ISSN: 1803-3814

MENDEL Soft Computing

OPENING THE BLACK BOX: ALTERNATIVE SEARCH DRIVERS FOR GENETIC PROGRAMMING AND TEST-BASED PROBLEMS

Krzysztof Krawiec

Institute of Computing Science
Poznan University of Technology
Piotrowo 2, 60-965 Poznań
Poland
krawiec@cs.put.poznan.pl

Abstract: *Test-based problems are search and optimization problems in which candidate solutions interact with multiple tests (examples, fitness cases, environments) in order to be evaluated. The approach conventionally adopted in most search and optimization algorithms involves aggregating the interaction outcomes into a scalar objective. However, passing different tests may require unrelated 'skills' that candidate solutions may vary on. Scalar fitness is inherently incapable of capturing such differences and leaves a search algorithm largely uninformed about the diverse qualities of individual candidate solutions. In this paper, we discuss the implications of this fact and present a range of methods that avoid scalarization by turning the outcomes of interactions between programs and tests into 'search drivers' – partial, heuristic, transient pseudo-objectives that form multifaceted characterizations of candidate solutions. We demonstrate the feasibility of this approach by reviewing the experimental evidence from past work, confront it with related research endeavors, and embed it into a broader context of behavioral program synthesis.*

Keywords: *Evolutionary computation, test-based problems, genetic programming, search drivers, coevolutionary algorithms, surrogate fitness*

1 Introduction

A great deal of research in artificial and computational intelligence revolves around *iterative problem solving*, because many problems of practical interest lack an analytical method or a closed-form formula for solving them in a single step. In such cases, building a preliminary, imperfect (e.g. random) *candidate solution*, and modifying it iteratively in expectation of gradual improvement, remains the only viable approach.

The traditional take on iterative problem solving involves division of methods into *gradient-based techniques* and *generate-and-test methods*. The defining feature of the former is the existence of an update rule that allows translating the evaluation of the current candidate solution into *directed* updates of parameters. Such a rule often has good theoretical grounding and, crucially, can be derived for all parameters of candidate solutions. Training neural networks via error backpropagation and many reinforcement learning algorithms belong to this category. Gradient-based algorithms excel at exploiting the local neighborhoods of solution space in close-to-optimal way, but may be not particularly good at at its large-scale exploration.

On the other extreme there are the generate-and-test algorithms, where it is assumed that a directional update rule (or its computationally efficient implementation) does not exist. Such algorithms use the feedback from evaluation process only to decide *which* candidate solutions are worth considering in next iterations, not *in which way* they should be modified. The poster child of this family are arguably evolutionary algorithms, in which the actions of search operators are usually random and thus undirected in the above sense. Compared with gradient-based methods, generate-and-test algorithms have complementary characteristics in being capable of exploring the search space at its global scale; however, they are not necessarily good at exploiting it locally.

The lead motif of this position paper is that, common wisdom to the contrary, there is a lot of middle ground between the above extremes which can be exploited, particularly in the framework in evolutionary computation (EC). Evaluating a candidate solution does not have to result in a single scalar: for many domains, means exist for opening the black box of evaluation, making it more informative, and exploiting the additional information for the sake of search efficiency. To bring arguments for this hypothesis, in Section 2 we identify the sources and implications of the *evaluation bottleneck* on the pathway between evaluation process and search operators. In Section 3 we briefly present concrete algorithms that open that bottleneck for the, strikingly common in practice yet largely not recognized, class of *test-based problems*. In Section 4 we discuss the implications of these opportunities, introduce the concept of *search driver*, and put our considerations in a broader perspective, pointing to research directions that seem to be particularly promising.

2 The evaluation bottleneck problem

Most of the ongoing research in EC builds upon the classical formulation of optimization problem: given a space S of *candidate solutions* and an *objective function* $f : S \to \mathbb{O}$ (where $\mathbb{O}$ is some ordered domain, typically $\mathbb{R}$), find an $s \in S$ that maximizes/minimizes f. Besides being succinct and elegant, this formulation is also practical in allowing to capture the desired characteristics of sought optimal solution by designing an appropriate objective function f. In other words, f is the means of expressing designer's *intent* concerning the expected features of the result of a search process.

