used as the genetic operators for all of the evolutionary runs. The selection mechanism for the weighted sum experiment is the tournament selection with a size of 7. A binary selection method is used for the NSGA-II experiment and individuals were compared using the crowded comparison operator ($\prec_n$) in Equation 1.

$$i \prec_n j \quad \text{if } (i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})) \tag{1}$$

Selection for mating in SPEA2 is also based on binary tournaments. Individuals designated as parents for mating are selected from both the main and archive population. Their selection are based on a comparison of their fitness values respectively. The fitness $F(i)$ for each individual i is given as a strength value $S(i) \in [0, 1]$. $S(i)$ represents the number of individuals in the population that i is equal to or dominates in terms of objective values divided by the population size plus one. The fitness is to be minimised hence the individual with lower fitness value will be chosen as a parent for mating [16].

$$f_1 = \frac{b}{t_{max}} \tag{2}$$

The goalkeeper is only rewarded directly for successful ball-catching and this is reflected by fitness function f_1. Any occurrence of blocking events would be incidental and it would be interesting to note whether the goalkeeper evolves a controller that regards ball-catching as paramount or still includes blocking as part of its defensive repertoire. Fitness case 1, f_1, is given in Equation 2 above, where b is the total number of balls successfully caught by a goalkeeper during an evaluation and t_{max} is the maximum number of trial cases per evaluation.

Fitness case 2 (f_2) is needed to measure the goalkeepers 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 tness is allocated. The position of the goalkeeper is provided directly by the localisation routine of the goalkeeper agent. This value is only computable if the goalkeeper is within the field of play due to the availability of landmarks within the field of play that are used by the localisation routine.

Fitness case 3 (f_3) measures the goalkeepers 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 tness can also be equated with the ability of the goalkeeper in turning towards the ball.

$$f_4 = \begin{cases} 0 & p_s = p_e \text{ (no movement is detected)} \\ 1 & p_s \neq p_e \iff p_e \leq p_s - 2.0, \\ & p_e \geq p_s + 2.0 \text{ (movement detected)} \end{cases} \tag{3}$$

Fitness case 4 (f_4) is given in Equation 3, 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. The design of fitness case 4 (f_4) are mostly in response to an anomaly that was discovered while analysing the experiments. It appears that a turn command displaces the location of the goalkeeper within a range of ± 2.0. The reason is that these displacements are the result of slight inaccuracies introduced by the localisation routines.

Whenever a turn is executed, the viewpoint of the agent is changed and different objects that are used for localisations come into play. Even though these variations are small enough for localisations purposes, it resulted in rewarding a controller for movement event even when only a turn command was executed. The intention was for the controller to evolve movements using a dash command. A more accurate localisation algorithm would reduce these inaccuracies. Boer et. al. described an implementation of the *Particle Filter* as a more accurate localisation technique [17, pp. 88-91]. The DAInamite team has also adapted this particular localisation technique [18, pp. 58-66].

$$f_5 = \begin{cases} 0 & (1 - (\frac{\log d_e}{\log d_s})) < 0 \\ 1 - \frac{\log d_e}{\log d_s} & (1 - (\frac{\log d_e}{\log d_s})) \geq 0 \end{cases} \tag{4}$$

Fitness case 5 (f_5) is given in Equation 4 where d_s is the distance between goalkeeper and ball at the start of an evaluation greater than 1 and d_e is the distance between goalkeeper and ball at the end of an evaluation greater than 1. Fitness case 5 (f_5) is used to allocate fitness for goalkeepers that moves towards the ball within the trial period. 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.

The fitness points to be awarded is distributed along a logarithmic curve and 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. In other words, more fitness points is given the closer the goalkeeper is to the ball at the end of the trial period.

C. Post-evolution Trials

Post-evolution trials are used to choose one candidate solution from the set of evolved solutions. Each of the evolved solutions from the final population is tested using the same experiment scenarios as the online experiment. In these experiments, the agent's controller trees are no longer being evolved. The evolved controller trees from the population at the end of the GP run for each agent is loaded at the start

of the experiment. The agent then acts using this controller tree during the course of the run. An agent is assessed by performing 500 trials per run. An agent's performance is calculated based on the number of balls saved; that is the number of events where the ball is caught or blocked by the goalkeeper agent. By comparing the agents' performances, an optimal controller tree that is preferred can be determined from the final population.

