Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Tournament Particle Swarm Optimization

Willem H. Duminy
Department of Computer Science
University of Pretoria
willem.duminy@gmail.com

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
engel@cs.up.ac.za

***Abstract*— This paper introduces Tournament Particle Swarm Optimization (PSO) as a method to optimize weights of game tree evaluation functions in a competitive environment using Particle Swarm Optimization. This method makes use of tournaments to ensure a fair evaluation of the performance of particles in the swarm, relative to that of other particles. The empirical work presented compares the performance of different tournament methods that can be applied to the Tournament PSO, with application to Checkers.**

Keywords: PSO, machine learning, game playing, tournament, competitive co-evolution

I. Introduction

The optimization of weights to improve the performance of an individual in a competitive environment is a general problem that is independent of the representation scheme in which the weights are found. Typical representation schemes are the linear evaluation function, first used for learning by Samuel [1], and neural networks that gained popularity as a method to represent non-linear knowledge. The optimization for competitors, such as game playing agents is different from many other optimization problems because these agents need good opponents for effective training. This difference leads to the need for innovative optimization methods geared for *competitive* optimization problems. Tournament PSO exploits the well-established methods used to identify the best team or the best individual in human competitive endeavours. This paper evaluates different tournament methods using Checkers.

Section II describes Particle Swarm Optimization (PSO). Special attention is given to the factors that influence the trajectory of a particle. Section III discusses the Competitive PSO, provides a review of the different tournament methods, and introduces the Tournament PSO. Section IV describes the aspects that are common to the experiments that follow. Section V describes an experiment that determines whether it is best to include a particle in its own neighbourhood. Section VI compares the performance of various tournament methods, and Section VII considers the behaviour of the best performing methods during the PSO run.

Section VIII summarizes the findings and contains some concluding remarks.

II. Particle Swarm Optimization

Kennedy and Eberhart [2] introduce Particle Swarm Optimization (PSO) as an algorithm that searches through a multi-dimensional problem space for an optimal solution. An optimal solution is a vector in the search space that maximizes (or minimizes) a given function. This function, called a fitness function, maps a real-valued vector to a real value. During the search, a constant number of search locations, called particles are kept current. The search is conducted by changing the velocity of every particle at each iteration. An iteration is referred to as an epoch, and the collection of particles form a swarm. PSO is distinguished from other searches that keep multiple locations current by the dynamic influence other locations have on the trajectory of a particle [3].

A particle, identified by a search location, has a few key properties: a velocity, a fitness value, a personal best, a neighbourhood and a local best. A particle is assigned a velocity at the start of each epoch. This vector is added to the current location of the particle to determine the new location. The fitness value of a particle is the value of the fitness function at the current location of the particle. The personal best is a location on the path of a particle at which the best fitness value was achieved. The neighbourhood is a static property of a particle; it is the subset of all the particles in the swarm that exert an influence on its velocity. The local best is the personal best of the best particle in the neighbourhood.

The outline of the PSO algorithm is shown in Figure 1. The notation for the location vector of particle p is $\vec{p}$. The personal best of p is *pbest*(p). The i^{th} component of $\vec{p}$ is denoted as $\vec{p}[i]$. The velocity of p is denoted as $v(p)$, with the i^{th} component of the velocity, denoted as $v(p)[i]$.

A. Particle trajectory

The trajectory of a particle p is influenced by its personal best, *pbest*(p) and its local best, *lbest*(p) [4]. These two locations to which a particle is attracted are fundamental to the hypothesis from which PSO originates. According to this hypothesis, the survival of a species depends on the ability of its individuals to gain information from personal experience and share it through social interaction [2]. This social information is represented in PSO by the local best, and the personal experience, called cognitive information, is captured by the personal best. The strong relationship between the two elements is highlighted by the following definition:

Definition 1: *Local Best.* For particle p with neighbourhood $\mathbf{N}(p)$, and maximising fitness function f, *lbest*(p) is an arbitrary selection from the local best set $\mathbf{L}(p)$, where

$\mathbf{L}(p) = \{pbest(q) \mid q \in \mathbf{N}(p) \ \wedge \ \forall_{x \in \mathbf{N}(p)} f(pbest(x)) \leq f(pbest(q))$

The movement of a particle is controlled by the velocity update function (line 2 in Figure 1). This function calculates the new velocity by adjusting each component of the velocity vector separately. Two constants regulate the magnitude of the velocity change: the cognitive acceleration constant, c_1, constrains the change towards the personal best, and the social acceleration constant, c_2, constrains the change towards the local best. In addition, Shi and Eberhart [5] introduced an inertia weight, w, that restricts the influence the current velocity has on the new velocity. Choosing different values for c_1, c_2 and w has a measurable effect on the performance of PSO [6], [7], [8].