What we however seem to overlook quite often is that this is essentially the *only* role of objective function in this setting: to characterize the *expected outcome* of problem solving. In most cases an objective function is not meant to indicate *how* a given problem should be solved or, in particular, how a search process should be *guided*. Yet virtually all heuristic search algorithms adopt objective function as a *search guidance*, using it as the only source of information in making decisions about individual candidate solutions.

Arguably, relying on objective function is sometimes justified by limited insight into domain of consideration. This is perhaps most evident in black-box optimization, where search algorithm's agnosticism about the function being optimized is the principal tenet. However, entirely black-boxed domains are few and far between in real world. It is actually very hard to point out to a domain where, nothing can be said about a candidate solution in addition to scoring it with a scalar evaluation.

As a result, an objective function, primarily intended to succinctly capture solution's characteristics that are relevant from experimenter's perspective, is employed as a primary (and often the only one) *search driver*. The implication of this practice is *evaluation bottleneck* [11]: a candidate solution, which is often an intricate combinatorial structure or a complex entity comprising numerous variables, is 'compressed' into a single number. This forces the search algorithm to 'reverse-engineer' that information in order to make the right choices concerning the search process. A sheer juxtaposition in terms of information contents reveals that the perspectives for conducting efficient search in this regime must be bleak when no additional assumptions about the nature of the problem can be made. For instance, synthesizing a 4-bit multiplexer, a basic benchmark in genetic programming (GP), requires synthesizing a candidate solution (logical expression) that implements one of the $2^{2^6} = 2^{64} \approx 1.8 \times 10^{19}$ 6-input Boolean functions (4 data bits plus 2 address bits). To completely characterize program's behavior in this domain, 64 bits are necessary. Yet, in common practice those 64 bits are compressed by an objective function that counts the number of correct outputs, so that fitness ranges from 0 to 64 inclusive and conveys thus only 6 bits of information.

Therefore, each evaluation earns the search algorithm mere 6 bits of information, and based on that it is expected to make decisions about entities that require an order of magnitude more information to describe their behavior. Generate-and-test search in exponentially large search spaces using such snippets of information cannot be efficient in general, and this disproportion becomes only more grave for more complex problems[1].

The question of practical relevance is: does it make sense to adhere to the scalar evaluation template and impose the evaluation bottleneck on oneself, particularly in domains where the only benefit obtained in return is the elegance of problem's formulation? We argue that forcing a search algorithm to rely exclusively on the feedback from scalar evaluation hampers its performance and is as such such be abandoned wherever possible. In the next section, we briefly present a range of approaches that facilitate opening the evaluation black box for the specific class of test-based problems.

3 Opening the evaluation bottleneck for test-based problems

In this section, we illustrate the opportunities for widening the evaluation bottleneck for *test-based problems* (TBPs), which have been originally proposed in studies on coevolutionary algorithms [1, 3]. In TBPs, it is assumed that the objective function $f : S \to \mathbb{O}$ is based on a set T of *tests*, i.e. entities that the candidate solutions from S can *interact* with. For instance, when applying the TBP perspective to a machine learning problem, S is the space of hypotheses (classifiers, regression models, etc.) and T is the set of available training examples; when learning a game-playing strategy for a two-person game, S hosts all possible strategies of the first player and T is the set of opponent strategies; when approaching the problem of learning a controller, S is the set of all possible controller designs, while T is the set of *environments* for those controllers (e.g., initial/boundary conditions used to start controller's simulation with).

