Discovery of Implicit Objectives by Compression of Interaction Matrix in Test-Based Problems

Paweł Liskowski and Krzysztof Krawiec

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
{pliskowski,krawiec}@cs.put.poznan.pl

Abstract. In test-based problems, commonly solved with competitive coevolution algorithms, candidate solutions (e.g., game strategies) are evaluated by interacting with tests (e.g., opponents). As the number of tests is typically large, it is expensive to calculate the exact value of objective function, and one has to elicit a useful training signal (search gradient) from the outcomes of a limited number of interactions between these coevolving entities. Averaging of interaction outcomes, typically used to that aim, ignores the fact that solutions often have to master different and unrelated *skills*, which form underlying objectives of the problem. We propose a method for on-line discovery of such objectives via heuristic compression of interaction outcomes. The compressed matrix implicitly defines derived search objectives that can be used by traditional multiobjective search techniques (NSGA-II in this study). When applied to the challenging variant of multi-choice Iterated Prisoner's Dilemma problem, the proposed approach outperforms conventional two-population coevolution in a statistically significant way.

Keywords: Test-based problems, coevolution, iterated prisoner dilemma, multiobjective evolutionary algorithms.

1 Introduction

Test-based problems are search and optimization tasks where candidate solutions are being evaluated by confronting them with *tests*. A single *interaction* between a candidate solution and a test produces a scalar outcome that reflects the capability of the former to *pass* the latter (expressed in the simplest case as a binary value). Canonical examples of test-based problems are games, where candidate solutions and tests are game strategies, while interactions boil down to playing games between them.

Solving a test-based problem consists in finding a candidate solution with certain properties. In the most common case, it should maximize the *expected utility*, i.e., the average outcome against all tests. Finding such a solution is challenging in many test-based problems, because the number of tests is usually large, and for some problems even infinite. This problem can be mitigated by estimating a solution's utility by confronting it with a *sample* of tests of a computationally manageable size. Some evolutionary algorithms (e.g., [3]) implement this idea by maintaining a population of candidate solutions and assessing their fitness on a sample of tests generated at random. Competitive coevolution algorithms (e.g., [14]) rely on tests that dwell either in the same population (for one-population coevolution), or in a separate, coevolving population of tests (for two-population coevolution). Hybrid approaches of coevolution with random sampling have also been studied [12].

At first sight, averaging interaction outcomes over the available tests seems natural, as the fitness obtained in this way approximates the expected utility, i.e., the ultimate objective of the search process. If the test sample is drawn at random, that approximation is even unbiased. On the other hand, such aggregation inevitably incurs information loss. The outcomes of interactions may compensate each other, so that candidate solutions can receive the same fitness (and thus be indiscernible for selection), even if they fare very differently with particular tests. It becomes thus natural to ask: do we have to 'compress' all the information about interactions outcomes into one scalar value? Why not exploit it more carefully, for the sake of making search more efficient?

Several studies in the past have investigated the possibility of scrutinizing individual interaction outcomes and leveraging them for better performance of an evolutionary process. Bucci [2] and de Jong [6] introduced *coordinate systems* that compress the interaction outcomes into a multidimensional structure. Technically, given the dominance relation as defined by interaction outcomes (where every test is treated as a separate objective), a coordinate system can be constructed that preserves this relation while typically featuring a lower number of derived objectives (dimensions). Interestingly, every such objective can be said to express a certain *skill* exhibited by the candidate solutions.

Unfortunately, even with a moderately large number of tests, it is unlikely for any candidate solution to dominate (in the above sense) any other candidate solution in the population. From such a sparse dominance relation, it is hard to elicit any information that would efficiently drive the search process. In this paper we propose a heuristic method that compresses the original interaction outcomes into a few derived objectives in a 'lossy' manner. The method does not guarantee to preserve the original dominance structure, but always produces a low number of derived objectives that approximately capture the skills exhibited by the candidate solutions. The experiments with two-population coevolutionary algorithm demonstrate that the objectives obtained in this way can be better 'search drivers' than the conventional, averaging fitness function.