The best individual from the weighted sum experiment, evolved during the evolutionary run, is selected to be used in one of the post-evolution trials. For both NSGA-II and SPEA2, trials are conducted on all of the individuals from their respective Pareto fronts. The best performing individual of the Pareto front is then chosen as the optimal solution. These post-evolution trials determines the performance level of an individual by executing the evolved controller. The execution of the controller allows the agent with that particular controller to act as a goalkeeper. This is repeated until 500 trials have elapsed. Note that for the post-evolution experiments, 50 evaluations of 10 trials each are conducted for each of the candidate solutions in order to judge its performance. The performance of an individual is directly determined by the number of goals prevented during its trials.

V. Experiment Results

This section shows the results of the experiments that have been conducted. Basically, the results can be divided into evolutionary result for the weighted sum, NSGA-II and SPEA2 evolution runs and the post evolution results. The post evolution results includes an analysis of which action nodes in the controller tree was activated and also counted. This analysis gives us a picture of which actions the GP algorithm has evolved as "important" and also gives us an idea why certain controllers' performance are better.

Two hand-coded goalkeepers were included for comparison purposes. The first hand-coded goalkeeper is a simple reactive goalkeeper that looks for the ball. If the ball is not located then the goalkeeper performs the *turn* action command until it sees the ball. When it has the ball within its sight, it would dash towards the ball. When the ball is judged to be within range, the goalkeeper would attempt to catch it.

The second hand-coded goalkeeper is a simple predictive goalkeeper. This goalkeeper checks whether the distance between itself and the ball is within the catchable distance and whether the ball is recently seen. If both of these conditions are true then the goalkeeper performs a catch command. If the goalkeeper is still far from the ball but the ball is recently seen then the goalkeeper infers the trajectory of the ball according to the ball's location and speed about 10 steps into the future. It would then perform a dash towards the predicted location with maximum speed. If a catch position is not available and the ball is far away and the ball probability[2] is less than 1, then the goalkeeper calculates a catch location and performs a dash towards the catch location. However, if a catch location is available, then the goalkeeper checks whether it is already at the catch location. If the ball is not seen at this location then it performs a 45° turn. Otherwise the goalkeeper performs a dash towards the catch location.

[2]The probability of the ball denotes how recent the ball's information is. The ball's probability starts at 1 and is multiplied by 0.5 at every time step that no sensor information concerning the ball is received. Thus, an agent's belief in the relevance of the ball information is decayed from the last time it saw the ball.

A. Baseline Controllers

The baseline controller used to represent the worst controller apart from a static controller is a controller that uses a randomiser to generate actions that are sent to the soccer server. As can be seen from Table II, the static controller is the worst performing controller.

TABLE II
Goalkeeper's comparative performance for attacker with KP100

Goalkeeper Type	Ball Caught	Blocked In-Field	Blocked Out-of-Field	Performance
Static	0	15	1	16 (3.2%)
Random	6	19	15	40 (8.0%)
Reactive	95	45	1	141 (28.2%)
Predictive	95	9	12	116 (23.2%)
Weighted Sum	0	30	9	39 (7.8%)
NSGA-II	49	19	15	83 (16.6%)
SPEA2	135	6	21	162 (32.4%)

There are two hand-coded controllers that were also evaluated in our experiments. These controllers provide further comparisons for the evolved controllers. The first controller is a reactive controller and the second controller is a simple predictive controller. What the latter controller predicts is the approximate location of the ball a few timesteps in the future. When KP100 is used, the reactive hand-coded goalkeeper performed slightly better (28.2%) than the predictive goalkeeper (23.2%). Lowering the attacker's kick power by half to KP50, the predictive hand-coded goalkeeper (55.2%) outperformed the reactive hand-coded goalkeeper (49.6%). These performance metrics are based on both the ball-catching and ball-blocking capabilities of the two hand-coded goalkeepers. Looking at these metrics in isolation allows us to understand the reason for the hand-coded goalkeepers' performances.