The interactions between candidate solutions and tests are implemented by an *interaction function* $g : S \times T \to \mathbb{O}$. In the machine learning domain mentioned above, one would assume that $g(s,t)$ returns 1 if a classifier s classifies an example t correctly, and 0 otherwise. For s and t being game strategies, it is typically assumed that $g(s,t)$ returns the payoff for the first player. For control problems, $g(s,t)$ could be the error of

[1] Note that even for this tiny problem, the actual search space is much larger than 2^{64}, because many syntactically different programs may exhibit the same output behavior (semantics).

G	t_1	t_2	t_3	t_4	t_5
a	1	1	0	1	1
b	0	1	0	1	0
c	1	0	1	1	0
d	0	1	0	0	0

a) Interaction matrix G

G	t_1	t_2	t_3	t_4	t_5
a	1	1	0	1	1
b	0	1	0	1	0
c	1	0	1	1	0
d	0	1	0	0	0

b) G after clustering

G'	t_{1+3}	t_{2+4+5}
a	0.5	1
b	0	0.66
c	1	0.33
d	0	0.33

c) Derived objectives G'

Figure 1: Discovery of Objectives by Clustering (DOC): The original matrix G (a) of interactions between candidate solutions ($a \dots d$) and tests ($t_1 \dots t_5$) is clustered column-wise (b), and the centroids of clusters become *derived objectives* (c), to be used in multiobjective selection process.

controller s simulated in the environment t, calculated with respect to some desired output. Depending on domain, interaction outcomes can be thus discrete or continuous.

The theory of TBPs developed within coevolutionary algorithms points to several *solution concepts*, i.e. the characteristics of candidate solutions (or subsets of candidate solutions) that one is looking for [16]. The solution concept that is arguably most common in practice is *maximization of expected utility* (MEU), in which the goal is to identify $s \in S$ such that maximizes (or in general optimizes) the expected interaction outcome g:

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

Therefore, MEU corresponds to the objective functions that one typically employs when solving a test-based optimization problem with EC, e.g., classification accuracy on T in machine learning problems, expected game outcome in game playing, or aggregated error[2] in control problems. In this sense the formalism of TBPs does not bring anything new to EC practice[3]. However, it helps us identifying the evaluation bottleneck (Section 2), which is due to f aggregating the individual interaction outcomes $g(s,t)$ into a single scalar value. This has several consequences that we argue to be detrimental. Two candidate solutions s_1 and s_2 with different outcomes of interactions with tests from T can be granted the same fitness $f(s_1) = f(s_2)$, rendering them indiscernible for selection operator – in which case it will most likely make a random choice, even if one of them is more desirable in some sense. If $f(s_1) < f(s_2)$, s_1 can be discarded in selection phase, even though it might have passed the test(s) that no other candidate solution manages to pass in the current population, and thus might possess a unique skill that may be worth preserving in the generations to follow.

In our past studies on these topics [9, 12, 15, 13], we proposed a range of methods that address these problems by avoiding aggregating interaction outcomes into scalar evaluations. The methods can be conveniently explained using the concept of *interaction matrix* G that gathers the interaction outcomes $g(s,t)$ between the candidate solutions (corresponding to rows in G) and the tests (corresponding to columns in G). The fitness of a given candidate solution is simply the sum of the corresponding row in G. In a generative evolutionary algorithm, G is built anew in each generation, based on the solutions in the current population and the set of tests T. We usually assume that the tests are given as a part of problem formulation (e.g., the set of training examples in machine learning) and thus remain fixed throughout a search process – unless we allow them to *coevolve* with solutions [12, 15].

In Discovery of Objectives by Clustering (DOC) [9], we seek for the common 'motifs' in behaviors of candidate solutions on tests. To that aim, G is clustered column-wise using a rudimentary clustering algorithm (Fig. 1). The centroids of clusters are then gathered in a *derived interaction matrix* G', where columns form multiple *derived objectives* that characterize solutions' behaviors on the tests gathered in particular clusters. We typically aim at a low number of objectives, which can be achieved by parameterizing the clustering algorithm accordingly. The derived objectives identified in this way can be subsequently used in a multiobjective selection procedure like Non-dominated Sorting Genetic Algorithm (NSGA) [4]. As multiobjective selection methods avoid direct aggregation of objectives and rely on a Pareto-ranking of candidate solutions, we obtain a variant of evolutionary algorithm that naturally maintains and promotes multiple 'skills' in population, which in turn improves diversity and lowers the likelihood of premature convergence. The derived objectives are extracted independently in each evolutionary generation, so they adapt to the dynamically changing characteristics of the population.

