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

Pareto Evolution and Co-Evolution in Cognitive Neural Agents Synthesis for Tic-Tac-Toe

Yi Jack Yau, Jason Teo and Patricia Anthony
Center for Artificial Intelligence
School of Engineering & Information Technology
Universiti Malaysia Sabah, Locked Bag No. 2073,
88999 Kota Kinabalu, Sabah, Malaysia.
yijackyau@gmail.com, jtwteo@ums.edu.my, panthony@ums.edu.com

Abstract— **Although a number of multi-objective evolutionary algorithms (MOEAs) have been proposed over the last two decades, very few studies have utilized MOEAs for game agent synthesis. Recently, we have suggested a co-evolutionary implementation using the Pareto Evolutionary Programming (PEP) algorithm. This paper describes a series of experiments using PEP for evolving artificial neural networks (ANNs) that act as game-playing agents. Three systems are compared: (i) a canonical PEP system, (ii) a co-evolving PEP system (PCEP) with 3 different setups, and (iii) a co-evolving PEP system that uses an archive (PCEP-A) with 3 different setups. The aim of this study is to provide insights on the effects of including co-evolutionary techniques on a MOEA by investigating and comparing these 3 different approaches in evolving intelligent agents as both first and second players in a deterministic zero-sum board game. The results indicate that the canonical PEP system outperformed both co-evolutionary PEP systems as it was able to evolve ANN agents with higher quality game-playing performance as both first and second game players. Hence, this study shows that a canonical MOEA without co-evolution is desirable for the synthesis of cognitive game AI agents.**

Keywords: Game AI, Co-Evolution, Evolutionary Artificial Neural Networks, Pareto Differential Evolution, Evolutionary Multi-Objective Optimization

I. Introduction

Single objective decision models are sufficient for some decision making processes, but there are many situations where decisions have multiple objectives. Most real-world problems consist of more than one objective, which are normally known as multi-objective problems (MOPs). Since objectives of MOPs (normally) conflict with each other, a multi-objective problem cannot be answered with a single solution alone. Based on "Pareto Optimum", a set of global non-dominated solutions (also known as Pareto optimal solutions) are optimal trade-offs among the objectives. Evolutionary Algorithms (EAs) are population-based stochastic search methods that apply the metaphor of natural biological evolution. EA operations apply the principle of survival of the fittest in trying to produce more optimized solution(s). Over the last few years, the use of EAs for multi-objective optimization tasks has become very popular with a rapid increase of new algorithms, theoretical achievements and novel applications [1].

Artificial intelligence for games (game AI) represents one of the most useful and practical platforms for studying evolutionary computation systems. In spite of the additional complexities of co-evolutionary models, they hold some significant advantages that have been exploited within the context of EAs to support the generation of solutions to a series of complex problems. Co-evolutionary techniques have been successfully applied to a number of games, for instance, Awari [2], Pong [3], Nim [4], Poker [5], and Go [6], [7]. Although many single-objective evolutionary techniques have been successfully applied to many different kinds of games, a large number of research issues and questions still remain for multi-objective evolutionary techniques when applied to games [3], [5].

In previous works [8], [9], implementation of Evolutionary Programming (EP) has also been used to create ANNs that are capable of playing Tic-Tac-Toe (TTT). It was reported to be able to automatically synthesize neural network game-playing agents both as the first player with reasonable playing strength only. Recently, we have suggested an enhanced version of a hybrid adaptive/self-adaptive co-evolutionary implementation using the Pareto Evolutionary Programming (PEP) algorithm known as the Pareto Co-Evolutionary Programming (PCEP) algorithm was reported to be able to automatically synthesize neural network game-playing agents both as the first and second players with reasonable playing strength through the introduction of Pareto multi-objective evolution [10].

In this study, the main objective is to look into the effects of the introduction of the co-evolution technique and whether it is actually beneficial or otherwise to the Pareto evolutionary optimization process. A comprehensive empirical comparison of performance between the systems of PCEP, a new archived-based version of PCEP called PCEP-A and the canonical PEP without co-evolution is carried out. All of the above implementations do not require an explicit evaluation function for the purpose of automatically

generating the game AI for TTT since the scoring from playing against a rule-based player is used as the objective evaluation method during evolution. Finally, the performance of the respective approaches will be measured according to the playing strength of the evolved ANN game-playing agents pitted against three different levels of players (expert, medium and random players).

A. Tic-Tac-Toe

TTT is a standard two-player zero-sum game of perfect information, in which two players alternately put crosses and circles in one of the compartments of a three-by-three board. The objective of the game is to get a row of three crosses or three circles before the opponent does. Player one is the player that moves first, making a cross, followed by player two, making a circle. If at the end of the game both players cannot meet the objective, it means that a draw is awarded to both players.