2 Background

Consider a test-based problem $(\mathbb{S}, \mathbb{T}, g)$, where $\mathbb{S}$ is the set of all candidate solutions, $\mathbb{T}$ is the set of all tests, and $g : \mathbb{S} \times \mathbb{T} \to [0,1]$ is interaction function that characterizes solution's capability to pass t. In particular, $g(s,t) = 1$ means that s passed test t and $g(s,t) = 0$ is interpreted as s failing t. Our goal is to solve this problem by finding a candidate solution that maximizes the expected utility, i.e., $s^* = \arg\max_{s \in \mathbb{S}} \sum_{t \in \mathbb{T}} g(s,t)$.

We approach the test-based problems using two-population coevolutionary algorithms, with a population of candidate solutions $S \subset \mathbb{S}$ of size m and a population of n tests $T \subset \mathbb{T}$. Evolving candidate solutions and tests in two separate populations has been shown superior to one-population configuration [13], as it allows them to specialize in their roles, where candidate solutions focus on maximizing the search objective, while tests are responsible for creating a *learning gradient* for them.

In the evaluation phase of evolutionary workflow, we apply g to $S \times T$ and obtain an $m \times n$ interaction matrix G, where rows correspond to candidate solutions and columns correspond to tests. G implicitly defines a dominance relation $\succ$ between the elements of S: $s_i \succ s_k \iff \forall j\, g_{i,j} \geq g_{k,j} \land \exists j : g_{i,j} > g_{k,j}$, where $g_{i,j}$ denotes the outcome of interaction between the ith candidate solution and jth test, $i = 1, \ldots, n$, $j = 1, \ldots, m$. This dominance relation, built on a limited number of individuals in S and T, is transient and never reveals their full characteristics. We assume that G is the only information available at the given generation (S and T do not feature *archives*).

To perform selection of candidate solutions in S, i.e., to determine a subset $S' \subset S$ of 'promising' candidate solutions, a coevolutionary algorithm has to elicit information from G. Though there is a gamut of possible elicitation methods, we first consider the following two extremes.

1. The **direct approach** could consist in employing the original dominance relation $\succ$ defined by G to carry out the selection process. Technically, one would assume that, for every test $t \in T$, the outcomes of interactions with t determines performance on the associated objective. In theory, this should enable applying conventional multiobjective evolutionary methods, like NSGA-II [7].

The advantage of the direct approach is that it relies on all available and undistorted information on interaction outcomes. The downside is that the likelihood of any solution in S dominating any other is low even for a moderate number of tests in T, and becomes even lower when the tests become diversified (which we want them to be). When the dominance relation becomes sparse, solutions are often incomparable, and there is no information to base the selection process on. Also, the conventional multiobjective evolutionary computation algorithms are known to perform well only if the number of objectives is moderate; while the population of tests in a typical coevolutionary setup hosts usually not a few, but at least a few dozens of tests.

2. **Scalarization.** As the other extreme, the information contained in G can be scalarized by aggregating the interaction outcomes over all tests available in T and adopting the resulting quantity as solutions' fitness:

$$f(s) = \sum_{t \in T} g(s,t) \tag{1}$$

The fitness defined in this way can be subsequently used to run an ordinary selection stage (e.g., tournament selection). The advantage of this approach is that f (when normalized) is an estimator of expected utility, which is our external objective of the search process, so an algorithm's 'search driver' is consistent with the ultimate search goal. On the downside, aggregation incurs information loss, and many pairs of solutions that were originally incomparable can receive similar, if not identical, fitness (the latter case being particularly likely if the underlying interaction function assumes only a few values, which is common in test-based problems).

