Fitnessless Coevolution

Wojciech Jaśkowski
wjaskowski@cs.put.poznan.pl

Krzysztof Krawiec
kkrawiec@cs.put.poznan.pl

Bartosz Wieloch
bwieloch@cs.put.poznan.pl

Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60965 Poznań, Poland

ABSTRACT

We introduce *fitnessless coevolution* (FC), a novel method of comparative one-population coevolution. FC plays games between individuals to settle tournaments in the selection phase and skips the typical phase of evaluation. The selection operator applies a single-elimination tournament to a randomly drawn group of individuals, and the winner of the final round becomes the result of selection. Therefore, FC does not involve explicit fitness measure. We prove that, under a condition of transitivity of the payoff matrix, the dynamics of FC is identical to that of the traditional evolutionary algorithm. The experimental results, obtained on a diversified group of problems, demonstrate that FC is able to produce solutions that are equally good or better than solutions obtained using fitness-based one-population coevolution with different selection methods.

Categories and Subject Descriptors: I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic methods

General Terms: Algorithms

Keywords: One-population Coevolution, Selection Methods, Games

1. INTRODUCTION

Coevolutionary algorithms are variants of evolutionary computation where an individual's fitness depends on other individuals. An individual's evaluation takes place in the context of at least one other individual, and may be of cooperative or competitive nature. In the former case, individuals share benefits of the fitness they have jointly elaborated, whereas in the latter one, a gain for one individual means a loss for the other. Past research has shown that this scheme may be beneficial for some types of tasks, allowing, for instance, task decomposition (in the cooperative variant) or solving tasks for which the objective fitness function is not known or unnatural (e.g., some types of games [1, 2]).

In biology, coevolution typically refers to an interaction of two or more species. By analogy, in evolutionary computation coevolution usually implies using multiple populations. Another reason for having more than one population is the inherent asymmetry of many problems. Popular competitive examples of such models include coevolving solutions with tests and parasites with hosts. In the cooperative case, individuals usually encode components of the complete solution. In either case, there are different roles to be played by particular individuals, so they (usually) should not be recombined, hence separate populations. However, there are environments for which the mutual relations between individuals are symmetric; therefore, there is no need for multiple populations[1]. Artificial life simulations and game playing are prominent application areas that meet this setting in its competitive variant.

The idea of evolving individuals in a single population and making them compete directly with each other without an external objective fitness measure has been termed *one-population coevolution* ([10]) or *competitive fitness environment* [1, 7]. As this approach has been exploited most intensely in the context of games, in the following we refer to the nomenclature of the game theory. Let us emphasize, however, that the actual interpretation of such terms like 'game' or 'win' depends on the context of particular application and may be distant from the intuitive meanings of these words.

In one-population coevolution, playing games between individuals substitutes for the objective external fitness measure, and for some evaluation methods the individuals in the current population are the only data available to enforce the selection pressure on the evolutionary process. One example is the *round-robin tournament* that involves *all* the remaining individuals from the population and defines fitness as the average payoff of the played games. A round-robin tournament requires $n(n-1)/2$ games to be played in each generation; therefore, it is computationally infeasible even for a moderately sized population. As a remedy, Angeline and Pollack [1] proposed the *single-elimination tournament* that requires only $n-1$ games. Starting from the entire population, the players/individuals are paired, play a game, and the winners pass to the next round. The last round produces the final winner of the tournament, and the fitness of each individual is the number of games won. Finally, the *k-random opponents* method [11] lets an individual play with k opponents drawn at random from the current population and defines fitness as the average payoff of games played,

[1]It has been argued [3] that in some cases the evolution may nevertheless benefit from using multiple populations.

requiring kn games to be played per generation. This evaluation scheme was also applied by Fogel to evolve neural nets that play checkers [4].

All the aforementioned methods follow the evaluation-selection-recombination mantra. Games played in the evaluation phase determine individuals' fitnesses that are subsequently used in the selection phase. Obvious as it seems, this scheme is essentially redundant. Playing games is selective by nature, so why not use them directly for selection?

This observation led us to propose the approach termed *one-population fitnessless coevolution* (FC). FC uses games to settle tournaments in the selection phase, skipping therefore the evaluation. Technically, our selection operator applies the single-elimination tournament to a randomly drawn group of individuals, and the winner of the last (final) round becomes immediately the result of selection. Therefore, FC *does not involve explicit fitness measure* and is significantly different from most of the contributions presented in literature. A related research direction, proposed in [12], has been discontinued to our knowledge.

In the experimental part of this paper, we demonstrate that, despite being conceptually simpler than standard fitness-based coevolution, FC is able to produce excellent players without an externally provided yardstick, like a human-made strategy. We present also a theoretical result: provided the payoff matrix of the game induces the same linear order of individuals as the fitness function, the dynamics of the fitnessless coevolution is identical to that of a traditional evolutionary algorithm. This makes it possible to study FC using the same research apparatus as for the standard evolutionary methods.

2. FITNESSLESS COEVOLUTION AND ITS EQUIVALENCE TO EA

In the traditional evolutionary algorithm, all individuals are tested in the environment and receive an objective fitness value during the evaluation phase. Afterwards, the fitness values are used in the selection phase in order to breed the new generation. In the single-population coevolutionary algorithm, there is no objective fitness function, and individuals have to be compared (pairwise or in larger groups) to state which one is better. Despite this fact, the scheme of a coevolutionary algorithm is similar to the evolutionary one. Typically, an individual receives a numerical fitness value that is based on the results of games played with some other individuals. Then, the selection procedure follows, most commonly a tournament selection that takes into account only the *ordering* of individuals' fitnesses, not their specific *values*. Thus, the outcomes of the games (relations *per se*) are converted into numerical fitness values which in turn determine the relations between individuals in the selection process. In this light, assigning fitness values to individuals seems redundant, because, in the end, only relations between them matter. Nonetheless, this is the common proceeding used in past work [1, 11, 4], except for the preliminary considerations in [12].