There are four player types in TTT. The novice player makes random moves, the intermediate player will block their opponent from winning, the experienced player knows that playing in certain first squares will lose the game, and the expert player will never lose [11]. When both players are at the expert level, a TTT first player's purpose is to force a win or a draw; however a second player should force a draw by blocking the first player's winning moves. This is because if the first player starts the game with an optimal first move, it will never lose if no mistake is made for following moves, so the second player can only force a tie. The only chance for a second player to force a win is when the first player did not make a best first move, or making a mistake during subsequent moves. Hence, if both players are playing with an optimal strategy with no mistakes, every game will end in a tie.

II. Pareto Evolutionary Programming (*PEP*)

We have a multi-objective problem with two objectives in this study, to: (1) optimize the performance as the first player (which always trying to force a win), and also (2) optimize the performance as the second player (which will try to force a draw at the expert level). The Pareto-frontier of the trade-off between the two objectives will have a set of ANNs with different levels of play-strength as the first and second players for TTT. The PEP algorithm is similar to the co-evolutionary EP system in [8] with the following modifications:-

1) Each agent is evaluated as both first and second TTT players.
2) Reproduction is undertaken only among first 50 non-dominated solutions from (first) n-Pareto layer(s) in each generation.
3) Offsprings are placed into the population if they dominate the main parent.

Evolved artificial neural networks (ANNs) act as the cognitive game AI agents in the game. The system was initialized with a population of 100 ANNs, each one having its weight connections and bias term value set at random in a uniform distribution ranging over $[-0.5, 0.5]$. Each parent created an offspring through mutation of each weight and bias term value by adding a Gaussian random variable with zero mean and a standard deviation of 1 ($GaussianF(0,1)$). Based on the mutation rate, the number of nodes in the hidden layer was allowed to vary, subject to the constraints on the maximum and minimum number of nodes. All new added node weights are set to 0.0. All layers (l) of ANN (the input layer, hidden layer and output layer) and all the synapses (connection between input layer and hidden layer, and also between hidden layer and output layer, bl) are involved in mutation, which was the only genetic operation for reproduction. The algorithm works as described in the following section.

A. Pseudocode of PEP

1) *Randomly initialize population of 100 ANNs, each with its weight of connections (W) and bias terms (B) value over* $[-0.5, 0.5]$.
2) *Repeat:*
   a. *Evaluate individuals in the population and mark non-dominated ANNs.*
   b. *If the number of non-dominated ANNs is less than 50, repeat the following until the number of non-dominated ANNs is greater than or equal to 50:*
      i. *Find the next layer of non-dominated solutions among those marked ANNs.*
      ii. *Re-mark the ANNs as non-dominated.*
   c. *Delete all dominated ANNs from the population.*
   d. *Repeat:*
      i. *Randomly select an ANN as the parent (p_1).*
      ii. ***Mutation**: with probability of $mutationRate$, do*

$$B_l^{child} \leftarrow B_l^{p_1} + GaussianF(0,1)$$
$$W_{bl}^{child} \leftarrow W_{bl}^{p_1} + GaussianF(0,1)$$

otherwise

$$B_l^{child} \leftarrow B_l^{p_1}$$
$$W_{bl}^{child} \leftarrow W_{bl}^{p_1}$$

and with probability of $mutationRate$, do

$$nodeNum \leftarrow vary(nodeNum)$$

where, all layers (l) of ANN (the input layer, hidden layer and output layer) and all the synapses (connection between input layer and hidden layer, and also between hidden layer and output layer, bl) are involved in both operations of reproduction.

e. *Until the population size is maximum (100).*

3) *Until termination conditions are satisfied, else return to (2) above.*

The non-dominated solutions (also known as Pareto optimal solution) here is one agent which none of the other agents have a higher fitness in at least one of the objectives. More details about optimality in MOPs can be found in [12].

B. Evaluation of individuals (in PEP)

Each ANN will compete with the same rule-based procedure as the first player "X" in 2 sets of 8 games and the second player "O" in 2 set of 9 games, where the number of game is based on number of possible first move for the rule-based player. The first move of the rule-based player will not be repeated in each set of games, based on all possible moves being stored in an array at the beginning of the particular set of games. Two different payoff functions were used to reward each agent is performance as the first and second player. For grading the performance of the ANN as the first player, the payoff function $\{+1, -10, 0\}$ are the rewards for winning, losing, and drawing, respectively. However, for grading the performance of the ANN as the second player, the payoff function $\{+2, -5, 3\}$ are the rewards for winning, losing, and drawing, respectively. Marking non-dominated solutions will be done directly based on the scores gained from these two payoff functions which were obtained from preliminary testing and early work of Fogel [8].