In this study, we are interested in combining the advantages of these approaches, i.e., to preserve *some* information on dominance while avoiding aggregating interaction outcomes into a single scalar value. To this aim, we will 'compress' the original information in G into a few *derived objectives*. In a similar spirit, several past studies on competitive coevolution [2,6,11] proposed formal methods for deriving coordinate systems (CS) from interaction matrices. However, they all implement an exact approach, i.e., the spatial arrangement of solutions in the CS exactly reproduces the original dominance structure. If that structure was sparse, such was also the structure of the derived CS (which typically manifested in the CS having many dimensions). As a result, CSs defined in this way are interesting tools for studying interaction outcomes, but not necessarily useful 'search drivers'. The approach we propose in the next section, by compressing the interaction outcomes in a lossy manner, guarantees to result in a few, albeit inexact, derived objectives.

3 Objective Compression Algorithm

Given an $m \times n$ interaction matrix G, the algorithm proceeds in two stages:

1. Clustering of tests. We treat every column of G, i.e., a vector of interaction outcomes of all solutions from S with the specific test t, as a point in an m-dimensional space. A clustering algorithm of choice is applied to the n points obtained in this way. We employ k-means, as it is conceptually straightforward and typically converges in a few iterations. With k being the number of clusters, the outcome of this step is the clustering/partitioning $\{T_1, \dots, T_k\}$ of the original n tests (columns in G) into k subsets.

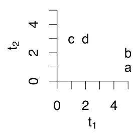
2. Defining derived objectives. For each cluster T_c, we derive from it a new search objective by averaging the corresponding columns in G row-wise. The resulting vector is the centroid of the cluster T_c. The overall outcome is an $m \times k$ *derived interaction matrix* G', where the columns correspond to the new *derived objectives*, while the rows correspond to candidate solutions in S.

The resulting derived objectives can be subsequently used to guide the selection process, e.g., employed as objectives in the NSGA-II algorithm, as in the experimental part of this paper.

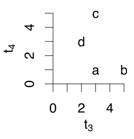
Example. Consider the matrix of interactions between the population of candidate solutions $S = \{a, b, c, d\}$ and the population of tests $T = \{t_1, t_2, t_3, t_4\}$, shown in Fig. 2a. Clearly, the only dominance holding in this space is $b \succ a$. The four-dimensional space of interaction outcomes is shown in two two-dimensional plots (Figs. 2b and 2c) that span $t_1 \times t_2$ and $t_3 \times t_4$, respectively. This helps to reveal that the performances of candidate solutions on the tests t_1 and t_3 are quite correlated. An analogous observation holds for t_2 and t_4. Assume the

G	t_1	t_2	t_3	t_4
a	5	1	3	1
b	5	2	5	1
c	1	3	3	5
d	2	3	2	3

(a)



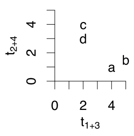
(b)



(c)

G'	t_{1+3}	t_{2+4}
a	4	1
b	5	1.5
c	2	4
d	2	3

(d)



(e)

Fig. 1. Example of compression of interaction matrix (a) featuring a four-dimensional dominance structure (b, c), into a derived interaction matrix (d), resulting with the dominance structure shown in (e)

clustering algorithm decided to cluster t_1 with t_3 and t_2 with t_4. The centroids of the clusters (Fig. 2d) form the derived objectives for this problem.

In the space of derived objectives (Fig. 2e) b still dominates a. However, now also c dominates d, thought originally these two solutions were incomparable. As a result of compressing the original interaction matrix and merging dimensions, some information about the dominance structure has been lost.

In the particular case of c and d, introducing dominance in favor of c seems reasonable, as c outperforms d on two original objectives (t_3, t_4), while only one objective (t_1) supports the opposite relation (t_2 being neutral in this respect). However, the clustering of interaction matrix columns in general does not provide this kind of guarantees. In this sense, the above transformation trades the lower number of resulting objectives for certain inconsistency with the original interaction outcomes. Nevertheless, we posit that this imprecision may be a price worth paying for reducing the number of objectives and eliciting a potentially useful learning gradient from them.