The two components, r and s, are the stochastic elements of the velocity update function. The values of these elements influence the rate of change toward personal best and local best respectively. This brings about an uncontrolled exploration of the search space.

The state of particle p at epoch t can be denoted as p_t. The velocity at epoch $t+1$, $v(p_{t+1})$ is determined from the state of the particle at the preceding epoch using the following formula:

$$\begin{aligned} v(p_{t+1})[i] &= w \times v(p_t)[i] + \\ &\quad r_t \times c_1(\mathit{pbest}(p_t)[i] - p_t[i]) + \\ &\quad s_t \times c_2(\mathit{lbest}(p_t)[i] - p_t[i]) \end{aligned} \tag{1}$$

where $r_t, s_t \sim U(0,1)$. Note that these random values are also time-dependent.

Clerc and Kennedy [9] define constraint equations that improve the probability of convergence. These equations are combined with Equation 1 to derive the constricted function:

$$\begin{aligned} v(p_{t+1})[i] &= \chi \times (v(p_t)[i] + \\ &\quad r_t \times c_1(\mathit{pbest}(p_t)[i] - p_t[i]) + \\ &\quad s_t \times c_2(\mathit{lbest}(p_t)[i] - p_t[i])) \end{aligned} \tag{2}$$

Algorithm Particle Swarm Optimization
Input The fitness function, f that takes m arguments and the size of the swarm n
Output A vector $\vec{p}$ encountered during the search where $f(\vec{p})$ had the best value.

```
PSO(f,n)
    Create a swarm S such that S = {p_1, p_2, ..., p_n}.
    For each p in S
        For 1 ≤ i ≤ m
            p[i] = r | r is a random value
            v(p)[i] = 0
        pbest(p) = p

1:  While not end of search
        For each p in S
2:          Update v(p)
            For 1 ≤ i ≤ m
                p[i] = p[i] + v(p)[i]
3:          If f(p) > f(pbest(p))
                pbest(p) = p
```

Fig. 1. Outline of the PSO algorithm

The value of the constriction factor, χ, is determined as follows:

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{3}$$

where $\phi = c_1 + c_2$. Clerc and Kennedy [9] found that ϕ must be greater than 4.0 to promote convergence.

III. Competitive Environments

A fitness function in a competitive environment has no absolute optimum. Consider the use of a random moving player as opponent to introduce an objective function. For the fitness function, f, let $L(f)$ denote the number of matches lost and $W(f)$ denote the number of matches won out of a total of N matches played against the random player using f. The performance of f can be calculated as follows:

$$F(N,p) = \frac{N + W(f) - L(f)}{2 \times N} \times 100 \tag{4}$$

Using Equation 4 as the fitness function provides the absolute maximum. Unfortunately, the random player is a weak player, and once this optimum is achieved, the PSO cannot make further improvements to the weights of the fitness function.

A better approach it to exploit the competitive nature of the optimization problem and use the individuals in the swarm as opponents during the optimization process. Angeline and Pollack [10] define a competitive fitness function as any calculation that is dependent on the current population to some degree.

Co-evolution is a learning process in which the learning environment changes as the process continues [11]. An algorithm that optimizes a fixed fitness function is not co-evolutionary, even though the individuals evolve by interacting with each other. Co-evolution requires a dynamic fitness function. The dynamic nature of a fitness function can be achieved by introducing a different set of individuals (or species) that has a cooperative or competitive relation with the original individuals. However, a species also co-evolve with its environment; as it becomes more adept in changing its environment, it needs to become better at adapting to the changed environment. Even if no other species are available, co-evolution of this kind can occur while the individuals compete. In order to separate related concepts, the term 'competitive co-evolution' is used when the fitness of an individual is determined by the fitness of other individuals in the population [12]. Therefore, a PSO that employs a competitive fitness function is an example of competitive co-evolution.

As a variation of self-play, competitive co-evolution is likely to suffer from the two problems that are associated with self-play that lead to strategies that perform poorly. The first problem is that it is likely to get stuck on a self-consistent but a non-optimal strategy [13]. Secondly, there is no guarantee that the portions of the strategy space searched are the most significant ones [14]. These problems are addressed by ensuring that the population diversity is

adequate to avoid local minima and to cover a larger search space.

A simple way to increase the diversity is to use a larger population. However, Tesauro's work on self-play learning illustrates that the learning of BACKGAMMON does not require a large population [13]. In this case, and with stochastic games in general, the source of the diversity is the stochastic element present in the game rules. This diversity is a primary contributor to the success of Tesauro's BACKGAMMON player [11].