III. Pareto Co-Evolutionary Programming (*PCEP*)

The PCEP algorithm introduced co-evolutionary techniques into the MOEA which is the PEP algorithm, where the force of evolution is from the competition among the (evolving) ANNs. The main difference between PEP and PCEP is the evaluation of each individual. For implementation of PCEP, after completing the evaluation using the rule-based agent and grading with payoff function similar with PEP (see Section II-B), each ANN will then be compared with a constant number of randomly picked ANNs from the population of the current generation. If the score of the ANN was greater than or equal to its opponent (the randomly picked ANN), it will receive a win. Furthermore, the ranking of the first Pareto layer (by marking non-dominated solutions) will be based on the number of wins as the main evaluation criteria.

*A. Pareto Co-Evolutionary Programming with an Archive (**PCEP-A**)*

Similar to PCEP, after grading each ANN using the payoff functions, a second competition is held. However, PCEP-A has an extra archive, which is used to store Pareto solutions at every 50^{th} generation. Consequently, each ANN is compared to a minimum number of randomly picked ANNs (without repetition) from the archive. Only if the number of ANNs in the archive is less then the minimum required number of random opponents, then the opponent list will be filled with randomly picked ANNs from the population. Similarly, an ANN will receive a win, if its score is greater than or equal to its competitor. The number of wins will then be used as the main evaluation criteria for marking non-dominated solutions to rank the first Pareto layer.

IV. Adaptive Evolution

In adaptive evolution, direction and/or magnitude of the strategy parameters' modification is decided using some form of feedback from the EA. However, in self-adaptive evolution, self-adaptation of parameters is the implementation of the **evolution of evolution** idea. A hybrid adaptive/self-adaptive evolution combines two adaptation methods mentioned above. The mutation rate is encoded into the chromosomes of individuals. Instead of undergoing genetic operations of mutation and recombination, these parameters will be varied by some deduction within a range. The highest ANN's score value of the current generation is used as the feedback from the EA. If the feedback comprises of a non-negative value, a deduction within a range will be applied to the strategy parameters. For all the experiments in this study, the probability of the initialization value and deduction range (for mutation rate which is only decreased overtime) are 1 and between 0.0001 and 0.0005, respectively.

V. Cognitive Game AI Representation

The cognitive game AI is represented by a standard multi-layered feed-forward ANN. A board pattern is received as the input of the ANN, and the output of the ANN is a position of the board as the corresponding move. Each node of the hidden layer and output layer performs a sum of the weighted input strengths, subtracts off an adaptable threshold and passes the result through a sigmoid filter as shown in (1),

$$\frac{1}{1 + e^{-x}} \tag{1}$$

where x is the sum of the weighted input strengths.

The ANN's input layer consisted of nine input nodes (with an additional bias unit), a hidden layer of varying size (between 1 and 10 hidden nodes with an additional bias unit) and the output layer consisted of nine nodes, where each of the input and output nodes corresponded to a square in the TTT grid. The three-by-three matrix board state is represented as a 2-dimensional three-by-three array of nine values. A blank open space was denoted by the value 0.0, an "X" was denoted by the value 1.0, and an "O" was denoted by the value (minus) -1.0. The two-dimensional array represents the current board pattern and is presented to the ANN to determine the move of the opposing player and correspondingly, the relative strengths of the nine output nodes were examined to determine the equivalent counter-move by the game AI system. An empty square's position

with the maximum output strength was chosen as the output. This is to ensure only legal moves are made. Placed squares were ignored and selection pressure was not applied to force the output to zero [9].

VI. Experimental Setup

This series of experiments was designed to examine and observe the effects of synthesizing TTT agents with and without the introduction of co-evolutionary techniques into the PEP algorithm. Settings of implementations of PCEP and PCEP-A involved in all experiments were directly adopted from the PEP setup (see section II). Each experiment was repeated for 50 trial runs, and each run was allowed a maximum of 800 generations. See Table I for the details of the experimental setup, the number and target location of randomly picked opponents.

TABLE I

Experimental setup details. The main differences between each system were the number of opponents (that will compete with each candidate ANN in the population) and where the opponents were picked from.

System	Opponent's No.	Origin of Opponent
PEP	0	None
PCEP1	30	Population
PCEP2	50	Population
PCEP3	70	Population
PCEP-A1	30	Archive & Population
PCEP-A2	50	Archive & Population
PCEP-A3	70	Archive & Population

After completing all of the experiments, all agents representing Pareto solutions at the 800th generation from each experiment are selected to be the representatives of each system. Pareto multi-objective optimization systems from all the above systems are able to synthesize both first and second players in a single run. Consequently, each selected ANN has to compete with three different levels of evaluation players, that are a near-perfect expert player, an average player and a random player. A competition consists of a set of 20 games for each level of player. A selected ANN will thus play a total of 60 games firstly as a first player and secondly as a second player.