Properties of the Method. The objective compression algorithm is allowed to distort the original dominance represented in the interaction matrix G. Though this can be considered a downside, note that the information in G does not fully characterize the candidate solutions in the first place, because of the limited number of tests available in T. In other words, it may not make sense to perfectly preserve the information in G if it is partial anyway (and not necessarily useful to drive the search process, as we discussed in Section 2). For the same reasons, we do not find it critical that the k-means algorithm employed here is a heuristic, and may produce different derived objecives depending on the initial random partitioning of original objectives.

The method features a few other properties. Trivially, clustering guarantees including any pair of original objectives that are mutually redundant (i.e., identical columns in G) into the same derived objective. Moreover, the more two tests are similar in terms of solutions' performance on them, the more likely they will end up in the same cluster and contribute to the same derived objective. The clustering discovers thus certain *skills* of candidate solutions, as revealed in G.

For $k = 1$, the method degenerates to a single-objective approach (case #2 in Section 2): all tests form one cluster, and G' has a single column that contains solutions' (normalized) estimate of expected utility defined in (1).

Setting $k = n$ implies $G' = G$, and the method implements the direct approach (case #1 in Section 2).

Finally, the derived objectives are additive components of scalar fitness (Eq. 1), i.e., $\sum_{j=1}^{k} g'_{i,j} = f(s_j)$, where the jth row in G' corresponds to s_j.

4 Experimental Verification

In the following we apply the proposed approach to the Iterated Prisoner's Dilemma (IPD), an abstract game that elegantly embodies the problem of achieving mutual cooperation in social, economic and biological interactions. The computational experiment is aimed at verification whether the objective compression algorithm influences the efficiency of coevolutionary learning.

Problem Definition. IPD is a two-player game involving a series of interactions, each of which is a Prisoner's Dilemma (PD) game. In a PD, a player can make one of two choices: cooperate or defect. If both players cooperate, they receive a payoff R, whereas if they both defect they get a smaller payoff P. Defecting against a cooperator gives a payoff T which is higher than R, and the cooperator in such a case receives the lowest possible payoff S. The PD payoff matrix must satisfy two conditions: $T > R > P > S$ and $2R > S + T$ [15].

In this study, we consider IPD extended to multiple choices (or *levels of cooperation*) [9,10,5,4] with payoff function that meets the above constraints:

$$p(c_A, c_B) = \begin{cases} 2.5 - 0.5c_A + 2c_B & \text{for player } A \\ 2.5 - 0.5c_B + 2c_A & \text{for player } B \end{cases}.$$

An example of a payoff matrix for the 3-choice Prisoner's Dilemma generated using the above function is shown in Table 1, the possible choices being $\{-1, 0, 1\}$.

Strategy Representation. We adopt the direct look-up table [1] to represent the strategies (candidate solutions and tests) and consider the *memory-one* form of IPD in which the players remember their moves from the previous iteration only. In such a case, the n-choice IPD strategy is an $n \times n$ matrix M, where m_{ij} for $i, j = 1, 2, \ldots, n$ specifies the choice to be made given the player's previous move i and the opponent's previous move j. The other element of the strategy is the initial move m_{00}.

In the experiments, we focus on IPD with $n = 9$ choices, which we found to be much more demanding than 3-choice IPD used in some earlier coevolutionary investigations [3]. Each strategy is a look-up table containing $9 \times 9 + 1 = 82$ choices and the size of search space is $9^{82} \approx 1.77 \times 10^{78}$.

Experimental Setup. We embed the objective compression algorithm in a typical two-population competitive coevolutionary setting (Section 2). The resulting $m \times k$ matrix of derived objectives forms the input for the NSGA-II algorithm [7] where they guide the selection process of candidate solutions. The role of NSGA-II is to maintain a diverse front of well-performing candidate solutions

Table 1. An exemplary payoff matrix for the 3-choice prisoner's dilemma. Choice −1 is full defection, 1 is full cooperation. (p_A, p_B) denotes payoffs to players A and B, respectively.