Empirical evaluation in [12, 15] and [9] confirmed the usefulness of DOC. In the former works, we applied it to well-known TBPs: numbers games (abstract two-player game), density classification task (synthesizing

[2] Where the sign of the error would have to be obviously negated in order to align with MEU being *maximized*.

[3] The central problem tackled in TBPs is that T is often very large or infinite, so (3) cannot be directly computed and must be estimated. Coevolutionary algorithms perform iterative, 'active' sampling of T. However, this characteristics of TBPs is largely irrelevant for our considerations here.

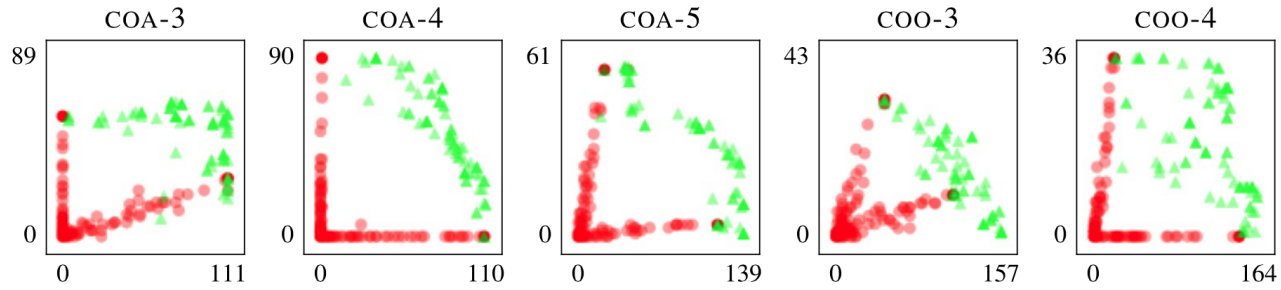

Figure 2: Candidate solutions from the last generation of an evolutionary run (green marks) tend to spread across the Pareto front that spans the two derived objectives automatically designed by DOC (for complete figure and explanation see Figs. 8 and 9 in [15]).

one-dimensional cellular automata that measure the density of 1s in the initial state) and multi-choice Iterated Prisoner's dilemma. DOC was combined with a simple two-population competitive algorithm, with the second population hosting a dynamically coevolving sample of tests from T. The interaction outcomes in G were clustered using standard k-means algorithm employing the Euclidean distance for measuring the similarity of interaction outcomes; the number of clusters k was set manually. DOC systematically outperformed the conventional single-objective approach, and analysis showed that the multiobjective perspective facilitated by DOC is essential: evolutionary runs that performed well usually had candidate solutions distributed across the Pareto-front in the space of derived objectives (Fig. 2 and Figs. 8 and 9 in [15]).

In [9], we combined DOC with genetic programming (GP) to evolve programs (expression trees) in discrete domains. In this case, GP operated as a conventional single-population evolutionary algorithm, i.e. the set of tests T was fixed and given as a part of task formulation. Each test $t \in T$ was a tuple of program input and corresponding desired output. The interaction function $g(s,t)$ granted a program s with a unit reward for passing a test t, i.e. producing the expected output for the given input. To avoid fixing the number of DOC's clusters k to an arbitrary value, this time we employed X-means, a variant of k-means that autonomously estimates k using the Bayesian Information Criterion. Also in this GP domain, DOC turned out to outperform standard GP regarding the likelihood of finding a correct program (*success rate*).