Angeline [10] observes that the stochastic elements present in the genetic algorithm population, such as the application of the mutation operator and the probabilistic selection of parents, bring about a level of non-determinism that can be exploited to create a population that is diverse enough for learning through self-play. A similar argument can be put forward for PSO: it also has stochastic elements, and as such it has the potential to maintain the required level of diversity to mitigate the problems related to self-play.

A. The Competitive PSO algorithm

The competitive PSO algorithm optimizes a competitive fitness function using the particles in the PSO swarm as opponents. The first application of competitive PSO, described by Messerschmidt and Engelbrecht [15], is an analysis that compares PSO to the Evolutionary Program (EP) described by Fogel [16]. This analysis indicates that competitive PSO obtains better TIC-TAC-TOE players than those obtained by competitive EP. Franken and Engelbrecht [17] extended the work of Messerschmidt and Engelbrecht on TIC-TAC-TOE by analysing the effect of different PSO structures and neural network topologies on the learning performance. This work was also applied to CHECKERS [18], [19].

The competitive PSO extends the original PSO outlined in Figure 1 by introducing a competition stage at the start of each epoch. During this competition stage, the fitness of every particle is determined using the competitive fitness function.

The competitive fitness function used by Messerschmidt and Engelbrecht selects a constant number of opponents randomly from the swarm for each participant. The participants include all the current locations as well as the personal best of each particle. For each participant a score is kept. At the start of the competition stage, all the scores are initialized to zero. During the competition, the score increments when the participant beats an opponent, and decrements when the participant loses. In the same manner, the opponent's score is also adjusted. The number of matches played during the competition stage depends on the swarm size, n and the number of opponents, k. The number of matches is $2 \times n \times k$.

The Messerschmidt and Engelbrecht competitive function is not fair and it produces fitness values that are too coarse. Depending on the quality of the randomly selected opponents, a good participant could be assigned a bad fitness value, and *vice versa*. The coarseness comes from the limited range of the fitness function and many individuals are bound to take the same fitness value. Consequently, the local best becomes more often than not, an arbitrary choice between seemingly equal candidates in the neighbourhood.

A solution to this problem (of fairness and coarseness) is to allow the personal bests to compete in a tournament that aims to be a fair assessment of the participant's ability. In such a tournament, the fitness value is not a tally of match outcomes; it is a ranking. By using the results obtained by previously matched participants, the number of new matches conducted during the competition step to determine the ranking, is kept to an absolute minimum.

When ranking is used, particle p is considered better than particle q if and only if p has a higher rank than q. This comparison does not imply that there was a match between p and q. Indeed, it is possible that q will beat p, even if p is ranked higher. The fairness of the ranking depends on the tournament structure and the elimination strategy used during the competition. The next section provides some alternatives to consider in this regard.

B. Tournaments

In a tournament, a number of participants compete to decide which participant is the best. A tournament is one of three types: an elimination tournament, a scoring tournament or a hybrid tournament. In an elimination tournament, participants are removed from the tournament until only the winner remains. In a scoring tournament a score is given to each competitor after a match, and the winner is the participant with the highest score. A hybrid tournament contains elimination and scoring stages.

Different types of scoring tournaments are available. The tournament used by Messerschmidt and Engelbrecht [15] and by Fogel [16] is a random subset scoring tournament. In a fixed subset tournament, the set of competitors for a participant does not change every epoch. In a round robin tournament, every participant is matched against every other participant. Because of the high likelihood that more than one participant will have the same score, a tie breaking procedure is required to identify the winner. During this procedure, criteria derived from the tournament can be considered, such as the results of the matches between winning participants; the ratio of number of wins against the number of losses; the difference between the number of wins and number of losses; or simply the number of wins.

Two types of elimination tournaments are common. In a knockout tournament, the loser of a match is eliminated and the winner continues to the next round. This elimination process continues until one participant remains. In a double-elimination tournament, the participant is eliminated after he loses a second time. This second chance is implemented by creating two brackets: a winners bracket, and a losers bracket. The process followed in the winners bracket is the same as the knockout tournament. However, when a participant in this bracket loses a match; the participant is moved to the losers bracket. The losers bracket has two stages to every round: firstly the winners of the previous losers bracket round (or the losers of the very first winners bracket round) compete. In the second stage, the winners of the first stage

compete against the losers of the same round in the winners bracket. The losers of the second stage are eliminated from the tournament, and the winners remain in the losers bracket. This process continues until both brackets have only one remaining participant. The champion of the losers bracket would have lost one match, and the champion of the winners bracket is undefeated. These two champions then compete to determine the winner of the tournament.