		Player B		
	choice	1	0	−1
	1	(4, 4)	(2, 4.5)	(0, 5)
Player A	0	(4.5, 2)	(2.5, 2.5)	(0.5, 3)
	−1	(5, 0)	(3, 0.5)	(1, 1)

by Pareto-ranking the population and resolving ties on selection by means of crowding distance.

In the population of tests, individuals are rewarded for making *distinctions* [8] between candidate solutions. This fitness function promotes tests that differentiate the candidate solutions and thus provide *search gradient* for them.

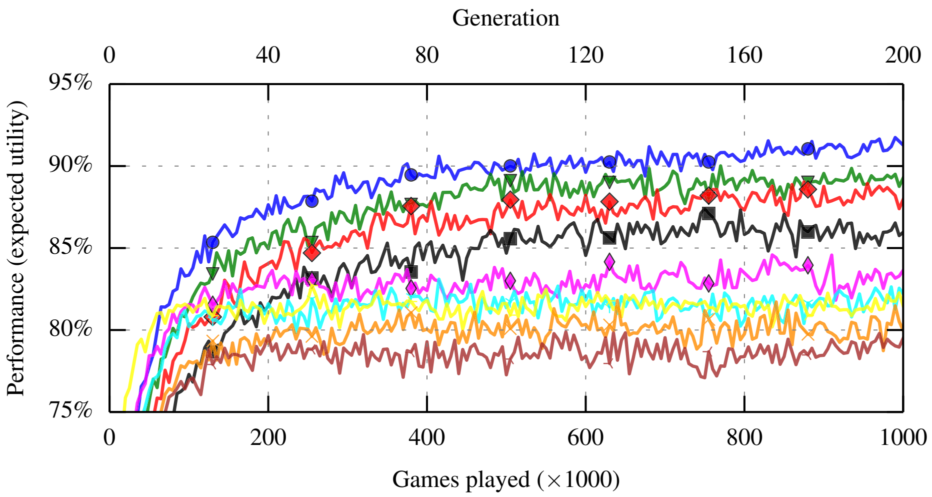
The objective compression algorithm (k-MEANS in the following) is examined in the presence of two control setups. In the first one, we replace the k-means algorithm with a naive approach to clustering in which individuals are assigned to clusters randomly (k-RAND). This configuration is intended to control for the relevance of clusters discovered by clustering. The second setup is an ordinary two-population coevolutionary algorithm (CEL, [14]), which is equivalent to 1-MEANS (see the discussion of algorithm properties at the end of Section 3).

All algorithms maintain populations of 50 candidate solutions and 50 tests. However, because NSGA-II effectively merges parents and offspring prior to selection (and in this sense features an internal archive), we set the candidate solutions population size to 100 for CEL. This provides for fair comparison between the methods. As a result, in each generation of every method, $100 \times 50 = 5{,}000$ IPD games are played, each of which consists of 150 PD episodes. Since each run consists of 200 generations, it requires the total effort of 1,000,000 games.

Both populations use tournament selection with tournament size 5. The only source of genetic variation is simple mutation which iterates over all elements of the look-up table and with probability 0.2 replaces the original choice with one of the remaining choices, selected at random. This operator has been found to provide sufficient variation of behaviors for multiple-choice IPD [4].

Results. Algorithms that solve test-based problems do not rely on an objective performance measure, so a candidate solution deemed good by an algorithm does not have to be such in reality. In other words, fitness as defined in an algorithm (if any) is *subjective* (internal) and may strongly differ from the true performance of a candidate solution. To *objectively* (externally) assess the performance of a candidate solution, we estimate its expected utility by letting it play 50,000 games against random opponents. Every random opponent is obtained independently by filling the look-up table with random choices. This assessment, commonly used with test-based problems, is external to a search algorithm and does not affect its behavior. Below we report algorithm performance meant in this way.