Using KP100, even though the reactive goalkeeper outperformed the predictive goalkeeper, in terms of ball-catching performance alone, it can be seen that they are both equal at 95. However, the reactive goalkeeper has more ball-blocked events (46 events) compared to the predictive goalkeeper (21 events). Additionally, a high proportion of the ball-blocked events for the reactive goalkeeper resulted in the ball being deflected within the soccer field. Because of the higher ball speed, the deflected ball speed is also higher resulting in lower second attempt scores by the attacker on the deflected balls.

A comparison of their performances also suggest that the predictive goalkeeper missed the ball more than the reactive goalkeeper. This means that it is harder to predict the future location of a ball than to get the current location of the ball when the ball speed is high. This is wholly dependant on the accuracy of the localisation routine. Therefore, a high

performance and a more accurate localisation routine might improve the predictive capability of a goalkeeper.

When the attacker's kick power is reduced to KP50, the predictive goalkeeper showed its true ability. In ball-catching alone, it was able to catch 276 balls compared to the reactive goalkeeper's 193 balls caught. This is a significant increase in ball-catching performance compared to the KP100 result. The rate of increase is shown in Table IV where each goalkeeper is only rated based on their ball-catching ability. The percentages are calculated based on the number of balls caught against the number of trials per goalkeeper at 500 trials. It is shown that the reactive goalkeeper has an increase of 19.6 compared to 38.8 for the predictive goalkeeper. Looking at the ball-blocked scores for both of the hand-coded goalkeepers, the reactive goalkeeper has 55 ball-blocked events and none for the predictive goalkeeper. This result shows that the predictive goalkeeper becomes more accurate in predicting the future location of the ball when the ball speed is lower. The predictive goalkeeper has no ball-blocked events at all, which tells us that either the predictive goalkeeper caught the ball or missed it completely.

In terms of controllers that were evolved by the GP algorithm, the lowest performing controller is the one that was evolved by the weighted sum based GP. Its performance is almost similar to the performance of a random goalkeeper. Its performance is lower than either of the reactive and predictive hand-coded goalkeeper. In constrast, the controllers that were evolved by both of the Pareto-based MOGP performed much better. The SPEA2 evolved controller has a higher performance than any of the hand-coded goalkeepers and in turn the NSGA-II evolved controller outperforms the SPEA2 controller. This clearly shows that in our specific case, a Pareto-based GP is able to evolve controllers that are better than the simple reactive and predictive hand-coded controllers as well as the weighted sum based GP.

TABLE III
GOALKEEPER'S COMPARATIVE PERFORMANCE FOR ATTACKER WITH KP50

Goalkeeper Type	Ball Caught	Blocked In-Field	Blocked Out-of-Field	Performance
Static	0	2	1	3 (0.6%)
Random	42	16	5	63 (12.6%)
Reactive	193	55	0	248 (49.6%)
Predictive	276	0	0	276 (55.2%)
Weighted Sum	0	1	1	2 (0.4%)
NSGA-II	208	2	0	210 (42.0%)
SPEA2	201	120	0	321 (64.2%)

The reason that the performance of the static goalkeeper using KP100 is high in terms of ball-blocking is that due to the high velocity of the ball, when it collides with the static goalkeeper, the ball rebounds off the static goalkeeper at a higher speed. This event requires that the attacker moves further in order to reach the ball and to attempt scoring for the second time. Reducing the power of the attacker's kick by half to 50 means that the rebound speed of the ball is also reduced. The ball would then be within a closer distance to the attacker in order for it to attempt to score at a second try. This effect is reflected in Table III where the ball-blocking success is only 3 out of 500 resulting in a low overall performance for the static goalkeeper at 0.6%.

This shows that taking into account the blocking of the ball as a successful save criterion, a static goalkeeper by virtue of being at the "right" place can prevent some balls from being scored but this happens very rarely especially if the ball speed is low.

B. Weighted Sum

For comparison purposes, the result of the evolutionary run for the multi-objective weighted sum algorithm is included. This run uses equal weights of 0.2 for each of the five objective fitnesses. This algorithm, in contrast to the Pareto-based algorithms like NSGA-II and SPEA2, produces a single optimal individual at the end of the run. The best-of-run individual was evolved at generation 37 and has a standardised fitness of 0.7058.