In the elimination tournaments, the pairing procedure decides which participants should compete. If unlucky, the second best participant can be paired with the best participant and eliminated at the first round of the tournament. Although the double-elimination tournament mitigates the problem, unfair pairing remains an issue. The simplest pairing procedure is to select the opponents at random. However, if the ability of the participants are known, the pairs can be organized such that the best players are likely to compete in the final rounds. A process called seeding orders the participants according to previous performance from best to worst. From this sequence, pairs are formed by repeatedly removing the first and the last seeded participant. If the number of participants in a round is not even, one participant receives a bye, and moves to the next round without competing.

C. The Tournament PSO algorithm

The tournament employed by Fogel's [16] Genetic Algorithm did not make use of a long term memory, and neither did Messerschmidt and Engelbrecht [15]. This tournament requires a re-evaluation of every individual for each new generation. An alternative approach is to use the personal best to decrease the number of re-evaluations required at each cycle. Fewer matches per cycle makes it practical to use the more match intensive tournament methods described in the previous section. It is from this idea that the Tournament PSO developed.

The key difference between Tournament PSO and the competitive PSO described in Section III-A, is that the particles in tournament PSO compete against their own personal best. Tournament PSO elaborates on competitive PSO by splitting the competition stage into two smaller stages: the personal competition stage and the tournament competition stage. In the personal competition stage each particle's current location competes against its own personal best. If the personal best is beaten, it is replaced with the current location. During the tournament competition stage, the personal bests of all the particles compete in a tournament. The tournament establishes a partial order ranking that is used to determine the local best particle in the neighbourhood.

The first stage of Tournament PSO requires a constant number of matches: for a swarm size of n, n matches are held during the personal competition stage. If, for a given epoch, no new personal bests are identified, the results of the competition phase for the previous epoch is used again to determine the fitness of the particles in the swarm.

Tournament PSO also makes use of a match cache. The match cache is a memory structure used during the tournament competition stage. Although this memory structure increases the memory requirement of the PSO, it does not affect the trajectory of any particle. Its purpose is to avoid needless matches between particles. The match cache keeps the result of the match between the pairs of personal best locations. When the tournament demands a match between two locations that have already been matched, the result is retrieved from match cache. If the result is not available in the cache the particles compete, and the result of the competition is added to the cache. Whenever a new personal best is identified, all matches in which the personal best participates are removed from the match cache. The number of new personal bests are likely to become fewer as the particles approach optimal configurations, and more matches will be replaced by cached results. The net effect is that more computing resources are spent on search space exploration, and less on the evaluation of personal bests.

If more than one neighbour has the same ranking, either the local best is chosen randomly from these, or a knockout tournament between the high rankers is used to decide which neighbour is best. These two approaches are called random best and knockout best respectively.

A tournament match between two participants consists of two games; each participant gets a turn as the first player. For these games, 2 points are awarded to the winner, 1 point each for a draw and 0 points to the loser. The participant with the most number of points after the two games wins the match. If the scores are equal, the winner is chosen at random.

The local match is more strict - in order to replace the personal best, the current particle location must win as first and as second player. This sterner rule aims to prevent situations in which a tested champion is replaced with a 'lucky' novice [11].

If a tournament requires pairing, either random pairing or rank seeding can be employed. In random pairing, opponents are paired by random selection. In rank seeding, the previous rank of the particle is used to order the participants. For a particle that obtained a new personal best, the rank obtained by the previous personal best is used for seeding. From this ordered list, pairs are chosen such that the best seeds are likely to compete in the final rounds.

For the tie breaker procedure, a knockout tournament is held that includes all the winners as competitors. If there are still ties, a winner is chosen randomly.

The tournaments (subset and elimination) can be combined with the random or knockout local best. For elimination tournaments, either random pairing or seeding can be applied. These choices lead to fourteen different permutations, each permutation is a different tournament method that can be used in the Tournament PSO. These methods are summarized and labelled in Table I.

IV. Experimental setup

The experiments were conducted on the game of Checkers (using the rules described in [20]) with no game-tree searching. The PSO found optimal values for the weights of a linear function, labelled F_b. This function is a complex

TABLE I
THE TOURNAMENT METHOD MATRIX

Tournament	**Random best**		**Knockout best**	
Random subset	RSR		RSK	
Fixed subset	FSR		FSK	
Round robin	RRR		RRK	
Pairing	**Random**	**Seeded**	**Random**	**Seeded**
Knockout	KRR	KSR	KRK	KSK
Double elimin.	DRR	DSR	DRK	DSK

expression that employs the notation described by Duminy and Engelbrecht [20]. The function itself produces a weak player that is unable to consistently beat a random moving player. Consequently Equation 4 was used to measure the performance of the function after optimization. The measurements shown in the experiments were obtained using a value of 15000 for N.