We performed separate experiments for $k = 2 \ldots 5$ clusters. Let us first discuss the search dynamics, illustrated in Fig. 2a, which reports the objective performance of the best-of-generation candidate solutions, averaged over 120



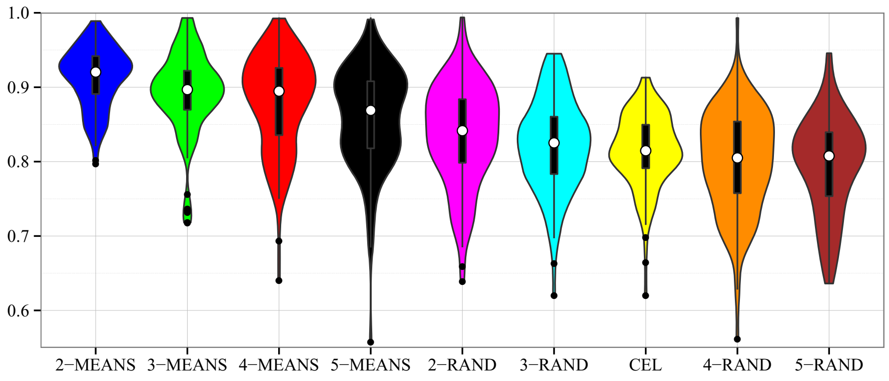


Fig. 2. The expected utility of best-of-generation individuals averaged over 120 runs (top) and distributions of expected utilities of the best-of-runs individuals (bottom). Violin plot legend: white dot: median, black box: interquartile range, line: 1.5 interquartile range, black dots: outliers. Both plots use the same colors.

coevolutionary runs. For reference, we plot the best-of-generation players found by standard coevolutionary search (CEL). Clearly, compression of the interaction outcomes allows k-MEANS outperform CEL for every k. In particular, $k = 2$ seems to be optimal, suggesting that this may be the required number of skills to effectively play the 9-choice IPD. We speculate that the two discovered skills correspond to full defection and full cooperation in IPD, since defecting is the dominant strategy of the game, while mutual cooperation pays off the most in the long term. This hypothesis requires, however, verification in the future.

Interestingly, we also observe certain positive influence of decomposing the scalar fitness function by *random* clustering (RAND). In this case however, improvement over CEL is noticeable only for $k = 2$ (2-RAND). Apparently, merging randomly selected tests into derived objectives cannot efficiently capture meaningful

Table 2. Expected utilities and 95% confidence intervals of best-of-run individuals obtained by the algorithms for 9-choice Iterated Prisoner's Dilemma

Method	Expected utility	Method	Expected utility
CEL	81.48 ± 0.90		
2-MEANS	91.28 ± 0.73	2-RAND	83.64 ± 1.21
3-MEANS	89.27 ± 0.99	3-RAND	82.21 ± 1.11
4-MEANS	87.96 ± 1.16	4-RAND	79.99 ± 1.23
5-MEANS	85.96 ± 1.20	5-RAND	79.49 ± 1.17

skills that would effectively guide the multiobjective learning process towards improving the population of candidate solutions.

When it comes to comparing algorithms' end-of-run outcome, Table 2 presents the average performance of the best-of-run individuals for each algorithm, accompanied by 95% confidence intervals, while Fig. 2b the distributions as violin plots. To compare these final performances of the algorithms, we applied Shapiro-Wilk test, which indicated the performance distributions to be likely non-normal (all p-values $< 10^{-6}$). We then conducted the nonparametric Kruskal-Wallis rank sum test, which revealed a statistically significant ($\chi^2 = 158.7$, p-value $< 2.2 \times 10^{-16}$) difference between the results obtained by particular algorithms. A post-hoc analysis using pairwise Wilcoxon rank sum test with Holm correction indicated the following ranking among the configurations:

2-MEANS > 3-MEANS > 4-MEANS = 5-MEANS>2-RAND = 3-RAND = 4-RAND=5-RAND=CEL

where '>' denotes significant difference and '=' means no statistical difference. This ranking corroborates our earlier observations: meaningful grouping of tests (and associated original objectives) and using the resulting derived objectives in multiobjective setting makes coevolutionary search more effective.