The mean standardised fitness of the population is higher for the evolutionary run involving an attacker with KP50 when compared to the evolutionary run involving KP100. This indicates that more individuals are fitter when KP50 is used. This also indicates that the problem of evolving a controller that is able to cope with KP50 seems to be relatively easier. In terms of the best-of-run individual's fitness, in both of the runs, their fitness are similar and the final climb before the run is terminated, occurs around the midpoint between generation 30 and generation 40.

When inspecting fitness function f_1 for all of the individuals in the final generation, it can be discovered that no controller was evolved that was able to catch a ball. Even the best-of-run individual do not have this capability. The increase in the overall population fitness is due to fitness f_4 and f_5, which means that a proportion of the individuals approaches the ball and the reverse is true for individuals that are stationary. This consequence is due to the difficulties of recognising whether the overall fitness of an individual is attributed to which fitness measure. This is a common problem with the use of weighted sum methods. One way to alleviate this problem is to give a higher weight value to specific fitness measures as a way of giving more importance to a particular type of controller to be evolved.

C. NSGA-II

For the NSGA-II runs, the results show that the performance for all of the individuals in the final generation are about average where none of the individuals' performances went beyond the 20% mark. The high number of ball-blocking events depicted in the chart show that the individuals performances rely as much on blocking as they do on catching the balls. As a proportion of each individual's performances, there are no individuals that have ball-catching as a dominant portion of their respective ball saves. In fact, some of the individuals have hardly any ball-catching success and rely entirely on blocking as a contributing factor to their performances. There are two

reasons for this outcome. The first reason is that these individuals do not have or do not use any *catch* action function and without this particular command, there will be no ball-catching event. Mostly, these controllers used other action functions such as the *dash*, *kick* and *turn* action functions. The analysis of these individuals by recording their nodes activation count shows that the majority of this type of individuals used the *dash* action function. Very rarely do they use both the *kick* and *turn* action functions. The prevalence of the *dash* action function is due to fitness functions f_4 (movements) and f_5 (positioning).

The results show that if the attacker's kicking power is reduced by half (KP50), it is noted that the performances for some of the more successful individuals increased almost twice that of the best individual's performance. A marked difference between individuals that use the *catch* action (high performance) and the individuals that use the *dash* action node (low performance) is also observed. Since the result shows that there are less ball-blocking events, these events are no longer contributing to the overall performance of an individual. Thus, there are very low performances for individuals that rely on ball-blocking. Additionally, due to the decrease in speed of the ball, any deflected ball can more easily be reached by the attacker to score at a second try.

Furthermore, the high performance seems to be mostly composed of ball-catching events. It is clear that the ball-catching events totally dominate the blocking events. This result suggests that the individuals evolved using KP50 are more accurate in catching the ball, resulting in far less or even no ball-blocking. Due to the low speed of the kicked ball, ball-catching ability becomes more crucial for a goalkeeper. The reason is that when a goalkeeper fails to catch a ball, the attacker has a higher chance of success when attacking the deflected ball. An attacker will not need to travel far to reach a slow moving deflected ball for a second scoring opportunity. A failure in catching a ball will almost inevitably result in a goal whereby the ball moves into the goal in its original trajectory or be deflected and scored by the attacker in its second attempt.

Looking at both set of results, the conclusion is that the ball speed affects the performance of a goalkeeper. Higher ball speeds results in ball-blocking becoming equally important as ball-catching in preventing a score. At slower ball speed, the ability of the individuals in catching the ball becomes more accurate and crucial as the blocking events do not contribute to preventing a score. In terms of performance, the best individual in KP50 (42%) is more than twice the performance of the best individual in KP100 (16.6%) (see Table III and Table II respectively).

D. SPEA2

Similar to the previous section, post evolution runs were performed for the SPEA2 final populations from the evolutionary runs involving KP100 and KP50. The run involving KP100 shows that the individuals have a larger difference between the high performing individuals compared to the low performing individuals. This is in contrast to the result of the NSGA-II KP100 experiment. The results also show that although ball-blocking events exist, the higher performing individuals has a higher proportion of ball-catching events compared to ball-blocking events. For these successful individuals, the ball-catching events contributes the most to their high performance.