VII. Experimental Results & Discussion

Figure 1 shows the overall performance of all experiments and is summarized by focusing on the number of lost games only. It clearly shows all experiments successfully produced agent(s) that never lost any games to all three level of players as the first player. Nevertheless, only PEP, PCEP2, PCEP3 and PCEP-A1 successfully produced intelligent agent(s) that never lost any game to the expert level of player as the second player. Figure 2 shows the global Pareto solution(s) front for all the experiments in this study. Figures 3, 4, 5 and 6 show

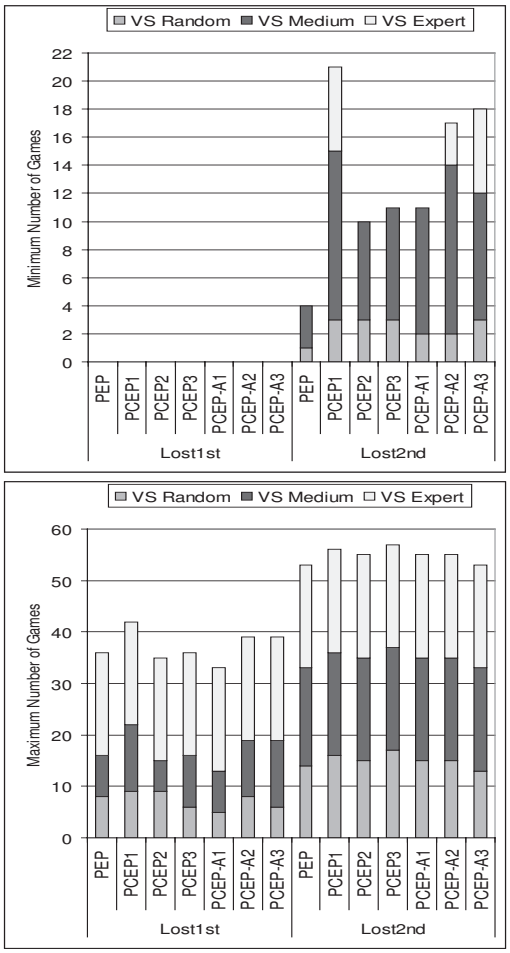


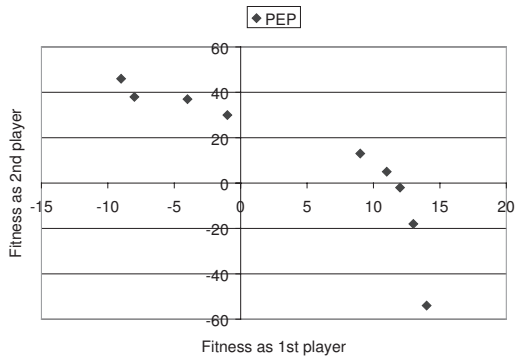
Fig. 1. Each selected ANN has to compete against three different levels of evaluation players, that are a near-perfect expert player (*VS Expert*), an average player (*VS Medium*) and a random player (*VS Random*). This figure shows minimum and maximum number of games lost as the first player (*Lost1st*) and the second player (*Lost2nd*), respectively.

details of the overall performances of the selected agents from each system involved in the competetion against all three level of players as the first and second players.

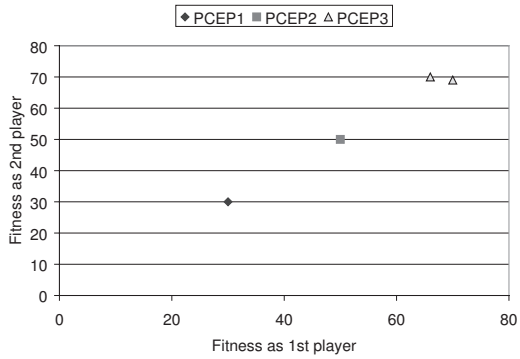
A. The Introduction of Co-evolution

Figure 3 shows the details of the overall performances of implementations from the PCEP system competing as the first and second players against three different levels (expert, medium and random) of evaluation players. Overall, PCEP2 was successful in outperforming PCEP1 and PCEP3 in terms of producing good performing agents both as first and second players.

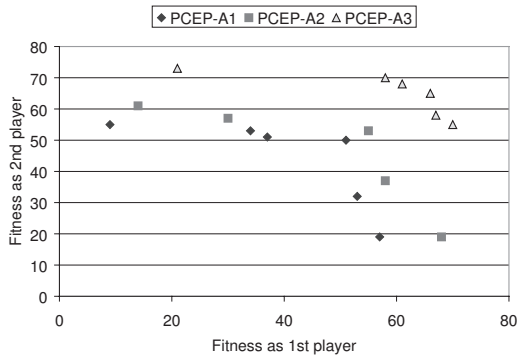
Based on the results of the competition against the expert-level player, PCEP2 and PCEP3 were successful in producing agent(s) that never lost to the expert-level player as the first and second players. PCEP1 was successful in producing agent(s) that never lost any game to the expert-level player