5 Conclusions

In this paper we proposed a heuristic method that compresses the original interaction outcomes into a few derived, implicit objectives in a lossy manner. Even though the method does not guarantee to preserve the original dominance structure, it succesfully maganges to produce a low number of objectives that approximately capture the skills exhibited by the candidate solutions. Crucially, our method avoids aggregating interaction outcomes into a scalar value, therefore allowing multiobjective approach to the problem. We demonstrated how this compression, combined with the NSGA-II algorithm, can be applied to effectively enhance the coevolutionary search.

The experiments with two-population coevolutionary algorithm demonstrate that the implicit objectives discovered in interaction outcomes are indeed better search drivers than the conventional, averaging fitness function. Our results support the claim that it is enough to preserve only some information on dominance to obtain a useful learning gradient. Applicability of this approach to other interactive and non-interactive domains is to be verified in future research.

Acknowledgments. P. Liskowski acknowledges support from grant no. DEC-2013/09/D/ST6/03932 and K. Krawiec acknowledges support from grant no. DEC-2011/01/B/ST6/07318.

References

1. Axelrod, R.: The evolution of strategies in the iterated prisoner's dilemma. The Dynamics of Norms, 1–16 (1987)
2. Bucci, A., Pollack, J.B., de Jong, E.: Automated extraction of problem structure. In: Deb, K., Tari, Z. (eds.) GECCO 2004, Part I. LNCS, vol. 3102, pp. 501–512. Springer, Heidelberg (2004)
3. Chong, S.Y., Tino, P., Ku, D.C., Xin, Y.: Improving Generalization Performance in Co-Evolutionary Learning. IEEE Transactions on Evolutionary Computation 16(1), 70–85 (2012)
4. Chong, S.Y., Yao, X.: Behavioral diversity, choices and noise in the iterated prisoner's dilemma. IEEE Transactions on Evolutionary Computation 9(6), 540–551 (2005)
5. Darwen, P.J., Yao, X.: Why more choices cause less cooperation in iterated prisoner's dilemma. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 987–994. IEEE (2001)
6. de Jong, E.D., Bucci, A.: DECA: Dimension extracting coevolutionary algorithm. In: Cattolico, M.C., et al. (eds.) GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, Washington, USA, pp. 313–320. ACM Press (2006)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. Ficici, S.G., Pollack, J.B.: Pareto optimality in coevolutionary learning. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 316–325. Springer, Heidelberg (2001)
9. Frean, M.: The evolution of degrees of cooperation. Journal of Theoretical Biology 182(4), 549–559 (1996)
10. Harrald, P.G., Fogel, D.B.: Evolving continuous behaviors in the iterated prisoner's dilemma. Biosystems 37(1), 135–145 (1996)
11. Jaśkowski, W., Krawiec, K.: Formal analysis, hardness, and algorithms for extracting internal structure of test-based problems. Evolutionary Computation 19(4), 639–671 (2011)
12. Jaśkowski, W., Liskowski, P., Szubert, M., Krawiec, K.: Improving coevolution by random sampling. In: Blum, C. (ed.) GECCO 2013: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, pp. 1141–1148. ACM (2013)
13. Juillé, H., Pollack, J.B.: Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. University of Wisconsin, pp. 519–527. Morgan Kaufmann (1998)
14. Popovici, E., Bucci, A., Wiegand, R.P., de Jong, E.D.: Coevolutionary Principles. In: Handbook of Natural Computing. Springer (2011)
15. Poundstone, W.: Prisoner's Dilemma: John von Neuman, Game Theory, and the Puzzle of the Bomb. Doubleday, New York (1992)