2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

# Competitive Coevolutionary Training of Simple Soccer Agents from Zero Knowledge

Christiaan Scheepers
Department of Computer Science
School of Information Technology
University of Pretoria
Pretoria 0002
South Africa
Email: cscheepers@acm.org

Andries P. Engelbrecht
Department of Computer Science
School of Information Technology
University of Pretoria
Pretoria 0002
South Africa
Email: engel@cs.up.ac.za

***Abstract*—A new competitive coevolutionary team-based particle swarm optimisation (CCPSO) algorithm is developed to train multi-agent teams from zero knowledge. The CCPSO algorithm uses the charged particle swarm optimiser to train neural network controllers for simple soccer agents. The training performance of the CCPSO algorithm is analysed. The analysis identifies a critical weakness of the CCPSO algorithm in the form of outliers in the measured performance of the trained players. A hypothesis is presented that explains the presence of the outliers, followed by a detailed discussion of various biased and unbiased relative fitness functions. A new relative fitness function based on FIFA's league ranking system is presented. The performance of the unbiased relative fitness functions is evaluated and discussed. The final results show that the FIFA league ranking relative fitness function outperforms the other unbiased relative fitness functions, leading to consistent training results.**

## I. Introduction

The particle swarm optimisation (PSO) algorithm [1] is a stochastic population-based optimisation method, with its roots in the simulation of the social behaviour of birds within a flock. First developed by Kennedy and Eberhart [1] in 1995, the PSO algorithm has been more successful in solving complex problems than traditional evolutionary computation (EC) algorithms on a variety of problems [2]. Particle swarm optimisers have proved successful in training board state evaluators for games such as Tic-Tac-Toe, Checkers and Bao [3]–[7]. The aforementioned training techniques require the construction of traditional game trees, and using a competitive coevolutionary PSO algorithm to train a neural network game state evaluator. Constructing traditional game trees can become impractical for games that are more complex, such as Go [8]. The same applies to games such as simulated soccer where it may not even be possible to always construct a game tree. In contrast to the above mentioned techniques, the technique presented in this paper does not make use of a game tree. Instead, the actions taken by each player is directly controlled by a neural network.

The competitive coevolutionary team-based particle swarm optimisation (CCPSO) algorithm presented in this paper is applied to train soccer-playing robot teams in a simplified soccer game. Teams compete against one another while team-members collaborate in order to win. In addition to training a team of players, the training is performed from zero knowledge, that is, no domain information is provided to the training algorithm; only the game outcome and rules of the game are known during training. As no domain information is required, the algorithm can easily be adapted for games other than simple soccer.

Analysis of the training performance showed the presence of outliers in the measured performance of the trained players. A hypothesis is presented to explain the presence of the outliers. A review of various alternative relative fitness functions is presented. The FIFA league ranking relative fitness function is then proposed. Final results confirm the above hypothesis and show that the FIFA league ranking relative fitness function overcomes the outlier problem.

The remainder of this paper is organised as follows: Section II presents the simple soccer model. Section III presents the CCPSO algorithm. Training performance is discussed in section IV. Section V provides an overview of various relative fitness functions and introduces the FIFA league ranking relative fitness function. It is also shown that the FIFA league ranking relative fitness function outperforms the alternative relative fitness functions. Finally, section VI presents the findings and conclusions of this paper.

## II. Simple Soccer

This section presents the simple soccer model used throughout this paper. Simple soccer is loosely based on the simulated soccer model introduced by Jeong and Lee [9]. Detail on the simple soccer field, team composition, agent sensors, agent actions, and overall game rules are presented.

### A. Soccer field

Games are played on a two dimensional virtual soccer field of size $l \times w$. Fig. 1 depicts the soccer field as it is used in this study. The size of the field is 5 wide by 6 long. Two additional blocks that represent the goal areas are added to the field on opposite sides.

### B. Team composition

Two teams compete against each other. Each team consists of two agents. Agents $A_1$ and $A_2$ form team $A$ while agents $B_1$ and $B_2$ form team $B$.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

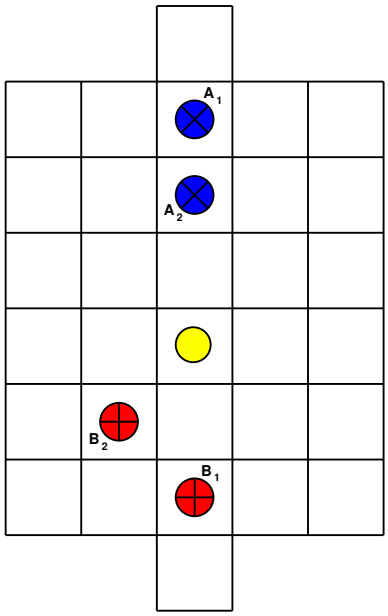


Fig. 1. $5 \times 6$ Simple Soccer field with the ball and players.