(a) PEP system



(b) PCEP systems



(c) PCEP-A systems

Fig. 2. Global Pareto Solution(s) of each experiment

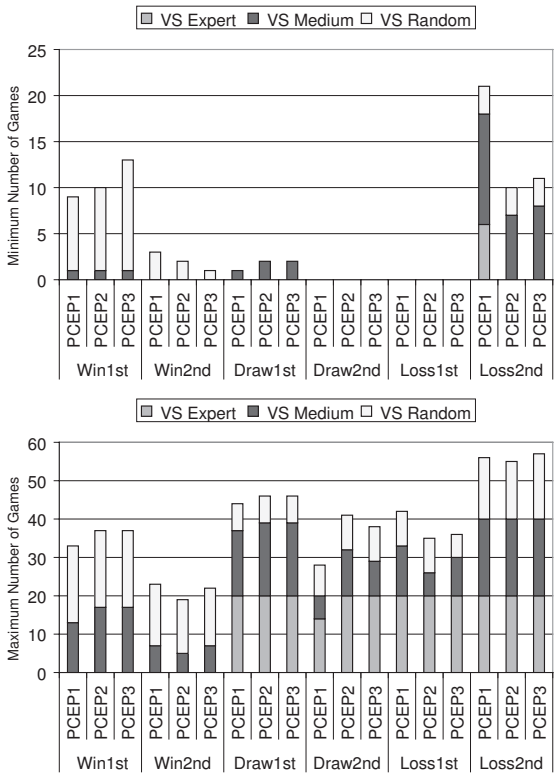


Fig. 3. Selected ANNs from implementations of PCEP competing against a near-perfect expert player (*VS Expert*), an average player (*VS Medium*) and a random player (*VS Random*). This figure shows the minimum (*Min*) and maximum (*Max*) number of games won (*Win1st, Win2nd*), drawn (*Draw1st, Draw2nd*) and lost (*Loss1st, Loss2nd*) as the first and second players, respectively.

as the first player but its best second player lost 6 games to the expert-level player.

Based on the results obtained after competing with the medium-level player, PCEP2 performed slightly better than PCEP1 and PCEP3. Each selected agent from PCEP2 had the lowest maximum number of lost games as the first player, and the lowest minimum number of lost games as the second player. Similarly, PCEP2 and PCEP3 were performing better than PCEP1, where both had the highest maximum number of games won as the first player. Nevertheless, PCEP3 was performing slightly better than PCEP1 and PCEP2 when competing against the random player. The minimum number of wins as the first and second player of the selected agent(s) from PCEP3 was slightly higher than other agents. In addition, the maximum number of losses as the first player of selected networks from PCEP3 was also slightly lower than other representatives as well.

The only difference between each implementation of this algorithm (PCEP) is the number of randomly picked opponents. The increase (from 30 to 50 comparisons) successfully improved the performance of PCEP but the performance decreased for PCEP3, which is closer to a round-robin (70 comparisons). The marking of dominated agents in PCEP is based on the number of wins obtained from the evaluation against a constant number of randomly picked opponents from the population of the current generation. This evaluation is only presenting "performance of the current generation" with a high probability of luck involved. All implementations of PCEP did not have a good spread of global non-dominated solutions. Furthermore, PCEP3 and the other two implementations of PCEP converged into two

and a single non-dominated point(s) respectively (see Figure 2). Hence, it proves that the PCEP systems were not able to fully exploit the range of good solutions between the first and second players offered by the Pareto evolutionary optimization process. Furthermore, although it may appear that the other systems' solutions dominate the canonical PEP solutions, post-evolution empirical evaluation proves otherwise (see Section VII-D).

B. Co-evolution with an Archive

Figure 4 shows details of the overall performances of implementations from PCEP-A competing as the first and second players against three different levels of evaluation players. Overall, PCEP-A1 was the best implementation from PCEP-A, outperforming PCEP-A2 and PCEP-A3. Selected agents from PCEP-A1 were able to perform very well when competing against the expert-level player. Furthermore, the best agent from PCEP-A1 never lost any game as the first and second player. However, the best second-player agent from PCEP-A2 lost 3 games as the second player and the best second-player agent from PCEP-A3 lost even more games as the second player (6 games).

Using the medium-level evaluator, representatives from PCEP-A1 successfully outperformed representatives from the other setups. They have the lowest minimum number of losses as the second player (9 games) and the lowest maximum number of losses as first player (8 games). Similarly, agents from PCEP-A3 have the lowest minimum number of losses as the second player. However, PCEP-A2 did not perform well, since its agents have the highest minimum number of losses as the second player (12 games). Comparing against the random player, PCEP-A1 again had the best performance compared with the other two implementations of PCEP-A. Representatives from PCEP-A1 have the lowest number of minimum losses as the second player (as well as PCEP-A2, 2 games) and the lowest maximum number of losses as the first player (5 games). The maximum number of losses as the second player of PCEP-A3's agent(s) was slightly lower (13 games), whereas the maximum number of losses as the second player of PCEP-A1's agents and PCEP-A2's agent(s) were both 15 games.