Clustering of tests is just one possible way of eliciting a multi-faceted characterization of candidate solutions from an interaction matrix. The Discovery of Objectives via Factorization (DOF) we proposed in [13] employs an approach that became recently particularly popular for designing recommender systems with machine learning [7]. The interaction matrix G is decomposed with *non-negative matrix factorization* (NMF) into a product of matrices W and H, where the number of columns in W and the number of rows in H must be obviously equal, and is controlled by method's parameter k. The k columns in W form *factors* that capture abstract characteristics of groups of tests: w_{ij} is the value of jth factor for the ith candidate solution in the current population. Because the direction of preference in the original interaction matrix G is positive (higher interaction outcomes are more desirable) and the factorization is non-negative, higher values of w_{ij} are also preferred, and w_{ij} can be interpreted as a *score* that the ith candidate solution achieved on the jth factor. Similarly to DOC, DOF treats the factors in W as derived objectives and passes them to a multiobjective selection method (NSGA-II). k controls the number of factors and the degree of 'compression' imposed on G in the process. If $k \geq rank(G)$, $WH = G$ and the factors completely preserve the dominance relation stored in G; in practice however $k \ll rank(G)$ so that the number of derived objectives can be handled by NSGA-II (it becomes problematic to elicit useful selection gradient in presence of many objectives, when the dominance relation defined by G becomes sparse).

We tested DOF on the same range of benchmark problems as DOC in [9], obtaining similarly encouraging results. The automatic 'multiobjectivization' [6] provided by DOF improved diversification in the population and facilitated co-existence of differently skilled candidate solutions in population, which in turn led to higher success rate.

Interestingly, NMF can be effectively calculated also for *sparse* matrices, only partially filled with interaction outcomes. Crucially, a factorization algorithm applied to a sparse G produces *dense* (complete) matrices W and H, with W containing the scores on all factors for *all* candidate solutions in the population. Therefore, it becomes possible to conduct only some interactions between candidate solutions and tests and still obtain derived objectives in W (though obviously those will only approximate the objectives one would obtain if G was known in full). Alternatively, W and H can be multiplied and so yield a dense matrix G', filling-in so the interaction outcomes that were originally missing in G, and enabling computation of the conventional scalar fitness (by summing G row-wise for individual candidate solutions). The main motivation here is obviously

the savings on computational cost: performing interactions, e.g., applying programs to tests in GP, is usually the most costly part of evolutionary search. The fitness function derived in this way is essentially a *surrogate fitness*. In [14], we implemented this idea in a method dubbed Surrogate Fitness via Factorization of Interaction Matrix (SFIMX), and demonstrated that it improves over the conventional GP.

4 Discussion

We hypothesize that the methods presented above only scratch the surface of possible ways of opening the evaluation bottleneck for TBPs. In particular, other means of processing interaction matrices may exist that yield derived objectives of better characteristics. The benchmark problems approached in the cited works illustrate that TBPs model a diversified range of problems of practical interest, which makes this methodology widely applicable in practice.

Moving to a broader context, relying on the TBP formulation and turning a single-objective problem into a multi-objective one is just one particular way of making a search process better informed about the candidate solutions. As every piece of information on candidate solutions can potentially make search more effective, we should be interested in extracting various characteristics of candidate solutions, including those not captured by conventional objective functions. An example of such a means used routinely in the past is building-in the preference for smaller programs into selection in GP (e.g. *lexicographic parsimony pressure* that resolves ties on fitness by preferring smaller programs). Another one is depreciation of programs that do not make use of (all) input variables.

The methods presented in Section 3 are just one of possible means of opening the evaluation black box. Several independent research efforts seem to gravitate in similar direction; examples include elementary fitness landscapes that enable principled derivation of optimal search operators that enable reaching the optimal solution in a single step for certain classes of problems [2]. For the specific class of GP problems, machine learning can be used to analyze program traces and automatically derive useful gradient from them [10].