The lower performing individuals have no ball-catching events at all. In addition, in terms of ball-blocking events, the out-of-field ball-blocking events are higher than the in-field ball-blocking events.[3] This suggest that for the higher performing individuals, ball-blocking events resulted from catch failure and subsequent collision and deflection of the ball. It also suggest that the deflected ball is more likely to be deflected out of the soccer field. From Table II, the highest performing individual from this run is also the best performing goalkeeper when compared to all of the goalkeepers involved in the KP100 offline experiments.

The results of the SPEA2 offline KP50 experiment is interesting for the fact that the individuals' ball-catching performance did not improve as much when KP50 is used. From the NSGA-II results, it is shown that slower ball speed encouraged evolution of controllers that are able to catch the ball more accurately. However, the SPEA2 results show that, the performance increase in ball-catching ability does not show much improvements. What has grown in importance is the ball-blocking occurrences. Furthermore, most of the ball-blocking events resulted in the ball being deflected within rather than out of the soccer field. Significantly, this makes the ball-blocking events equally important to the ball-catching events for the performances of the individuals. Some of the individuals' performances are due completely to ball-blocking events. For some of the individuals with low ball-catching performances, the remainder of its performance proportions are made up of ball-blocking events.

The result shows that the performance improvements of the individuals in terms of ball-catching are slight but the improvements in ball-blocking has improved much more than expected. This is due to the type of controller that was evolved to position itself during the trial in such a way that the angle of deflection does not take the ball out of the field of play. Furthermore, the angle of deflection is such that the distance that the attacker has to move to reach the ball is far enough for the play time to expire. Therefore, the attacker does not have enough time for a second attempt at scoring even though the ball speed has been reduced by half. It is also apparent that the performance level of all the individuals are almost equally comparable due to the population being populated by very similar type of controllers.

VI. Discussion

The best evolved goalkeeper from the NSGA-II evolution is a controller that attempts to capture the ball more accurately. In

[3]An out-of-field ball-blocking event denotes balls that ended up outside the field of play after being deflected in a collision. Whereas an in-field ball-blocking event denotes balls that remained in the field of play after being deflected and failed to be scored by the attacker before the trial period expires.

contrast, the best evolved goalkeeper from the SPEA2 evolution makes use of the fact that there is a time limit involved in the trials and the controller placed equal importance to both catching and blocking of the ball. The time limit imposed places a limit on the duration for the attacker to attempt a second scoring. The optimal goalkeeper is a goalkeeper that can deny scoring for all of the attacker's attempt. This does not preclude blocking of the ball, however a caught ball is better than a loose ball out in the soccer field that allows the attacker a second attempt.

TABLE IV
GOALKEEPER'S COMPARATIVE PERFORMANCE FOR BALL-CATCHING USING ATTACKER WITH KP100 AND KP50

Goalkeeper Type	KP100	KP50	Difference
Static	0%	0%	0.0
Random	1.2%	8.4%	7.2
Reactive	19%	38.6%	19.6
Predictive	16.4%	55.2%	38.8
Weighted Sum	0%	0%	0.0
NSGA-II	9.8%	41%	31.2
SPEA2	27%	40.2%	13.2

The result shown in Table IV is calculated based on narrowing down the requirement of a successful ball-saving to ball-catching only (ball-blocking is excluded). The data shows that using KP100, the best performing goalkeeper is the SPEA2 evolved goalkeeper achieving a 27% success rate. However, when the ball speed is slowed down by using KP50, the best performing goalkeeper is the predictive goalkeeper. The reason is because at a slower speed the predictive goalkeeper is able to more accurately predict the location of the ball. This is important because a reactive goalkeeper has to adjust its internal information about objects in its vicinity continuously in order to keep track of the location of the ball.