The PCEP-A has an embedded archive, that stores Pareto solution(s) of every 50th generation. At the 800^{th} generation, the number of agents stored in the archive can approach 50. The addition of an archive is to have a better quality of evaluation in terms of play strength representation and fairness by having the comparison against a similar set of opponents. Overall, PCEP-A1 was the best implementation of PCEP-A, outperforming the other two PCEP-A systems. PCEP-A2 and PCEP-A3 were initialized with a larger number of randomly picked opponents (50 and 70 opponents, respectively). Hence, PCEP-A2 and PCEP-A3 may still be randomly picking opponents from the population of current

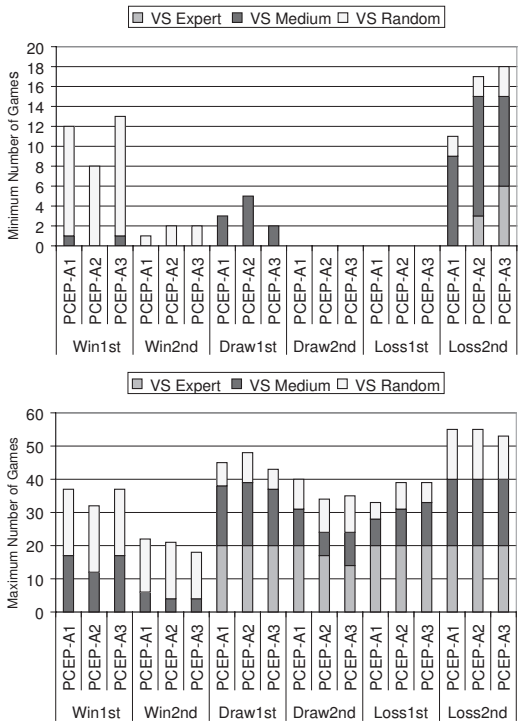


Fig. 4. Selected ANNs from implementations of PCEP-A competing against a near-perfect expert player (*VS Expert*), an average player (*VS Medium*) and a random player (*VS Random*). This figure shows the minimum (*Min*) and maximum (*Max*) number of games won (*Win1st, Win2nd*), drawn*Draw1st, Draw2nd*) and lost (*Loss1st, Loss2nd*) as the first and second players, respectively.

generation for comparison until the 800^{th} generation. However, since PCEP-A1 was initialized with a smaller number of comparisons (30 opponents), the archive will contain more non-dominated opponents compared to randomly picked opponents much earlier in the evolutionary run compared to the other two systems with larger archives. Thus, the most important observation in the success of the evolution of PCEP-A1 is that the number of randomly picked opponents was lower. The number of randomly picked opponents of PCEP-A1 was almost zero after the 500^{th} generation.

C. Performance With/Without the Additional Archive

PCEP2 and PCEP-A1 were the best systems among implementations of its own algorithm. For further discussion to look into the effects of the additional archive, the discussion continues with the comparison between PCEP-A1 and PCEP2. Since the performance of PCEP2 and PCEP-A1 were the same in the competition with the expert-level player, Figure 5 only shows the details of the performances of PCEP2 and PCEP-A1 competing against average and random players. The PCEP-A1 was performing better than PCEP2 in competing against the random player. The number of maximum losses as the first player of PCEP-A1 (5 games)

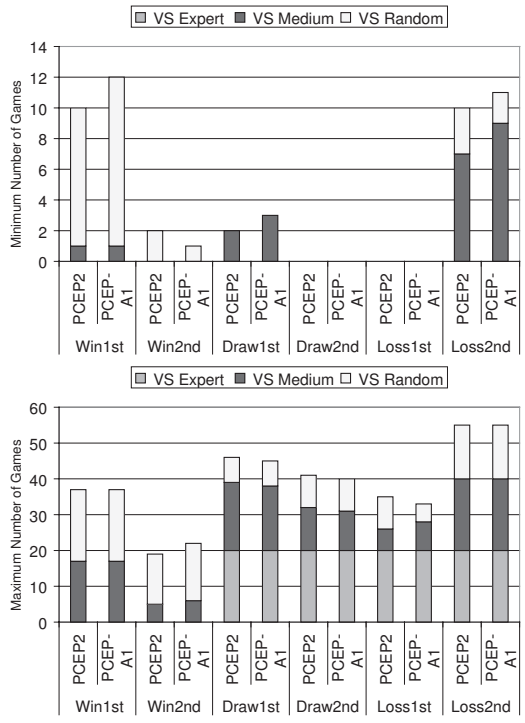


Fig. 5. Minimum (*Min*) and maximum (*Max*) number of games won (*Win1st, Win2nd*), drawn (*Draw1st, Draw2nd*) and lost (*Loss1st, Loss2nd*) as the first and second players, respectively from ANNs of PCEP2 and PCEP-A1