The redundancy of the explicit numerical fitness in one-population coevolution inspired us to get rid of it in an approach termed *fitnessless coevolution* (FC), which this paper is devoted to. In FC, there is no explicit evaluation phase, and the selection pressure is implemented in the *fitnessless selection*. Fitnessless selection may be considered a variant of a single-elimination tournament applied to a randomly drawn set S of individuals of size t, which is the only parameter of the method. The selection process advances in rounds. In each round, individuals from S are paired, play a game, and the winners pass to the next round (compare description of the single-elimination tournament in Section 1). For odd-sized tournaments, the odd individual plays a game with one of the winners of the round. In case of a game ending with a draw, the game winner is selected at random. This process continues until the last round produces the final winner of the tournament, which becomes also the result of selection. In particular, for $t = 2$, the winner of the only game is selected. The fitnessless selection operator is applied n times to produce the new population of size n, so the total number of games per generation amounts to a reasonable $(t-1)n$.

It should be emphasized that the term 'fitnessless' is not meant to suggest the absence of selection pressure in FC. The selection pressure emerges as a side-effect of interactions between individuals, but is not expressed by explicit fitness function.

Fitnessless coevolution, as any type of coevolution, makes investigation of the dynamics of the evolutionary process difficult. Without an objective fitness, individuals stand on each others shoulders rather than climb a single 'Mount Improbable'. In particular, it is easy to note that if the game is intransitive (beating a player P does not imply the ability of beating all those beaten by P), the winner of fitnessless selection does not have to be superior to all tournament participants. To cope with problems like that, Luke and Wiegand [10] defined conditions the single-population coevolutionary algorithm must fulfill to be *dynamically equivalent* to an evolutionary algorithm, i.e., to produce the same run, including the same contents of all generations. In the following, we first shortly summarize their work, then we determine when our FC approach is dynamically equivalent to evolutionary algorithm and comment on how our result compare with Luke's and Wiegand's.

Following [10], we define the payoff matrix and the utility.

Definition 1. $A = [a_{ij}]$ is a **payoff matrix**, in which a_{ij} specifies the score awarded to strategy $\#i$ when playing against strategy $\#j$.

Definition 2. Assuming an infinite population size and complete mixing (i.e., each individual is paired with every other individual in the population including itself), aggregate subjective values for genotypes (their **utility**) can be obtained as follows:

$$\overrightarrow{u} = A\overrightarrow{x},$$

where $\vec{x}$ represents proportions of genotypes in an infinite population.

Definition 3. Given a linear transformation, $a_{ij} = \alpha f_i + \beta f_j + \gamma$, the internal subjective utility u is linearly related to an objective function f, $u \sim_L f$, if the transitive payoff matrix A is produced using this transformation.

Luke and Wiegand proved the following theorem, which says when a single-population coevolutionary algorithm exhibits evolutionary dynamics.

THEOREM 1. *A single-population coevolutionary algorithm under complete mixing and the assumption that population sizes are infinite employing a non-parametric selection method using the internal subjective utility $\overrightarrow{u} = A\overrightarrow{x}$ is dynamically equivalent to an evolutionary algorithm with the same selection method, using the objective function f, if $u \sim_L f$ as long as $\alpha > 0$ [10].*

In order to guarantee this dynamic equivalence, Luke and Wiegand had to make several assumptions about the evolutionary algorithm and the payoff matrix A: infinite populations, complete mixing, and $u \sim_L f$. In the following, we prove that FC is equivalent to an evolutionary algorithm employing tournament selection under the only condition that f has to induce the same linear order of individuals as the payoff matrix A.

THEOREM 2. *A single-population coevolutionary algorithm employing fitnessless selection (i.e., fitnessless coevolution) is dynamically equivalent to an evolutionary algorithm with tournament selection using the objective function f, if*

$$\forall_{i,j} f_i > f_j \iff a_{ij} > a_{ji}. \tag{1}$$

PROOF. We need to show that, given (1), for any set of individuals S, each act of selection out of S based on f in the evolutionary algorithm produces the same individual as the fitnessless selection applied to the same set S. Let us assume, without loss of generality, that f is being maximized. As for an arithmetic objective function f,

$$f_i \geq f_j \wedge f_j \geq f_k \Rightarrow f_i \geq f_k,$$

it is easy to show that, under (1), a similar expression must be true for A:

$$a_{ij} \geq a_{ji} \wedge a_{jk} \geq a_{kj} \Rightarrow a_{ik} \geq a_{ki}.$$

In the fitnessless coevolution, the outcome of selection is the winner of the last game of a single-elimination tournament; let w be the index of that individual. The winner's important property is that it won or drew all games it played in the tournament; since the payoff matrix A is transitive, the winner is in fact superior to *all* individuals in S. Therefore, $\forall_{i \in S}\, a_{wi} \geq a_{iw}$, and this, together with (1), implies that $\forall_{i \in S}\, f_w \geq f_i$. Thus, the winner of fitnessless selection has the maximal objective fitness among the individuals in S and would also win the tournament selection in the traditional evolutionary algorithm. In result, under (1), both selection methods produce the same individual, and the course of both algorithms is identical. □