The difference column in Table IV shows the improvement rate of the goalkeeper for its ball-catching ability. The highest increase belongs to the predictive goalkeeper followed by the NSGA-II evolved goalkeeper. The NSGA-II evolved goalkeeper's higher increase rate shows that the evolved goalkeeper places more importance on the accuracy of ball-catching. The SPEA2 evolved goalkeeper has lower rate of increment because the controller favours both ball-catching and ball-blocking. The random goalkeeper's difference is mostly due to the slower moving ball having higher chances of colliding with the goalkeeper thereby being deflected away from the goal. The static and the weighted sum best goalkeeper shows no improvements at all. The weighted sum best goalkeeper did not perform any catch therefore in this respect it is no better than the static goalkeeper. It's performance was mostly due to ball-blocking. Failure in evolving a ball-catching capable controller and depending solely on ball-blocking results in low performance for the weighted sum controller.

Although fitness function f_4 is responsible for rewarding the movements of a goalkeeper, the results show that some of the evolved controllers still forgo movements. This is interesting when compared to the hand-coded goalkeeper design where the goalkeeper approaches and catches the ball. The majority of the successful evolved controllers are either static or with only small movements when executed. This is in contrast to both of the hand-coded goalkeepers where they actively seek out the ball, moving towards it and attempting to catch it. The evolved goalkeepers prefer to stay put or around their original location and wait for the ball to approach whilst attempting to catch it.

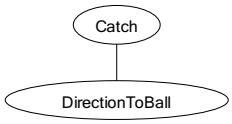

Fig. 2. A tree representation of the best individual in SPEA2 (KP100)

In the case of the SPEA2 offline runs with KP100, 26% of the individuals in the final population has the type of controller shown in Figure 2. Even though this controller looks simple, it is not at all obvious that it is a forgone conclusion that this type of controller should be evolved. For example, the weighted sum evolution did not produce this type of controller. The best NSGA-II controller in this category evolved the tree in Figure 3. In addition, there are many more terminals apart than this particular *DirectionToBall* terminal that might serve as the input to the *catch* function node. Furthermore, if the goalkeeper is not able to localise, the *DirectionToBall* terminal will not yield any useful value.

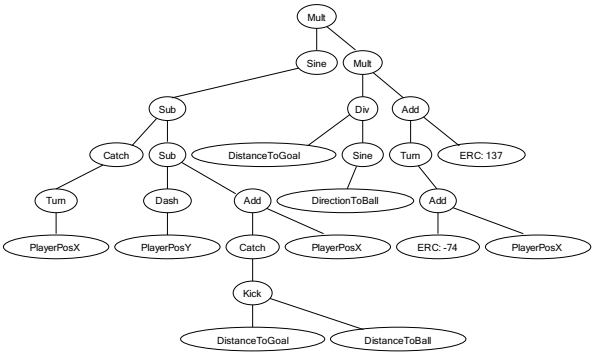

Fig. 3. A tree representation of the best individual in NSGA-II (KP100)

The best controller in this category is the SPEA2 evolved controller and it is this type of controller. It managed to achieve a performance level of 32.4%. This controller outperformed the reactive and predictive hand-coded controllers and also the best NSGA-II controller in terms of both ball-catching and ball-blocking abilities (see Table II).

There are 4 types of action functions included in the function set for the GP runs. These are the *catch*, *dash*, *kick* and *turn* functions. Summarising the number of times that this function set is used by the individuals in the final generations for both the NSGA-II and SPEA2 runs, gives the results in Table V. The results show that on average, the individuals from both the NSGA-II and SPEA2 evolved controllers favours

the execution of the *catch* and *dash* over the *kick* and *turn* action functions. With the exception of the SPEA2 (KP50) final population, the rest of the final populations executed the *dash* more than the *catch* action function. This establishes that the overall strategy used seems to be either a *dash* or *catch* on its own or a combination of *dash* and *catch* together. It can be inferred that the *dash* and *catch* action functions are deemed important by the GP algorithm for evolving good goalkeeping behaviours.