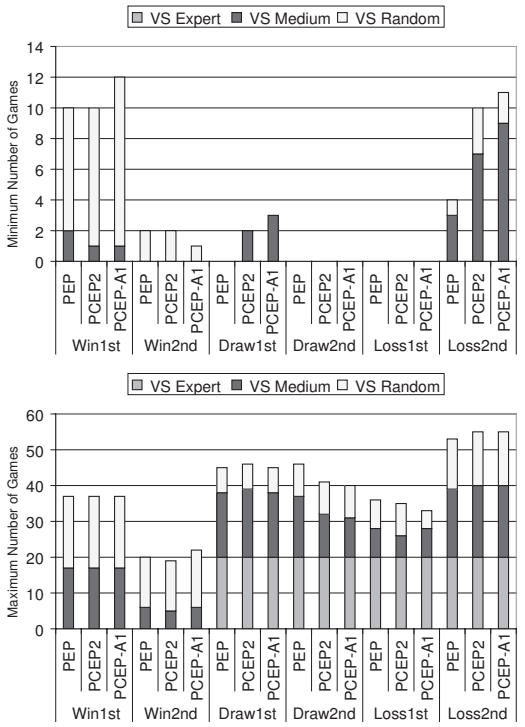


Fig. 6. Minimum (*Min*) and maximum (*Max*) number of games won (*Win1st, Win2nd*), drawn (*Draw1st, Draw2nd*) and lost (*Loss1st, Loss2nd*) as the first and second players, respectively from ANNs of PEP, PCEP2 and PCEP-A1

was smaller than PCEP2 (9 games). Furthermore for the competition against the random player, PCEP-A1 also has a slightly higher minimum number of wins and maximum number of wins as the first and second players, respectively. However for the competition against the medium-level player, PCEP2 performed slightly better than PCEP-A1.

The discussion continues with the focus on the suitability of both algorithms in terms of Pareto multi-objective optimization. Figure 1 clearly shows the suitability of PCEP-A1 was significantly higher than the PCEP2. Implementations of PCEP-A (especially PCEP-A1) have a significantly better spread of global non-dominated solutions compared to implementations of PCEP. Marking of dominated solutions in PCEP-A was based on the number of wins obtained from evaluation against agent(s) from the archive (as well as randomly picked agent(s) from population of the current generation, only if the size of the archive is less than the required minimum number of comparisons). This evaluation method is thus presenting more than the "performance of the current generation", which is more global (over the whole process of co-evolution) and less elements of luck, since the evaluation is using the same set of Pareto solutions. PCEP-A1 was randomly picking agent(s) from the population before the 500th generation approximately. However, for the rest of the evolution, PCEP-A1 was evaluated using agents from the archive only, since the size of the archive had already exceeded the minimum number of comparisons.

D. Performance without Co-evolution

Figure 6 shows the overall performances of PEP, PCEP2 and PCEP-A1 competing against three different levels of evaluation players. PCEP2 and PCEP-A1 were the best systems among implementations of its own algorithm. Overall, PEP was successful in outperforming PCEP2 and PCEP-A1 in terms of producing good performing agents as both the first player and second player. Compared to agents from PCEP2 and PCEP-A1, agent(s) from PEP were performing better as the second player, since they have the lowest minimum and maximum number of losses when competing against the medium and random players as the second player. However, compared to representatives of PEP, agents of PCEP2 and PCEP-A1 performed slightly better as the first player when competing against the medium-level player and random player, respectively.

Furthermore, PEP uses the fitness values from the payoff functions directly to mark dominated solutions, which is more suitable in Pareto multi-objective optimization as justified by the results of the comprehensive testing above. The

PEP has the best spread of global non-dominated solutions on the Pareto frontier compared with frontiers from other systems (see Figure 2).