We claim that the body of evidence gathered in the recent research on deriving underlying objectives suggests that opening the black boxes of evaluation and widening the information pathways that connect the components of metaheuristic algorithms is clearly worth pursuing. What seems to be particularly desirable is a general framing for such alternative sources of search-preferential information. The attempt of such framing, which we presented in [8], realizes the following line of reasoning. Objective functions are typically assumed to be well-defined in the entire domain (i.e., S) and to feature global optima that unequivocally indicate which candidate solutions are 'ideal'. What is more, they should induce possibly smooth fitness landscape, with no discontinuities and limited ruggedness. Designing such functions for challenging domains and problems can be difficult – GP, with the complex mapping from program code (syntax) to program behavior (semantics) to program fitness is an excellent example of this challenge. We argue that it might be reasonable to lower the bar and instead rely on *search drivers*: preference functions that are perhaps only locally defined and/or approximate, and which do not necessarily measure the entirety of desired characteristics of candidate solutions. Though imperfect, search drivers can be still helpful in making search process more effective by, for instance, not only by making population more diverse (like DOC and DOF in Section 3), but for instance by saving candidate solutions with unique skills from being discarded, or even explicitly pointing to desirable properties of candidate solutions. In [8], we elaborate on our vision of search drivers and embed them in a unified framework, arguing, among others, that multiple imperfect search drivers can be together more effective at guiding search, in a similar way that multiple weak classifiers can together form a compound classifier of superb quality. In the specific case of GP, it leads to the vision of *behavioral program synthesis*, where the detailed record of behavior of a candidate program (input, trace, output) is scrutinized to facilitate the search process (by, e.g., identifying and reusing potentially useful program fragments [10]).

Ultimately, we should aim at search and optimization methods that *automatically* exploit, whenever possible, the internal structure of the problem (or the part of problem specification that is available). One may liken this vision to the General Game Playing task coined in AI, where an algorithm is expected to devise an game strategy provided only formal game specification [5]. Recent attempts towards metaheuristic standardization [17] might pave the way towards achieving that goal.

Acknowledgement: This work was supported by the National Science Centre, Poland, grant number 2014/15/B/ST6/05205.

References

[1] Bucci, A., Pollack, J.B., de Jong, E.: Automated extraction of problem structure. In: K.D. et al. (ed.) Genetic and Evolutionary Computation – GECCO-2004, Part I, *Lecture Notes in Computer Science*, vol.

3102, pp. 501–512. Springer-Verlag, Seattle, WA, USA (2004). DOI doi:10.1007/b98643. URL http://link.springer.de/link/service/series/0558/bibs/3102/31020501.htm

[2] Burke, E., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches, chap. Handbook of Meta-Heuristics, pp. 449–468. Kluwer (2010). DOI 10.1007/978-1-4419-1665-5_15. URL http://dx.doi.org/10.1007/978-1-4419-1665-5

[3] de Jong, E.D., Pollack, J.B.: Ideal Evaluation from Coevolution. Evolutionary Computation **12**(2), 159–192 (2004)

[4] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on **6**(2), 182 –197 (2002). DOI 10.1109/4235.996017

[5] Genesereth, M.R., Love, N., Pell, B.: General game playing: Overview of the AAAI competition. AI Magazine **26**(2), 62–72 (2005)

[6] Knowles, J.D., Watson, R.A., Corne, D.: Reducing local optima in single-objective problems by multi-objectivization. In: EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, pp. 269–283. Springer-Verlag, London, UK (2001)

[7] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer (8), 30–37 (2009)

[8] Krawiec, K.: Behavioral Program Synthesis with Genetic Programming, *Studies in Computational Intelligence*, vol. 618. Springer International Publishing (2015). DOI doi:10.1007/978-3-319-27565-9. URL http://www.cs.put.poznan.pl/kkrawiec/wiki/?n=Site.BPS

[9] Krawiec, K., Liskowski, P.: Automatic derivation of search objectives for test-based genetic programming. In: P. Machado, M.I. Heywood, J. McDermott, M. Castelli, P. Garcia-Sanchez, P. Burelli, S. Risi, K. Sim (eds.) 18th European Conference on Genetic Programming, *LNCS*, vol. 9025, pp. 53–65. Springer, Copenhagen (2015). DOI doi:10.1007/978-3-319-16501-1_5