TABLE V
TOTAL NUMBER OF ACTION NODES USED BY CONTROLLERS IN THE PARETO FRONTS OF NSGA-II AND SPEA2

Algorithm	Catch	Dash	Kick	Turn	Total
NSGA-II (KP100)	30	67	1	1	99
NSGA-II (KP50)	39	59	0	0	98
SPEA2 (KP100)	37	63	0	0	100
SPEA2 (KP50)	53	46	1	0	100
Total	159	235	2	0	

Since the way the trial was designed was for an attacker from various positions along a circular diameter from the goalkeeper to kick the ball. The ball's trajectories will mostly crisscross in front of the goalkeeper before arriving in the goal.[4] The GP algorithm has exploited this situation and thus the evolved controllers that stays near to the location where the crisscrossing happens the most whilst executing *catch(ball.direction)* means that if the passing ball is within the catch area, the goalkeeper can catch the ball without even turning around to have the ball directly in its view. The high number of ball-blocking resulting in a ball-save event has also ensured that this type of controllers is kept in the population.

VII. CONCLUSION

This work has extended the canonical GP implementation to accommodate Pareto-based multi-objective optimisation and implemented two relatively recent algorithms namely NSGA-II and SPEA2 respectively. Both of these algorithms outperformed the weighted sum GP implementation. In the case of the best controller evolved using SPEA2, the results from the post evolution runs show that it outperformed both the simple reactive and predictive hand-coded controller. The analysis shows that the SPEA2 evolved controllers exploited the ability to block the ball in addition to catching the ball. Therefore, the results show that the GP algorithm has exploited some features of the experiment setting to evolve alternative controller designs in comparison to the fixed hand-coded controllers. If only ball-catching events are used to denote successful saves, i.e. ball-blocking events are excluded, the best performance for the evolved goalkeepers belongs to the NSGA-II evolved controller.

The results described are encouraging as they show that the approach of using Pareto-based MOGP is successful in evolving better performing goalkeeping behaviours. The best controller performed better than the simple hand-coded and weighted sum evolved controllers in preventing goals from being scored. Furthermore, different types of controllers were evolved exhibiting different goalkeeping behaviours.

[4]This is only true for some cases of the attacker's positions (see Figure 1)

REFERENCES

[1] A. Messac, G. J. Sundararaj, R. Tappeta, and J. E. Renaud, "Ability of Objective Functions to Generate Points on Non-Convex Pareto Frontiers," *AIAA Journal*, vol. 38, no. 6, pp. 1084–1091, June 2000.

[2] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*. Springer, 2000, pp. 839–848.

[3] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective 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, Tech. Rep., 1999.

[4] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[5] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*. New York: John Wiley & Sons, Inc., 1986.

[6] A. D. Belegundu and T. R. Chandrupatla, *Optimization Concepts And Applications In Engineering*. New Jersey: Prentice Hall, 1999.

[7] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research and Management Science. Dordrecht: Kluwer Academic Publishers, 1999, vol. 12.

[8] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1985, pp. 93–100.

[9] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Genetic Algorithms: Proceedings of the Fifth International Conference*. Morgan Kaufmann, 1993, pp. 416–423.

[10] N. Srinivas and K. Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[11] J. Horn and N. Nafpliotis, "Multiobjective Optimization using the Niched Pareto Genetic Algorithm," Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, Tech. Rep. IlliGAl Report 93005, 1993.

[12] J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation," in *Proceedings of the Congress on Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 1. Mayflower Hotel, Washington D.C., USA: IEEE Press, 6-9 1999, pp. 98–105.

[13] C. A. Coello Coello, "A Short Tutorial on Evolutionary Multiobjective Optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, Eds. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001, pp. 21–40.

[14] A. Goicoechea, D. R. Hansen, and L. Duckstein, *Multiobjective Decision Analysis with Engineering and Business Applications*. John Wiley & Sons, 1982.

[15] P. J. Fleming and A. P. Pashkevich, "Computer Aided Control System Design Using a Multiobjective Optimization Approach," in *Proceedings of the lEE Control '85 Conference*, 1985, pp. 174–179.

[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, Tech. Rep. 103, 2001.

[17] R. de Boer and J. R. Kok, "The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team," Master's Thesis, University of Amsterdam, The Netherlands, February 2002.

[18] Holger Endert and Robert Wetzker and Thomas Karbe and Axel Heßler and Philippe Büuttner and Felix Brossmann, "The DAInamite Agent Framework," Website, November 2006, http://www.dainamite.de/.