The consequence of the above condition is following. If the payoff matrix A is transitive, there always exists an objective function f, so that the evolutionary algorithm using f as a fitness function is dynamically equivalent to fitnessless coevolution using A. Thus, we refer to condition (1) as to *transitivity condition.*

Note that fitnessless coevolution does not need to know f explicitly. To make it behave as a standard evolutionary algorithm, it is enough to know that such objective f exists. One can argue that if there exists such a function f that the transitivity condition holds, it would be better to construct it explicitly, and run a traditional evolutionary algorithm using f as a fitness function, instead of running the fitnessless coevolution. One could even avoid the explicit function f and sort the entire population using the game outcomes as a sorting criterion (comparator), and then apply a non-parametric selection (like tournament selection) using that order. In both cases, however, fulfilling condition (1) is the necessary prerequisite. As we will show in the following experiment, FC performs well even if it does not hold.

We also claim that, where possible, one should get rid of numerical fitness because of Occam's razor principle: if it is superfluous, why use it? Note also that numerical fitness may be accidentally over-interpreted by attributing to it more meaning than it actually has. For instance, one could evaluate individuals using single-elimination tournament, which produces fitness defined on an ordinal scale, and then apply a fitness-proportional selection. As the fitness-proportional selection assumes that the fitness is defined on the metric scale, its outcomes would be flawed.

3. EXPERIMENTS

In order to assess the effectiveness of our fitnessless coevolution with fitnessless selection (FLS), we compared it to the fitness-based coevolution with two selection methods: single-elimination tournament (SET) and k-random opponents (kRO). In total, we considered twelve setups (FLS, SET, and kRO for $k = 1, ..., 10$), called *architectures* in the following. We apply each architecture to two games: the Tic Tac Toe (a.k.a. Noughts and Crosses) and a variant of the Nim game. As demonstrated in the following, both of them are intransitive so no objective fitness function exists that linearly orders their strategies.

Following [11], we also apply the architectures to standard optimization benchmarks of minimizing Rosenbrock and Rastrigin functions, by casting them into a competitive form of a two-player game. Of course, for this kind of task the objective fitness exists by definition (it is the function value itself) and the game is transitive. Normally, this kind of task is solved using an ordinary fitness-based evolutionary algorithm, but casting this problem into the game domain serves here the purpose of exploring the dynamics of the fitnessless one-population coevolution. Otherwise, as shown below for Tic Tac Toe and Nim, no such problem casting is needed to apply the fitnessless coevolution to any two-player game.

Instead of designing our own genetic encoding, we followed the experimental setups from [1] (Tic Tac Toe) and [11] (the rest). All three reference architectures used tournament selection of size 2. Note that we did not limit the number of generations; rather than that, each evolutionary run stops after reaching the total of 100,000 of games played. It is a fair approach, as some selection methods need more games per generation than the others, and simulation of the game is the core component of computational cost. We performed 50 independent runs for each architecture to obtain statistically significant results.

We implemented our experiments with ECJ [8].

3.1 Tic Tac Toe

In this game, two players take turns to mark the fields in a 3x3 grid with two markers. The player who succeeds in placing three marks in line wins the game.

Tic Tac Toe does not fulfill the transitivity condition (1), which is easy to demonstrate by an example. Let us consider

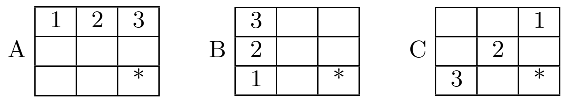

Figure 1: Three simple Tic Tac Toe strategies that violate condition (1).

a triple of trivial strategies A, B, C, shown in Fig. 1. Each of them consists in placing the marks in locations and in an order shown by the numbers when the grid cell is free, or placing the mark in the asterisk cell if the numbered cell is already occupied by the opponent. Clearly, no matter who makes the first move, strategy A beats B, as already its first move prevents B from having three marks in the leftmost column. By the same principle, B wins with C. According to transitivity condition, these two facts require the existence of f_A, f_B, f_C such that $f_A > f_B$ and $f_B > f_C$. This, in turn, implies $f_A > f_C$. However, Fig. 1 clearly shows that C beats A, which contradicts $f_A > f_C$. There is a cycle: none of these strategies outperforms the two remaining ones and their utilities cannot be mapped onto an ordinal (or numerical, in particular) scale.

Each individual-player in this experiment has the form of a single genetic programming (GP, [6]) tree, built using a function set of nine terminals and seven functions. The terminals represent the nine positions on the board (pos00, pos01, ..., pos22). All functions process and return board positions or a special value *NIL*. The binary function *And* returns the second argument if neither of them are *NIL*, and *NIL* otherwise. *Or* returns the first not-*NIL* argument or *NIL* value if both are *NIL*. *If* returns the value returned by the second argument if the first (conditional) argument is not *NIL*, otherwise the third argument. The *Mine*, *Yours* and *Open* operators test the state of a given field. They take a board position as an argument and return it if it has an appropriate state, otherwise they return *NIL*. The one-argument operator *Play-at* places player's mark on the position given by the argument if the field is empty and, importantly, stops the processing of the GP tree. If the field is busy, *Play-at* returns its argument and the processing continues.

As this function set does not guarantee making *any* move, we promote players which make some moves. Player's final score is, therefore, the number of moves made plus an additional 5 points bonus for a draw or 20 points for winning. The player with more points wins. As the player that makes the first move is more likely to win (there exist a strategy that guarantees draw), we let the players play *double-games*. A double-game consists in a pair of games, each starting with a different player. The player that wins both games in the double-game is declared the winner, otherwise there is a draw.