### C. Agent sensors

Fehérvári and Elmenreich demonstrated the importance of well-designed sensors to serve as neural network inputs when attempting to train neuro-controllers for a soccer-like environment [10]. Fehérvári and Elmenreich proposed the use of four neurons to detect each class of object. Classes of objects are the ball, nearest team-mate, nearest opponent, and wall. The four neurons represent the distance to the object in a specific direction, i.e. north, south, east or west. The input functions for the four neurons are defined as follows:

$$f(north) = \begin{cases} \frac{\Delta y}{d^2} & \text{if } y > 0 \\ 0 & \text{if } y \leq 0 \end{cases} \quad (1)$$

$$f(south) = \begin{cases} \frac{\Delta y}{d^2} & \text{if } y < 0 \\ 0 & \text{if } y \geq 0 \end{cases} \quad (2)$$

$$f(west) = \begin{cases} \frac{\Delta x}{d^2} & \text{if } x < 0 \\ 0 & \text{if } x \geq 0 \end{cases} \quad (3)$$

$$f(east) = \begin{cases} \frac{\Delta x}{d^2} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (4)$$

where $\Delta x$ is the distance along the east-west direction and $\Delta y$ is the distance along the north-south direction; $d$ is defined as $d = \sqrt{\Delta x^2 + \Delta y^2}$.

### D. Agent actions

Each agent can perform one of several actions per turn:

- **Move**: An agent can move one square to either of the eight adjacent squares.
- **Dribble**: An agent that has possession of the ball can move one square to either of the eight adjacent squares, taking the ball along with it.
- **Kick**: An agent that has possession of the ball can kick the ball two squares away in either the left or right, or forward or backward directions.

Fig. 2 depicts the actions that an agent can perform, along with their directions. In the event that an agent executes an invalid action, for instance kicking without having possession of the ball, that action is simply ignored.

### E. Rules

The rules that govern how each game is played are:

- For each team one agent starts the game in front of the goal area, the second agent starts in either of the three blocks in front of the first agent. The exact placement of the second agent is determined using a uniform random placement function.
- The ball starts in the middle of the field. If the middle is between two or four blocks, the placement is done uniform randomly within either of the blocks.
- The game stops once a goal is scored, or if a specified maximum number of iterations is reached.
- In the event that multiple agents occupy the same block as the ball, the ball ownership is determined using the probability $\frac{1}{n}$, where $n$ is the number of agents occupying the block.
- A goal score is awarded once the ball enters the goal area on either side.
- If an agent is in possession of the ball and the move action is executed, the agent will automatically perform the dribble action.
- An agent cannot leave the field with either a move or dribble action.
- The ball cannot be kicked off the field. If a kick is performed that would result in the ball leaving the field, the ball is simply stopped at the field's boundary.

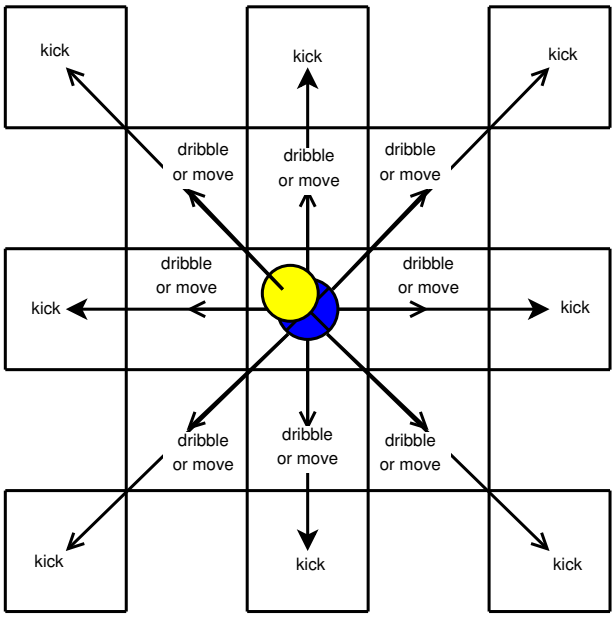


Fig. 2. Simple soccer agent actions.

## III. Cooperative Competitive Coevolution with Charged PSO

This section introduces the CCPSO algorithm for training multi-agent teams from zero knowledge. The neural network architecture, as used by the neuro-controlled players, is also discussed.

Each game agent, or neuro-controlled player, is represented in the CCPSO algorithm as a separate swarm of particles. Each particle position represents a weight vector for the neural network of the corresponding player. The training objective for the CCPSO algorithm is to find the best performing particle positions for each of the swarms. These best performing particle positions represent the best performing players for each of the corresponding player positions.

Scheepers showed that the competitive coevolutionary training environment can be seen as a dynamic environment [11]. The CCPSO algorithm therefor uses the charged PSO introduced by Blackwell and Bentley [12]–[14] to improve performance in the presence of a constantly changing search space.