Based on a few preliminary experiments, the following values were chosen for the PSO parameters: $w = 1$, $c_1 = 2.1$, $c_2 = 2.1$, $\text{val}_{min} = 1$ and $\text{val}_{max} = 99$. The swarm consisted of 25 particles arranged in a 5×5 lattice. The Von Neumann [3] neighbourhood topology was chosen because it has been shown to be superior in related work [3], [18], [19]. The neighbourhood contains the 4 particles that surround the subject particle in the lattice.

The level of consistency for the experimental conditions was improved by keeping the initial particle locations the same for every PSO run.

The PSO was terminated when 30000 matches were played. During the tournament competition phase, the match count was incremented only when a new match was added to the match cache. The matches were played without searching the game tree. The move choice fell on the next position in the play line with the highest evaluation. In the case where more than one move with the best evaluation were encountered, the next move was randomly chosen from amongst these moves.

V. EXPERIMENT: SELF IN NEIGHBOURHOOD

Given a fixed topology such as Von Neumann, Kennedy and Mendes [3] suggests two neighbourhood composition strategies. The first strategy includes the subject particle[1] in its own neighbourhood, and the second strategy excludes this particle. This choice is significant only when the subject particle is better than the other particles in its neighbourhood. When this is the case the self-including strategy would reinforce the position of the subject particle, while the self-excluding strategy would nudge the subject particle towards the second best particle in the inclusive neighbourhood. Consequently, the self-excluding strategy explores more of the solution space than the self-including strategy and therefore self-excluding is expected to perform better. This hypothesis is empirically evaluated by comparing the performance of the two composition strategies in the context of the Tournament PSO.

[1]the particle for which the fitness value is determined

In order to compare the two strategies, the mean performance of the strategies in the context of every tournament method was considered. Using the Tournament PSO algorithm, each of the fourteen tournament methods were used in twenty runs. Ten of these runs included the subject particle in the neighbourhood, and ten employed the self-excluding strategy. At the end of each run, the performance of the best particle in the swarm was measured using the objective function described above.

The small size of the sample necessitated the use of Student's *t*-distribution[21]. If the observed *F*-value was below the *F*-value cut-off, the two sample t-test were used. For the *F*-value the cut-off for a sample size of 10 is 4.0260. The cut-off for the t-test is:

$$t_{10+10-2}^{(0.025)} = t_{18}^{(0.025)} = 2.101 \tag{5}$$

The measurements taken for the self-excluding and the self-including strategy are shown in Table III and Table II respectively. These tables show the sample mean, sample variance, minimum and maximum for each run.

TABLE II
SELF-INCLUDING PERFORMANCE

S	$\overline{X}_s$	s_s^2	Max	Min
DRK	65.5266	2.2152	67.0433	62.6933
DRR	64.9607	3.6231	67.8300	61.5700
DSK	65.1237	2.0082	69.0767	64.2767
DSR	64.9407	0.8509	65.8167	62.5000
FSK	66.1740	1.6380	67.8967	64.1333
FSR	65.7873	1.9372	67.5900	64.1133
KRK	65.5377	1.3065	67.6200	63.8800
KRR	66.1990	2.4128	68.4833	63.2367
KSK	64.9130	0.2899	65.5967	63.5433
KSR	65.3693	0.3255	66.4967	64.7000
RRK	65.6947	3.1974	68.9167	63.6767
RRR	65.6273	1.5950	67.4167	63.5500
RSK	65.9980	2.6762	68.0833	63.7233
RSR	65.4710	2.7587	67.4300	62.2733

TABLE III
SELF-EXCLUDING PERFORMANCE

S	$\overline{X}_s$	s_s^2	Max	Min
DRK	65.4016	1.1772	66.9133	63.3167
DRR	65.0917	2.6561	67.4033	62.2300
DSK	66.7290	2.5015	69.3867	64.5700
DSR	65.4183	1.7178	67.6000	63.0700
FSK	65.9664	1.4284	68.1233	64.0267
FSR	65.2980	2.7477	68.2867	63.4667
KRK	65.7903	3.4113	68.9633	62.7800
KRR	66.0460	1.5852	68.5467	64.5333
KSK	64.9390	0.6566	66.9633	64.2367
KSR	66.2663	2.9189	68.4400	63.4600
RRK	65.5190	3.5560	68.4367	63.3900
RRR	66.1187	3.7925	68.7533	63.7033
RSK	65.9110	2.3872	67.6500	62.5667
RSR	65.5840	1.5597	67.8100	63.5333

The values for the statistical calculations of the observed measurements are shown in Table IV. Except for KSR all the