[10] Krawiec, K., O'Reilly, U.M.: Behavioral programming: a broader and more detailed take on semantic GP. In: C. Igel, D.V. Arnold, C. Gagne, E. Popovici, A. Auger, J. Bacardit, D. Brockhoff, S. Cagnoni, K. Deb, B. Doerr, J. Foster, T. Glasmachers, E. Hart, M.I. Heywood, H. Iba, C. Jacob, T. Jansen, Y. Jin, M. Kessentini, J.D. Knowles, W.B. Langdon, P. Larranaga, S. Luke, G. Luque, J.A.W. McCall, M.A. Montes de Oca, A. Motsinger-Reif, Y.S. Ong, M. Palmer, K.E. Parsopoulos, G. Raidl, S. Risi, G. Ruhe, T. Schaul, T. Schmickl, B. Sendhoff, K.O. Stanley, T. Stuetzle, D. Thierens, J. Togelius, C. Witt, C. Zarges (eds.) GECCO '14: Proceedings of the 2014 conference on Genetic and evolutionary computation, pp. 935–942. ACM, Vancouver, BC, Canada (2014). DOI doi:10.1145/2576768.2598288. URL http://doi.acm.org/10.1145/2576768.2598288. Best paper

[11] Krawiec, K., Swan, J., O'Reilly, U.M.: Behavioral program synthesis: Insights and prospects. In: R. Riolo, W.P. Worzel, M. Kotanchek, A. Kordon (eds.) Genetic Programming Theory and Practice XIII, Genetic and Evolutionary Computation. Springer, Ann Arbor, USA (2015). URL http://www.cs.put.poznan.pl/kkrawiec/wiki/uploads/Research/2015GPTP.pdf

[12] Liskowski, P., Krawiec, K.: Discovery of implicit objectives by compression of interaction matrix in test-based problems. In: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (eds.) Parallel Problem Solving from Nature – PPSN XIII, *Lecture Notes in Computer Science*, vol. 8672, pp. 611–620. Springer (2014). DOI 10.1007/978-3-319-10762-2_60

[13] Liskowski, P., Krawiec, K.: Non-negative matrix factorization for unsupervised derivation of search objectives in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, pp. 749–756. ACM, New York, NY, USA (2016). DOI 10.1145/2908812.2908888. URL http://doi.acm.org/10.1145/2908812.2908888

[14] Liskowski, P., Krawiec, K.: Non-negative matrix factorization for unsupervised derivation of search objectives in genetic programming. In: T. Friedrich (ed.) GECCO '16: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, pp. 749–756. ACM, Denver, USA (2016). DOI doi:10.1145/2908812.2908888

[15] Liskowski, P., Krawiec, K.: Online discovery of search objectives for test-based problems. Evolutionary Computation pp. 1–32 (2016). DOI 10.1162/EVCO_a_00179. URL http://dx.doi.org/10.1162/EVCO_a_00179. PMID: 26953882

[16] Popovici, E., Bucci, A., Wiegand, R.P., de Jong, E.D.: Handbook of Natural Computing, chap. Coevolutionary Principles. Springer-Verlag (2011)

[17] Swan, J., Adriaensen, S., Bishr, M., Burke, E.K., Clark, J.A., Causmaecker, P.D., Durillo, J., Hammond, K., Hart, E., Johnson, C.G., Kocsis, Z.A., Kovitz, B., Krawiec, K., Martin, S., Merelo, J., Minku, L.L., Ozcan, E., Pappa, G.L., Pesch, E., Garcıa-Sánchez, P., Schaerf, A., Sim, K., Smith, J.E., Stützle, T., Voß, S., Wagner, S., Yao, X.: A research agenda for metaheuristic standardization. In: MIC 2015: The XI Metaheuristics International Conference (2015)