This experiment used maximal tree depth of 15, population size 256, crossover probability of 0.9, and replication probability of 0.1.

Following [11], we determined the best-of-run solution by running the SET on all best-of-generation[2] individuals. After carrying out 50 independent runs, we got 50 representative individuals from each architecture. To compare our twelve architectures, we let all $12 \times 50 = 600$ representative individuals play a round-robin tournament. The final evaluation of each individual was the average score against the other individuals in the tournament. The mean of these evaluations was the final architecture's score presented in the following graphs.

[2]In fitnessless approach appointing the best-of-generation individual is not obvious, so we simply choose it randomly.

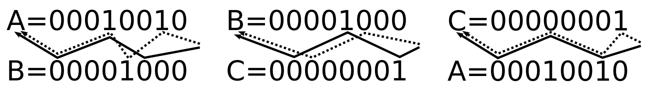

Figure 2: Three games played between three players *A*, *B*, and *C* demonstrate Nim's intransitivity. The bitstrings encode the strategies of the players. The dotted line shows the advancement of the game when the upper player makes the first move. The solid line shows the advancement of the game when the lower player makes the first move. The upper players win in all three cases, no matter who makes the first move, so none of the strategies is better that the remaining two.

3.2 Nim Game

In general, the game of Nim involves several piles of stones. Following [11], we used only one pile of 200 stones. Players take in turn one, two, or three stones. The player who takes the last stone wins.

The Nim game individual is encoded as a linear genome of 199 bits (the 200^{th} bit is not needed because 200 stones is the initial state). The ith gene (bit) says whether i stones in the pile is a *desirable game state* for the player (value 1) or not (value 0). A player can take one, two, or three stones in its turn. It takes three stones if it leads to a desirable game state (i.e., if the corresponding bit is 1). Then, it tests in the same way taking two and one stone. If all considered states are not desirable, the player takes three stones.

The outcome of the Nim game may depend on who moves first. For instance, let us consider a simplified Nim starting with just 7 stones and two strategies encoded in the way discussed above: A=001000 (meaning three stones is a desirable state, while 1, 2, 4, 5, and 6 stones are not) and B=001001 (only 3 and 6 stones are desirable). If A moves first, it takes 3 stones (as all three considered genes are 0), then B takes 1 stone (according to the third bit in its strategy), and finally A takes the last three stones and wins. However, if B moves first, it takes only one stone (due to the rightmost '1' in its genotype), A takes three stones, thus B is left with three stones to be taken and wins. Due to this property of Nim, we make our individuals face each other in a double-game, similarly to Tic Tac Toe.

Despite its simplicity, Nim is intransitive too. Let us consider three 9-stone Nim strategies A=00010010, B=00001000, and C=00000001 (as it turns out, nine stones is the minimum number required to demonstrate intransitivity). The double-game between A and B results in A's win (see Fig. 2). Thus, according to Condition (1), A should have better fitness than B: $f_A > f_B$. As B beats C, also $f_B > f_C$ should hold. However, C wins against A, requiring $f_C > f_A$. No numerical (or even ordinal) fitness can model the mutual relationships between A, B, and C.

Our experiments involved population size of 128, a 1-point crossover with probability 0.97, and mutation with probability 0.03. The architectures were compared in the same way as in Tic Tac Toe.

3.3 Rosenbrock

The Rosenbrock function has the following form for the N-dimensional case:

$$Rosenbrock(X) = \sum_{i=1}^{N-1} \left[(1 - x_i)^2 + 100 \left(x_{i+1} - x_i^2 \right)^2 \right].$$

We converted the problem of minimizing this function to a competitive counterpart by defining

$$Reward(A, B) = \frac{Rosenbrock(B) - Rosenbrock(A)}{max(Rosenbrock) - min(Rosenbrock)},$$

where $max(Rosenbrock)$ and $min(Rosenbrock)$ are the maximum and minimum values of Rosenbrock function in the considered domain; $Reward(A, B)$ determines the score (in the range $[-1, 1]$) of player A playing against the opponent B. Of course, $Reward(A, B) = -Reward(B, A)$.

In this experiment, we used genomes of $N = 100$ real values between -5.12 and 5.12 (function domain), population size of 32, a 1-point crossover, and mutation of a single gene with probability 0.005.

In the Rosenbrock problem, unlike in Tic Tac Toe and Nim games, there exists an objective and external (i.e., not used during the evolution) individual's fitness—the Rosenbrock function itself. Therefore, as the best-of-run we chose the individual that maximizes the external fitness value, defined as:

$$1 - \frac{Rosenbrock(X) - min(Rosenbrock)}{max(Rosenbrock) - min(Rosenbrock)} \qquad (2)$$

For the same reason, in the Rosenbrock problem, to compare the architectures we also used this external fitness. It should be emphasized, however, that the fitnessless run has no access to the external fitness function, which is used only for the purpose of the best-of-run selection and comparison of best-of-runs between particular runs.

3.4 Rastrigin

As the last problem, we considered minimizing the Rastrigin function, defined as:

$$Rastrigin(X) = A \cdot N + \sum_{i=1}^{N} \left[x_i^2 - A \cdot \cos(2\pi x_i) \right],$$

where $A = 10$ and $N = 100$. The Rastrigin minimization problem was converted to a competitive problem in the same way as the Rosenbrock function. Also, the setup of the experiment and comparison between architectures was identical to Rosenbrock's.