Fig. 3 provides a pseudocode implementation of the CCPSO algorithm. A hall of fame (HOF) [15], [16] is also maintained to avoid early stagnation and to aid exploration. The HOF maintains a collection of the best performing individuals from each generation since the evolutionary process has started. This collection of previous best performing individuals helps to combat the loss of exploration by preserving past good characteristics.

```
Create and initialise a swarm of neural networks for each game position.
repeat
    for all swarms O(t) do
        for all swarms O_s(t) ≠ O(t) do
            Add each personal best position to the competition pool for swarm O_s(t).
            Add each particle to the competition pool for swarm O_s(t).
        end for
        for all particles P_i (or NN) in the swarm O(t) do
            repeat
                Select team members and opponents from the competition pools.
                Play a game using the selected players.
                Record if game was won, lost or drawn.
            until predefined number of games has been played.
        end for
        Determine a score for each particle.
    end for
    Compute new neighbourhood best position.
    Update particle velocities.
    Update particle positions.
until all swarms converge or iteration limit is reached.
```

Fig. 3. Pseudocode for the basic CCPSO algorithm.

The experimental work in this study made use of the basic feed-forward neural network algorithm as neuro-controller for the simple soccer agents. A single hidden layer with five neurons was used. The hyperbolic tangent activation function was used as activation function. The neural network input consisted of a 16-dimensional floating-point vectors. The values for the input vector is calculated using equations (1), (2), (3), and (4). The input vector is constructed by combining the four calculated values for each of the four object classes. The four object classes are the ball, the closest team member, the closest opponent, and the field boundary.

The neural network output consists of an eight-dimensional floating-point vector with values scaled between 0 and 1. The vector is constructed by combining the movement and kick vectors. The movement vector indicates the movement direction, and the kick vector indicates the direction to kick the ball. The four values per vector are desirability factors to move or kick in each of the primary directions: forward, backward, left, and right. A desirability factor larger than $0.5$ indicates a desire to move or kick in the corresponding direction. If desirability is indicated in two conflicting directions (i.e. both left and right), the direction with the larger desirability factor is given precedence. If two conflicting desirability factors are equal, no desire is indicated and no action will be taken. For example, the vector $\{0.6, 0.8, 0.2, 0.4, 0.3, 0.4, 0.1, 0.2\}$ represents a desire to move backward.

For the experimental work presented in this paper the parameters of the CCPSO algorithm were optimised using a parallel coordinate virtualisation technique [17]. Table I provides a summary of the parameter values that was used.

TABLE I. CCPSO parameters

| Parameter | Value |
|---|---|
| Swarm type | Atomic (50%) |
| Neighbourhood structure | Von Neumann |
| Inertia weight | 0.729844 |
| Social constricting coefficient | 1.49618 |
| Cognitive constricting coefficient | 1.49618 |
| Initial particle velocity | 0 |
| Initial particle positions | $R(-1, 1)$ |
| Swarm size | 20 particles |
| Perception limit | 500.0 |
| Perception core | 2.0 |
| Charge magnitude | 15.0 |
| Maximum particle velocity | 15.0 |
| Hall of fame size | 4 |
| Hall of fame update iterations | 30 |
| Competition pool size | 15 |
| Personal best re-evaluation iterations | 3 |

## IV. Outlier Problem

This section discusses the performance of the CCPSO algorithm. Performance is measured by playing games against randomly behaving players, using the *S* measure defined as

$$S = \frac{w_{won} n_{won} + w_{draw} n_{draw}}{n_{total}} \tag{5}$$

where $w_{won}$ and $w_{draw}$ is the weight assigned to games won and drawn respectively. The *S* measure is calculated over 15000 games with $w_{won}$ and $w_{draw}$ set to 1.0 and 0.5 respectively. The average *S* measure along with the standard deviation is reported over 30 simulations.

Fig. 4 depicts the $S$ measure for 30 simulations over 2000 iterations. A quick visual inspection of the $S$ measure values at iteration 500 reveals that the majority of $S$ measure values are clustered above 0.6. Two bad performing teams, or outliers

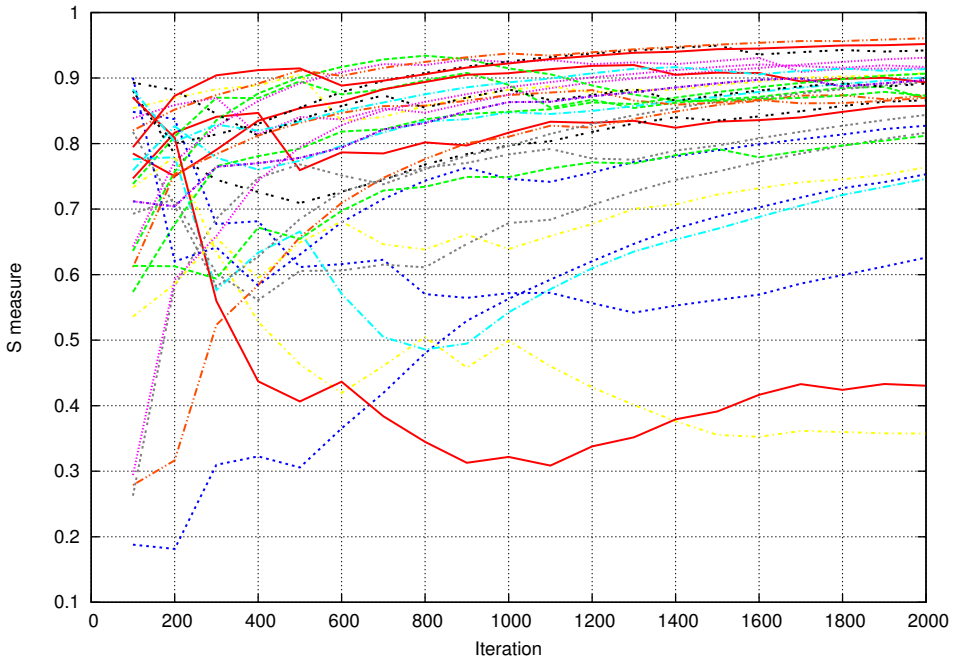


Fig. 4. $S$ measures over 2 000 iterations for 30 teams.