TABLE IV
COMPARING THE MEANS OF SELF-INCLUDING AND SELF-EXCLUDING

Method	*F*-value	*t*-value	Result
DRK	1.8817	0.2146	Inconclusive
DRR	1.3640	0.1653	Inconclusive
DSK	1.2456	2.3905	Excluding is better
DSR	2.0188	0.9425	Inconclusive
FSK	1.1467	0.3750	Inconclusive
FSR	1.4184	0.7149	Inconclusive
KRK	2.6110	0.3678	Inconclusive
KRR	1.5221	0.2420	Inconclusive
KSK	2.2652	0.0846	Inconclusive
KSR	8.9671	-	Inconclusive
RRK	1.1122	0.2138	Inconclusive
RRR	2.3777	0.6694	Inconclusive
RSK	1.1210	0.1223	Inconclusive
RSR	1.7688	0.1720	Inconclusive

tournament methods produced an *F*-value well below the cut-off value. Consequently the two-sample test must be applied to most of the tournament methods. Almost all the calculated *t*-values were in the acceptance region, $-2.101 < t_{18} < 2.101$, and the conclusion is that most means are equal. The only tournament method that shows a better mean for the self-excluding strategy is DSK.

For the KSR tournament method $n^* = 11.42329$ and $t^* = 1.57479774$. Using the nearest integer value gives the cut-off value for the approximate t-test at a significance level of 5% as:

$$t_{11}^{(0.025)} = 2.201 \tag{6}$$

The measured value, 1.57479774, is below the cut-off, and therefore the hypothesis that the two means of KSR are equal cannot be rejected.

Although the minimum and maximum measurements hint toward the conclusion that self-excluding strategies are better, this conclusion would not be statistically sound. A more accurate general conclusion is that the performance of a specific Tournament PSO that uses the self-excluding strategy is no worse than the performance of a PSO that is the same as the specified PSO in other respects but uses a self-including strategy. However, a specific conclusion can be stated for the DSK tournament method – for this method, the self-excluding strategy outperforms the self-including strategy.

VI. EXPERIMENT: TOURNAMENT METHOD PERFORMANCE COMPARISON

This experiment applied the self-excluding strategy to each of the tournament methods identified in Section III-C to compare the performance of the optimized playing agents.

The method described in Section V is also used for this experiment, and the discussion below provides a comparative analysis of the statistics obtained from the self-excluding runs conducted during the previous experiment.

The results for F_b is shown in Table III, and depicted as a box-and-whisker diagram in Figure 2. Taken as a whole, the mean performance of all tournament methods is 65.71995. From Figure 2, it is clear that no single strategy can be isolated as the best strategy, but it seems very plausible that KSK is the worst tournament method for Tournament PSO.

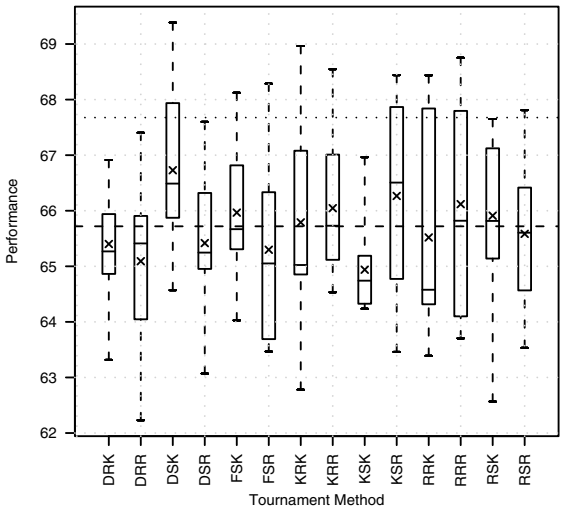

Fig. 2. The performance of the tournament methods

In order to identify the better strategies accurately, the mean value of every strategy were tested for equality against the mean value of the strategy that obtained the greatest mean value. The best performer was DSK with a mean value of 66.7290. As before, the *F*-value and the *t*-value were used to determine the equality of the two means. The results of these calculations are shown in Table V.

TABLE V
COMPARING OTHER METHOD MEANS WITH THE MEAN OF DSK

S	$\overline{X}_S$	s_S^2	**F-value**	**t-value**	**Result**
DRK	65.4016	1.1772	2.1249	2.1885	Not equal
DRR	65.0917	2.6561	1.0618	2.2799	Not equal
DSK	66.7290	2.5015	1.0000	0.0000	Equal
DSR	65.4183	1.7178	1.4562	2.0178	Not equal
FSK	65.9664	1.4284	1.7512	1.2166	Equal
FSR	65.2980	2.7477	1.0984	1.9751	Not equal
KRK	65.7903	3.4113	1.3637	1.2207	Equal
KRR	66.0460	1.5852	1.5780	1.0684	Equal
KSK	64.9390	0.6566	3.8100	3.1852	Not equal
KSR	66.2663	2.9189	1.1669	0.6284	Equal
RRK	65.5190	3.5560	1.4216	1.5547	Equal
RRR	66.1187	3.7925	1.5161	0.7693	Equal
RSK	65.9110	2.3872	1.0479	1.1699	Equal
RSR	65.5840	1.5597	1.6038	1.7967	Not equal

Only six of the fourteen tournament methods were found to be less effective than DSK. Table VI emphasises the stronger tournament methods by marking the places of the six weaker methods with asterisks ('***').