4. RESULTS

Figures 3 to 14 compare the architectures of FLS, SET and kRO for k ranging from 1 to 10. These charts present the average external fitness of the best-of-run individuals from each architecture.

As we can see in Fig. 3, FLS was hardly better than the other architectures at playing Tic Tac Toe and slightly worse than SET at evolving the Nim player (Fig. 6). On the other hand, in problems that fulfill the transitivity condition (Fig. 9 and 12), the FLS architecture was clearly better than SET and kRO, which is especially visible in case of Rastrigin function. More precisely, FLS is statistically better than kRO for all values of k on Nim, Rosenbrock, and Rastrigin; for Tic Tac Toe, it beats kRO for 8 out of 10 values of k (t-Student, $p = .01$). When compared to SET, FLS is significantly better than it on Rosenbrock and Rastrigin and worse on Nim; for Tic Tac Toe, the test is inconclusive. Table 1 summarizes the outcomes of the statistical comparison of FLS to kRO and SET.

Following [9], we tested also how the noisy data influences evolution. We introduced noise by reversing the game outcome (thus swapping players' rewards) with a given probability. For instance, adding 100% noise would aim at evolving the worst possible player. Figures and Table 1 show the effect of adding 30% and 40% noise. Note that the presence of noise renders *all* four problems intransitive.

It seems that FLS is less affected by noise than SET. In the hierarchical process of SET, each distorted game impacts the subsequent rounds. Even the (objectively) best-of-generation individual may be dropped behind due to noise. FLS turns out to be more resistant to noise, as the random reversal of game outcome influences only one selection act. Therefore, SET slightly outperforms FLS, though insignificantly, only in case of high noise in the Nim game (Fig. 8).

In the overall picture, kRO shows the ability to attain the best resistance to noise among all the considered architectures: it performs at least as good or better than FLS and SET for some values of k, especially for the highest noise level considered (40%). However, the optimal value of k varies across the noise levels and problems and is difficult to tell in advance. In general, higher values of k compensate for the presence of noise, but also shorten the evolutionary run by increasing the required number of games in each generation. FLS almost always offers a statistically equivalent or better performance and thus may be considered as an attractive option.

5. CONCLUSIONS

In this paper we proposed a fitnessless selection scheme dedicated to one-population coevolution. We also proved that an evolutionary process employing that scheme is equivalent to the fitness-based coevolution provided the fulfillment of transitivity condition (1).

The presented experimental results demonstrate that fitnessless coevolution is competitive to single-elimination tournament and the k-random opponents method, especially when the task fulfills the transitivity condition. Though this constraint may be difficult to meet globally in the entire domain of the problem, we hypothesize that effectiveness of FLS increases with the *extent* of transitivity (meant as, e.g., the probability that transitivity holds for a pair of individuals randomly drawn from a population). However, this phenomena may be more complex and depend, e.g., on the *structure* of transitivity as well, so this supposition requires verification in a separate study.

The mechanism of FLS is elegant and simple in at least two ways: in getting rid of the numerical fitness and in combining the evaluation and selection phase. Despite this simplicity, it produces effective solutions and is immune to noise to an extent that is comparable to kRO (assuming the optimal value of the k parameter for kRO is known in advance). In a separate study [5], we demonstrated its ability to evolve human-competitive players in a complex game with partially observable states. The downside of the method is the extra effort required to appoint the best-of-run individual.