in the measured performance can be seen at iteration 500 with $S$ measure values of 0.3 and 0.4 respectively. The available information shows that the evolutionary process is not always able to drive the performance of the swarm high enough. Outliers remain visible in the data even with an iteration limit of 2000.

The outliers indicate that the training algorithm failed to consistently train well-performing simple soccer players using a given parameter configuration. One interpretation of these results might be that the relative fitness function does not reveal enough information to drive the evolutionary process fast enough to a better-performing area of the hyper-dimensional search space. Further analysis of the relative fitness function and the effect thereof on the training performance is required to confirm or deny this hypothesis. A detailed analysis of the relative fitness function is presented in the next section.

## V. Relative Fitness

This section provides an overview of alternative relative fitness functions. The concept of biasing behaviour towards a specific gameplay strategy through a fitness function is discussed. A performance comparison between the alternative unbiased relative fitness functions is presented.

### A. Avoiding Biased Behaviour

One of the primary focuses of this study is to develop a training technique that trains playing strategies from zero knowledge. Zero knowledge implies that the behaviour of the players may not be predetermined or guided towards a specific playing style or strategy. It is of key importance that the relative fitness function therefor does not introduce a bias towards specific strategies by rewarding a certain gameplay style more than others.

The next two subsections discuss two alternative relative fitness functions to train game agents based on previous studies. Both of these relative fitness functions introduce a bias by rewarding certain gameplay styles more than others.

*1) Simple soccer absolute fitness:* Lee and Jeong developed an absolute fitness function for their genetic algorithm (GA) to train simple soccer game agents [9]. The simple soccer absolute fitness function to be maximised is defined as

$$\mathcal{F} = \sum_{i=1}^{i=N} L'(i) \tag{6}$$

where

$$L'(i) = L_0 + \sum_{t=0}^{t_{final}} (f_{goal-A}(t) + f_{posses-A}(t) + f_{goal-B}(t) + f_{posses-B}(t)) \tag{7}$$

with

$$f_{goal-A}(t) = \begin{cases} 1000 & \text{if team } A \text{ scores at iteration } t \\ 0 & \textit{otherwise} \end{cases}$$
$$f_{goal-B}(t) = \begin{cases} -90 & \text{if team } B \text{ scores at iteration } t \\ 0 & \textit{otherwise} \end{cases}$$
$$f_{possess-A}(t) = \begin{cases} 10 & \text{if team } A \text{ possesses the ball at iteration } t \\ 0 & \textit{otherwise} \end{cases}$$
$$f_{possess-B}(t) = \begin{cases} -10 & \text{if team } B \text{ possesses the ball at iteration } t \\ 0 & \textit{otherwise} \end{cases} \tag{8}$$

where $i$ is the $i$'th simulation of the game, $N$ is the number of simulations (Lee and Jeong defined $N$ to be five for their work), $t_{final}$ is the number of iterations per simulation until a goal is scored or 30 if no goal is scored in the first 30 iterations,

$t$ is the $t$'th iteration of the simulation, and $L_0$ is a constant set to 400.

Analysis of equation (7) reveals that the maximum value for $\mathcal{F}$ is achieved when $f_{possess-A}(t) = 10$ for $t < 30$ and $f_{goal-A}(t) = 1000$ for $t = 30$. This maximum fitness value can only be achieved by holding on to the ball for all iterations up to the last, at which point the team must score a goal. Because $\mathcal{F}$ is maximised the optimisation process will likely result in players exhibiting this specific behaviour. Thus introducing a bias towards this specific gameplay strategy.

This bias violates the *training from zero-knowledge* goal of this study.

*2) Rampup absolute fitness:* Fehérvári and Elmenreich used a genetic algorithm (GA) to evolve neural network controllers to play soccer in a simplified soccer simulation [10]. They made use of a fitness function, hereafter referred to as the *rampup absolute fitness function*, to drive the evolutionary process. The rampup absolute fitness function was defined as follows:

$$F_{ramp} = W_p P_p + W_{bd} P_{bd} + (W_k P_k - W_{fk} P_{fk}) + W_{bg} P_{bg} + W_s P_s \quad (9)$$

with the parameters as summarised in table II, where $P_p$, $P_{bd}$, $P_k$, $P_{fk}$, $P_{bg}$ and $P_s$ represent the number of points and $W_p$, $W_{bd}$, $W_k$, $W_{fk}$, $W_{bg}$ and $W_s$ are the weighting of these points.