The first deduction that can be made from these results is a predictable one: random best is weaker than knockout best because the latter was used by only two of the six eliminated

TABLE VI
THE DOMINANT TOURNAMENT METHOD MATRIX

Tournament	**Random best**		**Knockout best**	
Random subset	***		RSK	
Fixed subset	***		FSK	
Round robin	RRR		RRK	
Pairing	**Random**	**Seeded**	**Random**	**Seeded**
Knockout	KRR	KSR	KRK	***
Double elimin.	***	***	***	DSK

methods. The second deduction is that double elimination is less effective than knockout tournaments. Double-elimination takes more rounds to determine the winner, and more epochs can be achieved with the same number of games using the knockout tournaments. However, DSK is a double-elimination strategy that did very well. The reason for this could be that DSK also used the two other aspects that intuitively introduce more fairness: that is the use of seeding for tournaments and the use of a knockout tournament to determine the best of equal particles in the neighbourhood.

VII. EXPERIMENT: TOURNAMENT METHOD INTERVAL ANALYSIS

This experiment aimed to identify the better methods among the 8 methods isolated by the previous experiment. Here, the performance of the functions created at various intervals during the PSO run were considered.

For each of the following tournament methods: KSR, RRR, RRK, FSK, KRK, KRR and RSK, the Tournament PSO was run 10 times. During each run, 31 measurements were taken. The first measurement used the weights of the swarm champion after the very first tournament, and the other measurements were taken using the weights of the current champion after every interval of 1000 matches. In total 2400 functions were measured.

For the analysis the measurements were aggregated into interval means. The interval mean is the mean of the ten measurements taken at an interval for a tournament method. Thus, for each tournament 31 interval means were obtained.

A formal statistical analysis of the interval means is not done for this experiment because the previous experiment concluded that the performance of the final interval mean is statistically equal and there is no reason to expect the sample variances of the other intervals to be less than the variances of the final intervals. Therefore, a statistical analysis is likely to conclude that all the interval means are equal. The results of this experiment undergoes a less formal, but still a reasonable analysis with two steps.

The first step of the analysis is simply a count of the number of interval means that is greater than the initial interval mean for that method. If this count is thirty, it means that the method constantly produces a function that performs better than the initial function. However, a count less than 15 indicates that the method is likely to produce functions that perform worse than the initial function. In such a case the tournament method fails to identify and to promote better functions. It could also indicate that the method is susceptible to local minima.

The second step of the analysis compares the interval means of each tournament at the intervals. If one of the methods consistently produces a mean greater than the other methods, it is likely that this method is better than the others.

Figure 3 shows the number of interval means that were greater than the mean of the performance of the initial function. Only DSK and KSR obtained a maximum count

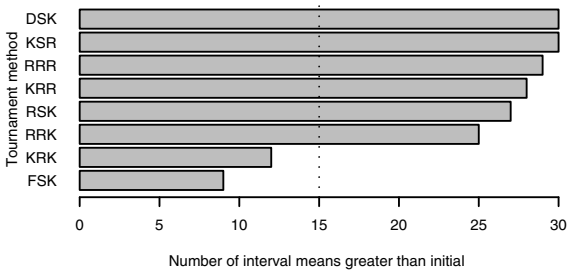

Fig. 3. Tournament method interval performance

of thirty. KRK and FSK had a count lower than fifteen. The conclusion is that these two methods failed to produce functions that consistently perform better as the PSO search continues.

The second analysis excludes KRK and FSK. Figure 4 shows the interval means for each of the six remaining tournament methods. The greatest values are amongst the

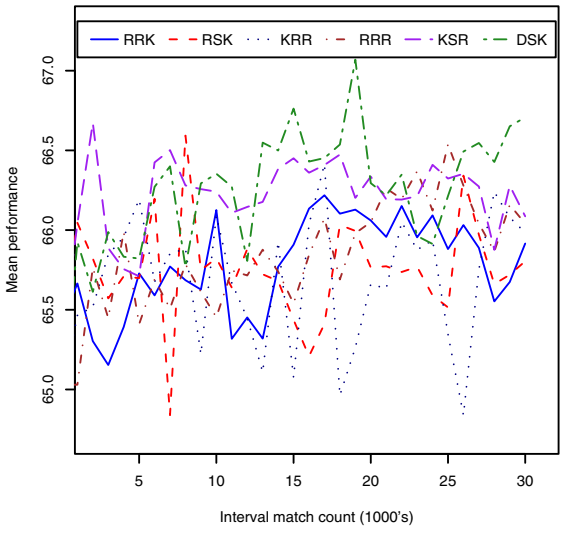

Fig. 4. Tournament method convergence

means of the following methods: KSR, DSK, KRR, RSK

and RRR. The only tournament method that had no interval with the greatest mean is RRK. The methods that has intervals with the smallest mean are restricted to RRK, RSK and KRR.