One could argue that, no matter whether the objective

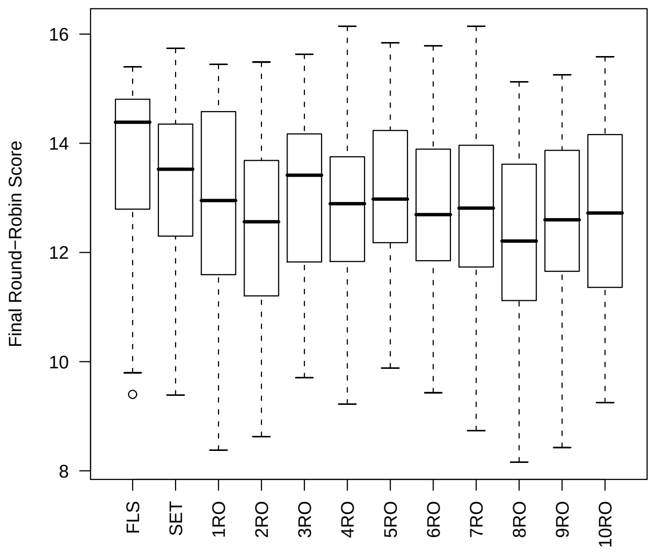

Figure 3: Tic Tac Toe with 0% noise

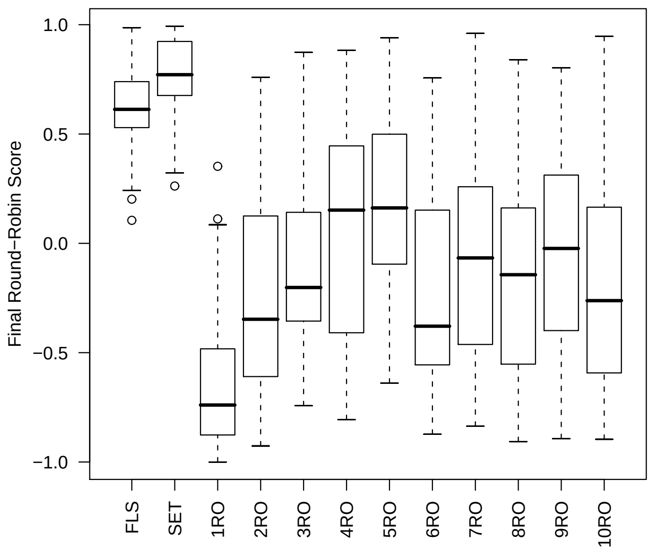

Figure 6: Nim with 0% noise

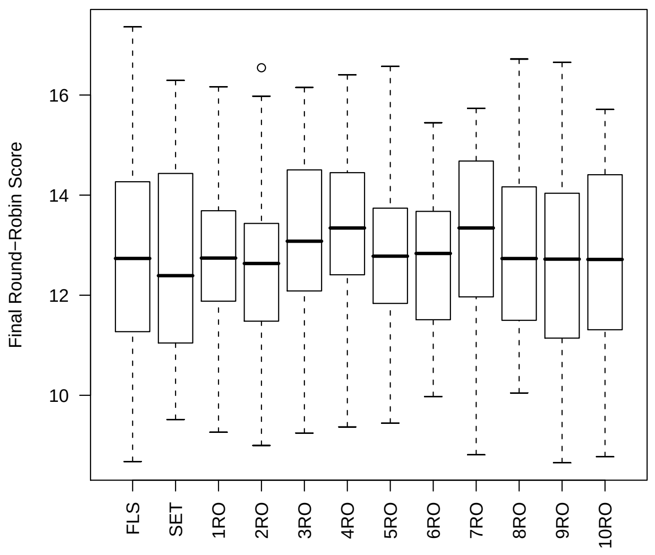

Figure 4: Tic Tac Toe with 30% noise

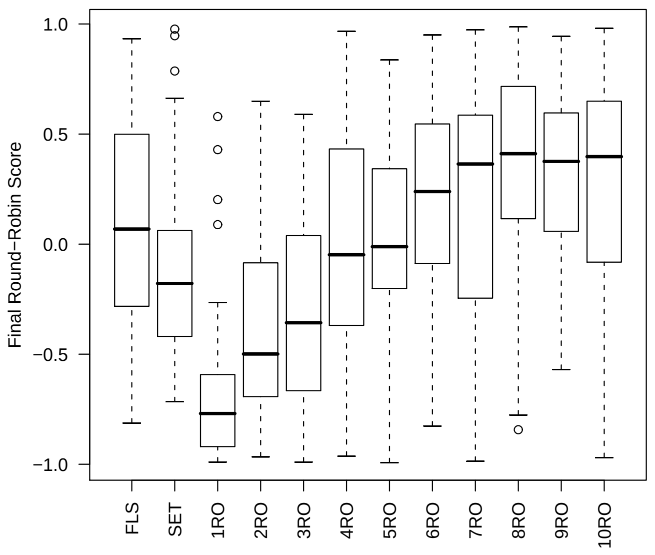

Figure 7: Nim with 30% noise

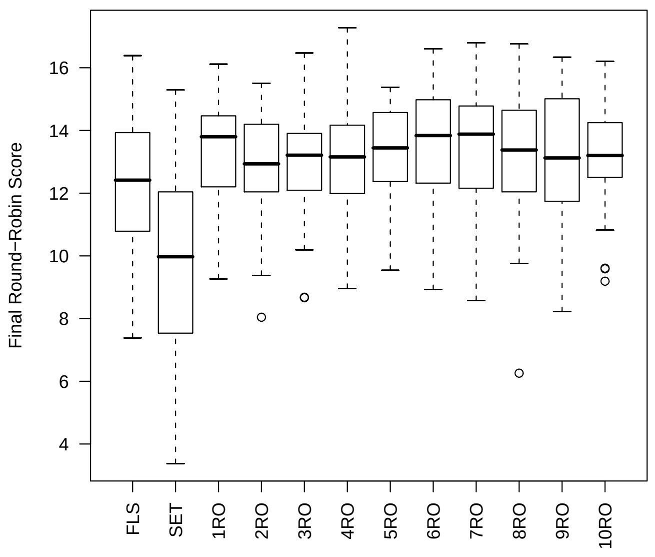

Figure 5: Tic Tac Toe with 40% noise

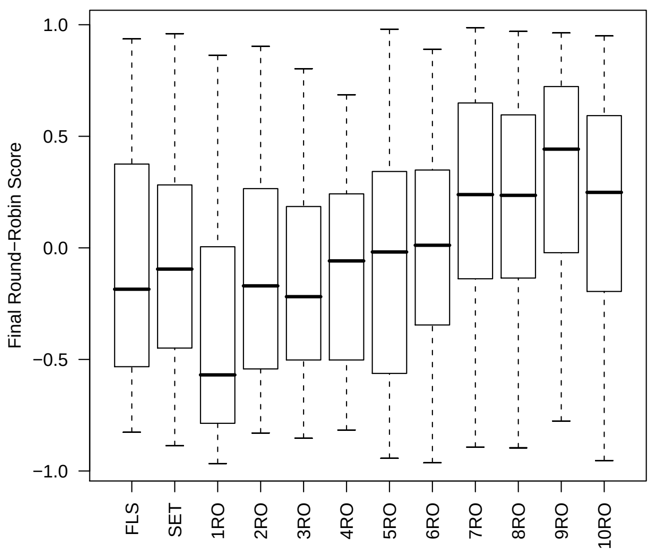

Figure 8: Nim with 40% noise

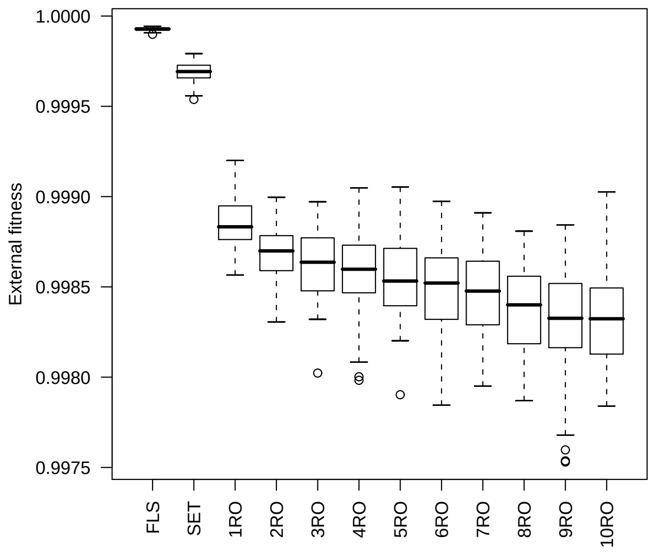

Figure 9: Rosenbrock with 0% noise

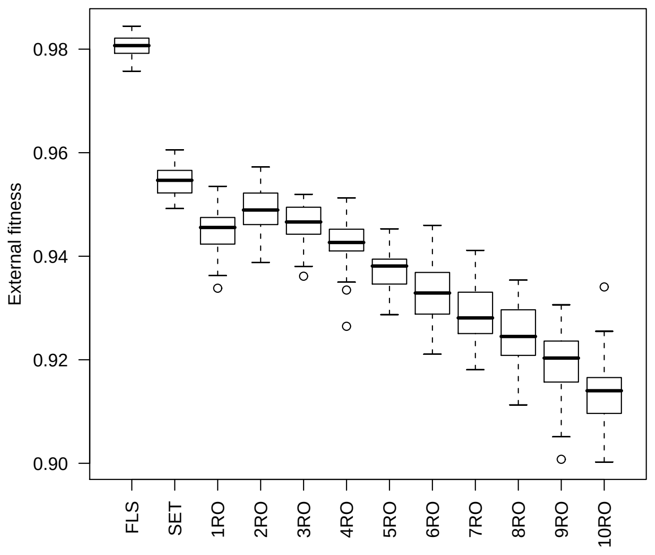

Figure 12: Rastrigin with 0% noise

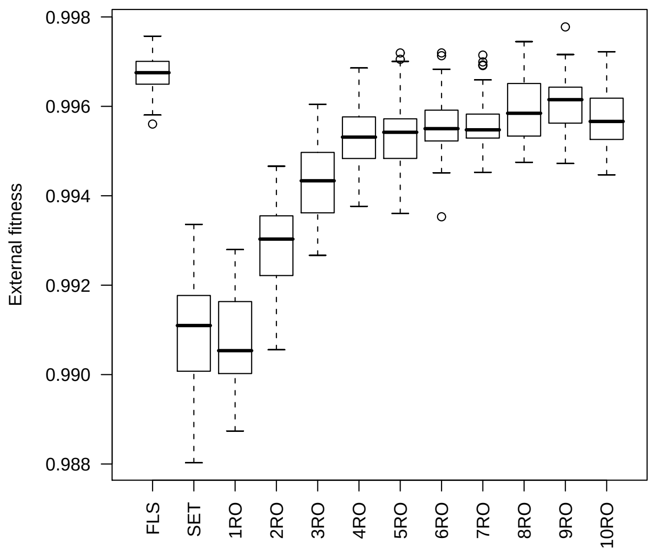

Figure 10: Rosenbrock with 30% noise

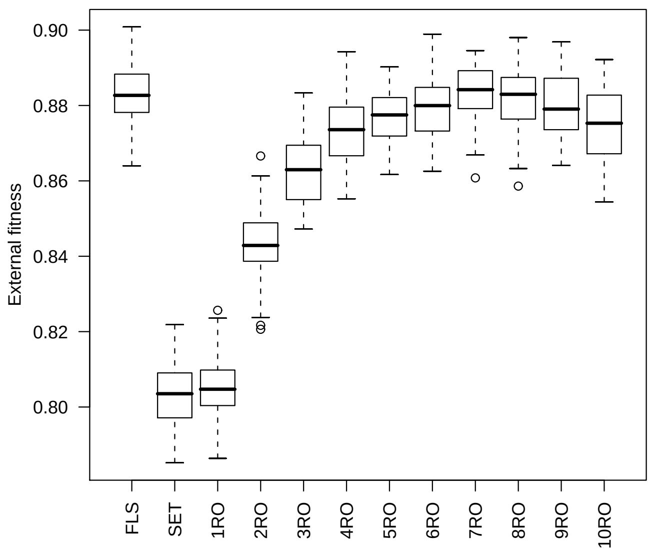

Figure 13: Rastrigin with 30% noise

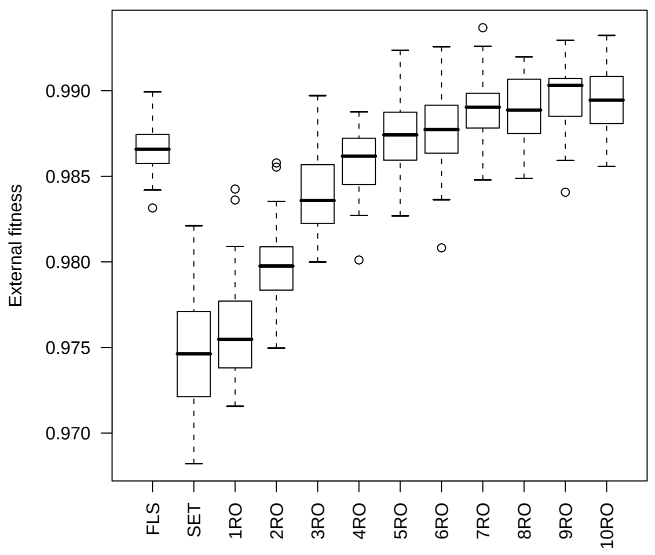

Figure 11: Rosenbrock with 40% noise

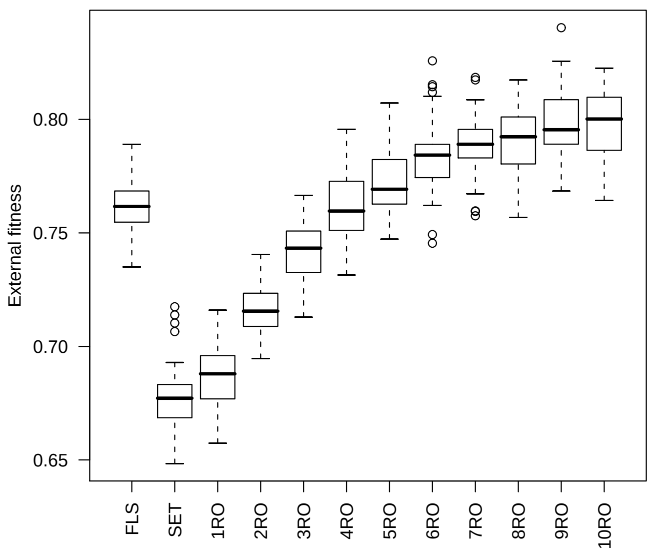

Figure 14: Rastrigin with 40% noise

Table 1: The outcomes of pairwise statistical comparison of FLS vs. kRO and SET (significance level 0.01). Symbols ¡, =, and ¿ denote respectively FLS being worse, equally good, and better than the other method. For kRO, figures tell how many times FLS was in particular relation to kRO.

FLS vs.	Tic Tac Toe				Nim				Rosenbrock				Rastrigin			
	kRO			SET	kRO			SET	kRO			SET	kRO			SET
Noise	$<$	$=$	$>$		$<$	$=$	$>$		$<$	$=$	$>$		$<$	$=$	$>$	
0%		2	8	$=$			10	$<$			10	$>$			10	$>$
30%		10		$=$	2	5	3	$=$			10	$>$		4	6	$>$
40%	4	6		$>$	3	6	1	$=$	6		4	$>$	6	1	3	$>$
Total	4	18	8		5	11	14		6		24		6	5	19	