Points are awarded every five seconds for field distribution, every four seconds for distance to the ball and every two seconds for ball distance to the opponent's goal. Field distribution is based on 64 evenly distributed checkpoints. Each checkpoint is controlled by the nearest player. A point is awarded for each checkpoint controlled by a team. A single point is awarded to the team with the player nearest to the ball. Kicking-related points are awarded a point per kick. Fig. 5 presents a visual representation of how the evolutionary process is guided towards specific behaviour.

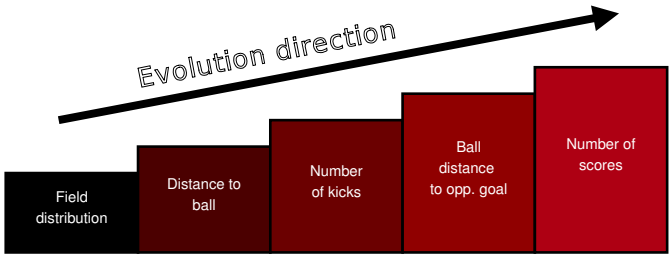


Fig. 5. Rampup absolute fitness direction of evolution.

It is clear to see how the *rampup absolute fitness function* drives the evolutionary process towards a specific bias. Inspection of the reward function reveals that only offensive-type strategies are rewarded. Given enough generations, a defensive playing team will not be rewarded enough to survive. The objective of the soccer game is indeed to score goals; however, drawing behaviour is also worth rewarding. Drawing a game against a very good team can be compared to winning against a mediocre team.

TABLE II. RAMPUP ABSOLUTE FITNESS FUNCTION PARAMETERS.

| Parameter | Symbol | Weight |
|---|---|---|
| Field distribution | $P_p$ | $10^0$ |
| Distance to the ball | $P_{bd}$ | $10^3$ |
| Number of kicks | $P_k$ | $2 \times 10^4$ |
| Number of false kicks (ball is kicked out of bounds) | $P_{fk}$ | $10^4$ |
| Ball distance to the opponent's goal | $P_{bg}$ | $10^5$ |
| Number of scores | $P_s$ | $4 \times 10^6$ |

### B. Unbiased Fitness

The previous subsection showed that it is possible to introduce bias into the training process through the relative fitness function. Developing a well-performing unbiased relative fitness function is extremely important. Two unbiased relative fitness functions were developed for the simple soccer problem by the authors of this study. The two relative fitness functions are discussed below.

*1) Fixed Reward:* The reward scheme used in the initial experimentation rewarded victories with one point, draws with zero points and defeats with $-2$ points [18]–[20]. In contrast to this point structure international soccer leagues typically use a point system where three points are awarded for a victory, one point for a draw, and zero for a defeat.

Updating the point allocation to be inline with the soccer league scoring system will not address the primary weakness of the fixed reward scheme. Draws will become a less desirable outcome, but no additional information about the actual game performance will be revealed. A very good team winning $15:0$ against a very bad team will be rewarded the same as a team winning $9:6$ against a very good team. This leads to a low diversity between the relative fitness value of particles in a swarm in cases where a small number of games are played. Higher diversity in the relative fitness function reveals more information about the search space, which in turn, allows for faster convergence on well-performing areas of the search space. A relative fitness function that reveals more information about the search space is thus more desirable.

*2) Goal Difference:* In order to increase population diversity, more information must be gathered from the competitions being played. One approach is to award points based on the score difference (and not just based on the competition outcome) between competing two teams. For the simple soccer model, the score difference is simply the goal difference at the end of the game. A team winning 15 to zero will be rewarded with 15 points for their victory, whereas the losing team will receive a penalty of $-15$ points for their defeat. A draw will result in both teams receiving zero points. The relative fitness, $\mathcal{F}(t)$, for iteration $t$ is calculated using

$$\mathcal{F}(t) = \sum_{n=1}^{N} (\mathcal{G}_n - \mathcal{O}_n) \quad (10)$$

where $\mathcal{G}_n$ is the number of goals scored in the $n$'th competition, $\mathcal{O}_n$ is the number of goals the opponents scored in the $n$'th competition, and $N$ is the number of games played per tournament.

The goal difference relative fitness function gathers more information from each game being played when compared to the fixed reward relative fitness function. It is, however, more influenced by the game rules than a fixed reward scheme. For instance, if the game is played only up to five goals it does

not take the number of game steps (or iterations) into account for a team to score the five goals. One team might score five goals very fast, whereas another team might take a very long time to score five goals. Limiting the game in question to a fixed number of steps (or iterations) addresses this problem to some extent. The fast scoring team will still be able to score five goals, whereas the slow scoring team might only score, say, two goals. This approach may not always be an option for all games as certain games dictate game-end conditions as part of the game rules and iteration limits cannot be applied. For simple soccer, an iteration limit can be added.