From Figure 4 the dominance of DSK as the maximum mean for each interval is clear. What follows is the count of intervals for which each strategy achieved the maximum interval mean: KRR = 1, RSK = 2, KSR = 6, RRR = 4 and finally DSK had 17 of the maximum intervals.

For DSK all the intervals produced a better performing function than the initial function, and DSK had the best performing function in more than half of the measured intervals. Thus according to this analysis, DSK performs better than the other tournament methods.

VIII. Conclusion

The Tournament PSO operates in a competitive environment and employs a competitive fitness function that uses the other particles in the swarm to determine a particle's fitness. It is more fair than the Competitive PSO because it uses a global tournament to choose the local best particle. This fairness contributes to an improved optimization ability.

The Tournament PSO introduces the idea of ranking all particles according to a tournament method, and the fitness value of the particle is its rank. In addition, this new PSO redefines the personal best: if the current location beats the personal best in a match, it becomes the new personal best.

Tournament PSO brings with it the question of which Tournament method is better. The experimental results support the intuitive notion that the most fair and least coarse method does indeed perform better. Also, it is less likely to converge early. This method, labelled DSK uses the double-elimination tournament with seeding, and the knockout tournament to select the local best amongst equals.

References

[1] A. Samuel, "Some studies in machine learning using the game of Checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, Perth, Australia, Nov/Dec 1995, pp. 1942–1948.

[3] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of IEEE congress on Evolutionary Computation*, vol. 2, May 2002, pp. 1671–1676.

[4] R. C. Eberhart and J. Kennedy, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[5] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE congress on Evolutionary Computation*, vol. 3, Anchorage, AK, May 1998.

[6] P. Suganthan, "Particle swarm optimizer with neighbourhood operator," in *Proceedings of IEEE congress on Evolutionary Computation*, Washington DC, USA, July 1999, pp. 1958–1961.

[7] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Annual Conference on Evolutionary Programming*, San Diego, 1998.

[8] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of IEEE congress on Evolutionary Computation*, vol. 3, Washington DC, USA, July 1999, pp. 1949–1950.

[9] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–60, February 2002.

[10] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, pp. 264–270.

[11] J. B. Pollack, A. D. Blair, and M. Land, "Coevolution of a backgammon player," in *Proceedings of the Fifth International Conference on Artificial Life*, Nara, Japan, May 1996.

[12] R. P. Wiegand, W. C. Liles, and K. A. D. Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California, USA: Morgan Kaufmann, 7-11 2001, pp. 1235–1242. [Online]. Available: citeseer.ist.psu.edu/481900.html

[13] G. Tesauro, "Practical issues in temporal difference learning." *Machine Learning*, vol. 8, pp. 257–277, 1992.

[14] S. L. Epstein., "Toward an ideal trainer," *Machine Learning*, no. 15, pp. 251–277, 1994.

[15] L. Messerschmidt and A. P. Engelbrecht, "Learning to play games using a PSO-based competitive learning approach," in *Proceedings of the 40th Asia-Pacific Conference on Simulated Evolution and Learning*, Singapore, 2002.

[16] D. B. Fogel, *Blondie24: Playing at the edge of AI*. Morgan Kaufmann, 2001.

[17] N. Franken, *PSO-Based Coevolutionary Game Learning (M.Sc. dissertation)*. University of Pretoria, 2004.

[18] N. Franken and A. P. Engelbrecht, "Comparing PSO structures to learn the game of checkers from zero knowledge," in *Proceedings of IEEE congress on Evolutionary Computation*, Canberra, Australia, 2003.

[19] ——, "Evolving intelligent game-playing agents," in *Proceedings of SAICSIT*, 2003, pp. 111–113.

[20] W. H. Duminy and A. P. Engelbrecht, "Composing linear evaluation functions from observable features," *South African Computer Journal*, no. 35, pp. 48–58, December 2005.

[21] L. Underhill and D. Bradfield, *Introstat, Second Edition*. Juta & Co., Ltd, 2001.

[22] *Proceedings of IEEE congress on Evolutionary Computation*, Washington DC, USA, July 1999.