function exist, does not exist, or is difficult to define, there is always *some* way of estimating the numerical fitness, so there is no need of such fitnessless approach. Indeed, SET and kRO are examples of such ways. Note however how arbitrary they are. Fitnessless selection, on the contrary, is conceptually simpler and requires little assumptions.

Another attractive property of fitnessless coevolution is its locality with respect to the population. SET requires simultaneous access to all individuals in the population. FLS, on the contrary, works on the same, usually small, subset of individuals when performing both evaluation and selection. This may have positive impact on the performance in case of using a parallel implementation, and may be nicely combined with other evolutionary techniques that involve locality, like the island model or spatially distributed populations.

Fitnessless coevolution has also the virtue of being more natural. Similarly to biological evolution, the success of an individual depends here *directly* on its competition with other individuals. Also, the fitness function used in standard evolutionary algorithm is essentially a mere technical means to impose selective pressure on the evolving population, whereas its biological counterpart (fitness) is defined *a posteriori* as probability of survival. By eliminating the numerical fitness, we avoid subjectivity that its definition is prone to.

6. ACKNOWLEDGMENTS

The authors wish to thank the reviewers for valuable feedback on this work. This research has been supported by the Ministry of Science and Higher Education grant # N N519 3505 33.

7. REFERENCES

[1] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 264–270, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[2] Y. Azaria and M. Sipper. GP-gammon: Genetically programming backgammon players. *Genetic Programming and Evolvable Machines*, 6(3):283–300, Sept. 2005. Published online: 12 August 2005.

[3] A. Bucci. *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*. PhD thesis, Brandeis University, 2007.

[4] D. B. Fogel. *Blondie24: playing at the edge of AI*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[5] W. Jaśkowski, K. Krawiec, and B. Wieloch. Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection. In M. O'Neill, L. Vanneschi, S. Gustafson, A. I. E. Alcázar, I. D. Falco, A. D. Cioppa, and E. Tarantino, editors, *Genetic Programming*, volume 4971 of *LNCS*, pages 13–24. Springer, 2008. LNCS49710013.

[6] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[7] S. Luke. Genetic programming produced competitive soccer softbot teams for robocup97. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 214–222, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[8] S. Luke. ECJ evolutionary computation system, 2002. (http://cs.gmu.edu/ eclab/projects/ecj/).

[9] S. Luke and R. Wiegand. Guaranteeing coevolutionary objective measures. *Poli et al.[201]*, pages 237–251.

[10] S. Luke and R. Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 236–241, 2002.

[11] L. Panait and S. Luke. A comparison of two competitive fitness functions. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 503–511, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[12] A. G. B. Tettamanzi. Genetic programming without fitness. In J. R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996*, pages 193–195, Stanford University, CA, USA, 28–31 July 1996. Stanford Bookstore.