### C. FIFA League Ranking

An alternative approach to increasing the information gain of the relative fitness function is to make use of the Official FIFA World Ranking points system.[1] After the 2006 FIFA World Cup, a revised ranking calculation procedure was introduced to significantly simplify the procedure. The new ranking system makes use of the following formula to calculate the points awarded to a team per match:

$$P = M \times I \times T \times C \times 100 \tag{11}$$

where

- M indicates if the match ended in a victory (3 points), a draw (1 point), or a defeat (0 points). In cases where a penalty shoot-out is required, the winning team is awarded 2 points and the losing team 1 point. Should a team lose a match, no points are awarded.
- I indicates the importance of the match based on a weight scheme. For friendly matches $I = 1.0$, for World Cup qualifier and continental qualifier matches $I = 2.5$, for continental final competitions and FIFA confederation cup matches $I = 3.0$, and for World Cup final competitions $I = 4.0$.
- T indicates the strength of the opposition, calculated as

  $$T = \frac{200 - (\text{opposition rank position})}{100} \tag{12}$$

  The top team is assigned $T = 2.0$ and teams ranked lower than 150 is assigned $T = 0.5$. Rank is based on the most recent FIFA ranking publication.
- C indicates the strength of the confederation. For intercontinental matches, the average of the confederations the two competing teams belong to, is used. The confederation strength is based on the number of victories by the confederation at the last three FIFA World Cup competitions. After the 2010 FIFA World Cup the strengths were *UEFA* 1.00, *CONMEBOL* 1.00, *CONCACAF* 0.88, *AFC* 0.85, *CAF* 0.86, and *OFC* 0.85.
- P indicates the points awarded to a team for a match.

Generally speaking, the greatest winners in the new point system are teams who win competitive matches against high-ranking opponents. Friendly matches and draws provide limited gains and losses provide none.

[1] http://www.fifa.com/worldfootball/ranking/procedure/men.html

Rank is calculated for a team as the sum of all the games played in the last four years devalued over the period ($100\%$ for the first year, $50\%$ for the second year, $30\%$ for the third year, and $20\%$ for the fourth year back).

The ranking system introduced by FIFA recognises the fact that a victory against a good team (or a high-ranking team) is more difficult to achieve - scoring goals against a good team is inherently more difficult, and should thus be rewarded more than a victory against a bad team (or a low-ranking team). The ranking system also takes into account past results, albeit at a reduced weight when compared to current results, thus avoiding a situation where a single team can dominate the ranks based on a single year's results.

Competitive coevolution makes use of tournaments to determine a relative fitness for each individual. The relative fitness can be seen as a rank. Similarly, the relative fitness function can be seen as a ranking system. Building on this idea, the experimental work in this study made use of a new relative fitness function based on the FIFA ranking system. Points are calculated using

$$\mathcal{P}(t) = \sum_{n=1}^{N} \left( \mathcal{M}_n \times \frac{200 - \mathcal{T}_n}{100} \right) \tag{13}$$

where $N$ is the number of games played in the competitive coevolution tournament, $\mathcal{M}_n$ is the $n$'th game outcome, defined as

$$\mathcal{M}_n = \begin{cases} 3 & \text{for a victory} \\ 1 & \text{for a draw} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

and $\mathcal{T}_n$ is the $n$'th opposition team rank. Each tournament team consists of randomly chosen team members. The use of randomly constructed teams creates a problem as no team rank can be maintained with consistency between iterations. Instead, the rank must be maintained per player, and then calculated for a team based on the members. The opposition team rank, $\mathcal{T}_n$, is calculated by averaging the rank of the members of the opposition team as follows:

$$\mathcal{T}_n = \frac{\sum_{p=1}^{P} r(p)}{P} \tag{15}$$

where function $r(p)$ represents the rank of player $p$ in a team consisting of $P$ members. A player's rank is determined based on the relative fitness of the particle in the previous iteration. The relative fitness, $\mathcal{F}(t)$, for iteration $t$ is calculated using

$$\begin{aligned} \mathcal{F}(t) = \mathcal{P}(t) &+ 0.5 \times \mathcal{P}(t-1) + 0.3 \times \mathcal{P}(t-2) \\ &+ 0.2 \times \mathcal{P}(t-3) \end{aligned} \tag{16}$$

taking into account previous points awarded using the same weighting scheme as the FIFA ranking system.

Two concepts not incorporated into the newly developed relative fitness function are the addition of a game importance weight ($I$ in the FIFA equation) and confederation strength ($C$ in the FIFA equation). In theory it would be possible to incorporate these two components if the training algorithm made use of multiple populations for a single position. Currently, the algorithm uses a single population per position, rendering these components redundant.

As historic performance is indirectly maintained by the rank, $r(p)$, used when determining the new fitness value (which in turn determines the new rank) the function is further simplified to $\mathcal{F}(t) = \mathcal{P}(t)$.

It should be noted that the FIFA league ranking system does not incorporate any problem domain information other than game outcome similar to the fixed reward functions. This property makes it possible to apply the ranking system as a relative fitness function, as described here, to any competitive coevolutionary problem.

### D. Relative Fitness Function Evaluation

Of the three unbiased relative fitness functions reviewed above, the FIFA league ranking function stands out. FIFA league ranking takes into account past player performance when calculating the relative fitness value. The past performance allows for teams to be ranked over a number of games, allowing teams to be rewarded according to their expected performance. The function also provides a more detailed indication of a players' performance than the other two relative fitness functions discussed. As stated in the previous section, functions that reveal more information about the search space is considered more desirable as they improve training performance.