VIII. Conclusion

This paper reports the first comprehensive study in evolving cognitive systems for game AI in board games using Pareto evolution as well as co-evolutionary techniques. Overall, the canonical PEP system was able to automatically synthesize neural network game-playing agents successfully both as the first and second players without the introduction of co-evolution. Furthermore, PEP was successful in outperforming all the other systems in terms of producing high play-strength game agents as the second player. Moreover, using an archive in PCEP-A did not produce significantly better results as first expected. Using an evaluation value which represents a bigger picture (not the current generation only) and is less dependent on luck for marking of dominated solutions should introduce significant effects on the performance of the Pareto multi-objective optimization process. The poor performance of the PCEP systems, even those utilizing an archive, in producing a good spread of solutions along the Pareto front is further proof that co-evolutionary methods are not particularly beneficial for synthesizing intelligent agents for game AI in Pareto evolution, at least for games like TTT where good rule-based players are readily available. In terms of playing strength, PCEP2 was successful in outperforming PCEP1 and PCEP3, similarly PCEP-A1 was the best system compared with PCEP-A2 and PCEP-A3. Only PEP, PCEP2 and PCEP-A1 were successful in outperforming all other systems in terms of producing high play-strength game agents that never lost any game to the expert-level player both as the first and second players. Furthermore, the significant difference in results between PCEP2 and PCEP-A1 is the evaluation against the medium and random players. Agent(s) of PCEP2 were performing better against the medium level evaluator, but agent(s) of PCEP-A1 were performing better against the random evaluator. Based on the performances of PCEP and PCEP-A systems, the co-evolution process was very sensitive to the number of randomly picked opponents. Initializing the co-evolution process with a suitably small number to limit the number of randomly picked opponents early enough in the evolutionary process can cause significant effects on the performance of co-evolutionary process.

Lastly, it is also shown clearly here that all the implementations of the co-evolutionary algorithms faced one problem, that is the "forgetting" problem. Although the majority of agents can win or draw against an expert-level player, they "forgot" how to win or draw against medium-level and random players. This is one of the known problems in co-evolutionary techniques. This problem is particularly obvious when the focus is on the synthesis of intelligent agents that act as the second player. As such, this should be investigated further in future work and may be utilized at the same time as a suitable test bed for evaluating new methods proposed for solving the "forgetting" problem in co-evolutionary algorithms. For future work, we will also investigate scalability of the algorithm by extending to more complex game such as Othello, the game of Go, and Checkers.

Acknowledgments

The authors wish to thank MIMOS Berhad for providing time on their computational cluster to conduct the evolutionary runs. In particular, we wish to thank J.Y. Luke, H. Kauthary and W.K Ng for their time and assistance. We are also grateful to the anonymous reviewers for their helpful comments.

References

[1] C. A. Coello Coello, G. Toscano Pulido, and E. Mezura Montes, "Current and future research trends in evolutionary multiobjective optimization," in *Information Processing with Evolutionary Algorithms: From Industrial Applications to Academic Speculations*, M. G. na, R. Duro, A. d'Anjou, and P. P. Wang, Eds. Springer-Verlag, 2005, pp. 213–231.

[2] J. E. Davis and G. Kendall, "An investigation, using co-evolution, to evolve an Awari player," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02)*. IEEE Press, 2002, pp. 1408–1413.

[3] G. A. Monroy, K. O. Stanley, and R. Miikkulainen, "Coevolution of neural networks using a layered Pareto archive," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06)*. ACM Press, 2006, pp. 329–336.

[4] L. Panait and S. Luke, "A comparison of two competitive fitness functions." in *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers, 2002, pp. 503–511.

[5] J. Noble and R. A. Watson, "Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*. Morgan Kaufmann Publishers, 2001, pp. 493–500.

[6] H. A. Mayer and P. Maier, "Coevolution of neural Go players in a cultural environment," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC2005)*. 'IEEE Press, 2005, pp. 1017–1024.

[7] A. Lubberts and R. Miikkulainen, "Co-Evolving a Go-Playing Neural Network," in *Coevolution: Turning Adaptive Algorithms upon Themselves*, R. K. Belew and H. Juillè, Eds., San Francisco, California, USA, 2001, pp. 14–19.

[8] D. B. Fogel, "Using Evolutionary Programming to Create Neural Networks That Are Capable of Playing Tic-Tac-Toe," in *Proceedings IEEE International Conference on Neural Networks*. IEEE Press, 1993, pp. 875–880.

[9] K. Chellapilla and D. B. Fogel, "Evolution, Neural Networks, Games, and Intelligence," *Proceedings of IEEE*, vol. 87, no. 9, pp. 1471–1496, 1999.

[10] Y. J. Yau, J. Teo, and P. Anthony, "Pareto Evolution and Co-Evolution in Cognitive Game AI Synthesis," in *Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO2007)*, ser. LNCS, vol. 4403. Springer-Verlag, 2007, pp. 227–241.

[11] Y. J. Yau and J. Teo, "An Empirical Comparison of Non-Adaptive, Adaptive and Self-Adaptive Co-evolution for Evolving Artificial Neural Network Game Players," in *Proceedings of the 2006 IEEE Conference on Cybernetics and Intelligent System (CIS2006)*, 2006, pp. 405–410.

[12] H. A. Abbass and R. Sarker, "The Pareto differential evolution algorithm," *International Journal on Artificial Intelligence Tools*, vol. 11, no. 4, pp. 531–552, 2002.