To objectively compare the performance of the three unbiased relative fitness functions a comparative study was carried out. Each function was tested using 30 simulations over 2000 iterations with the optimised parameter configuration shown in table I.

Fig. 6 provides the performance comparison of the three unbiased relative fitness functions. The average *S* measure and standard deviation for each of the three unbiased relative fitness functions are shown. The FIFA league ranking relative fitness function clearly outperformed the other two relative fitness functions. A higher *S* measure and a lower standard deviation were achieved. The lower standard deviation indicates a higher level of consistency in the algorithms' ability to produce well performing players. Fig. 7 depicts the $S$ measure for 30 simulations over 2000 iterations using the FIFA league ranking relative fitness function. The lack of outliers confirm that the FIFA league ranking relative fitness functioned solved the outlier problem.

## VI. Findings and Conclusions

This paper proposed a new competitive coevolutionary team-based particle swarm optimisation (CCPSO) algorithm to train multi-agent teams from zero knowledge. The simple soccer model was introduced as a testbed for the CCPSO algorithm. The rules governing play were also presented. It was shown how the CCPSO algorithm could be applied to the simple soccer model.

The *S* measure was presented to measure training performance of the CCPSO algorithm. Analysis of the training performance of the CCPSO algorithm revealed the presence of outliers in the training data. The outliers indicated the training algorithm failed to consistently train well-performing simple soccer players. It was hypothesised that the relative fitness function could be the root cause of the outliers.

Various design aspects of the relative fitness function were presented and a new function based on FIFA's league ranking algorithm was presented. Analysis of the training performance when using the FIFA league ranking relative fitness function revealed that the outliers were no longer present. The newly introduced relative fitness function solved the outlier problem.

Overall it was shown that the CCPSO algorithm, along with the FIFA league ranking relative fitness function, was capable of training well performing players, from zero knowledge, as measured by the $S$ measure.

## References


[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proc. IEEE International Conference on Neural Networks (ICNN'95)*, vol. 4, Apr. 1995, pp. 1942–1948.

[2] ——, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[3] L. Messerschmidt and A. Engelbrecht, "Learning to Play Games using a PSO-based Competitive Learning Approach," *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 444–448, 2002.

[4] C. Franken and A. Engelbrecht, "Comparing PSO Structures to Learn the Game of Checkers from Zero Knowledge," in *Proc. Congress on Evolutionary Computation (CEC'03)*, 2003, pp. 234–241.

[5] ——, "Evolving intelligent game-playing agents," in *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*. South African Institute for Computer Scientists and Information Technologists, 2003, pp. 102–110.

[6] ——, "PSO Approaches to Coevolve IPD Strategies," in *Proc. Congress on Evolutionary Computation (CEC'04)*, vol. 1, 2004, pp. 356–363.

[7] J. Conradie and A. Engelbrecht, "Training Bao Game-Playing Agents using Coevolutionary Particle Swarm Optimization," in *Proc. IEEE Symposium on Neural Networks (ISSN'06)*, 2006, pp. 67–74.

[8] D. Davis, T. Chalabi, and B. Berbank-Green, "Artificial-life, agents and GO," in *New Frontiers in Computational Intelligence and Its Applications*, M. Mohammadian, Ed. Amsterdam, The Netherlands: IOS Press, 2000, pp. 125–139.

[9] I. Jeong and J. Lee, "Evolving Multi-agents using a Self-organizing Genetic Algorithm," *Applied Mathematics and Computation*, vol. 88, no. 2-3, pp. 293–303, 1997.

[10] I. Fehérvári and W. Elmenreich, "Evolving Neural Network Controllers for a Team of Self-organizing Robots Self-organizing Systems," *Journal of Robotics*, 2010.

[11] C. Scheepers, "Coevolution of Neuro-controllers to Train Multi-Agent Teams from Zero Knowledge," MSc dissertation, University of Pretoria, 2013.

[12] T. Blackwell and P. Bentley, "Dont push me! Collision-avoiding swarms," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1691–1696.

[13] ——, "Dynamic Search with Charged Swarms," in *Proceedings of the 2002 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers, 2002, pp. 19–26.

[14] T. Blackwell, "Swarms in Dynamic Environments," in *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*. Springer-Verlag, 2003, pp. 1–12.

[15] C. Rosin and R. Belew, "Methods for Competitive Co-evolution: Finding Opponents Worth Beating," in *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers, Inc., 1995, pp. 373–380.

[16] ——, "New Methods for Competitive Coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1–29, Jan. 1997.

[17] C. Franken, "Visual Exploration of Algorithm Parameter Space," *Proc. IEEE Congress on Evolutionary Computation (CEC'09)*, pp. 389–398, 2009.

[18] K. Chellapilla and D. Fogel, "Evolving Neural Networks to Play Checkers without Expert Knowledge," *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1382–1391, 1999.

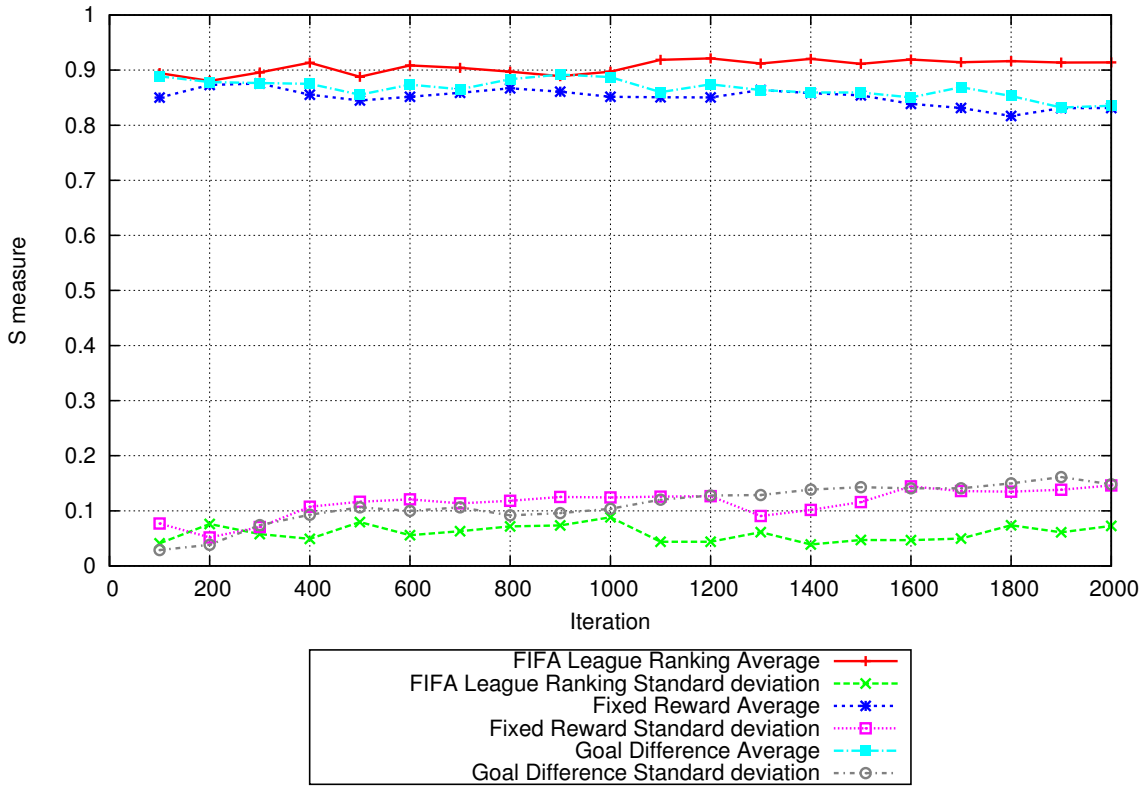


Fig. 6. Relative fitness function comparison.

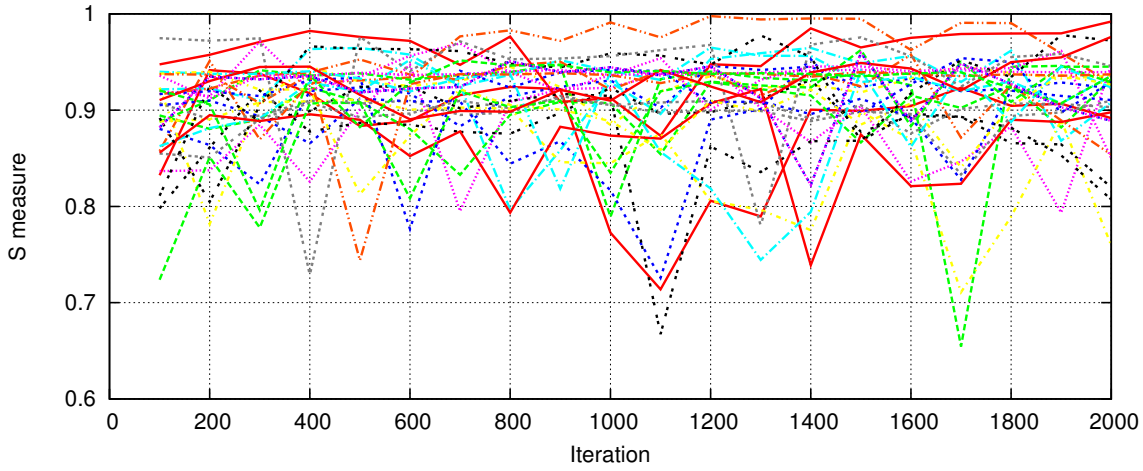


Fig. 7. $S$ measures over 2 000 iterations for 30 teams using the FIFA league ranking relative fitness function.

[19] ——, "Anaconda Defeats Hoyle 6-0: A Case Study Competing an Evolved Checkers Program against Commercially Available Software," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 2, 2002, pp. 857–863.

[20] D. Fogel, *Blondie24: Playing at the Edge of AI*. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 2002.