Automatic Derivation of Search Objectives for Test-Based Genetic Programming

Krzysztof Krawiec(✉) and Paweł Liskowski

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

Abstract. In genetic programming (GP), programs are usually evaluated by applying them to tests, and fitness function indicates only how many of them have been passed. We posit that scrutinizing the outcomes of programs' interactions with individual tests may help making program synthesis more effective. To this aim, we propose DOC, a method that autonomously derives new search objectives by clustering the outcomes of interactions between programs in the population and the tests. The derived objectives are subsequently used to drive the selection process in a single- or multiobjective fashion. An extensive experimental assessment on 15 discrete program synthesis tasks representing two domains shows that DOC significantly outperforms conventional GP and implicit fitness sharing.

Keywords: Genetic programming · Program synthesis · Test-based problems · Multiobjective evolutionary computation

1 Introduction

In genetic programming (GP), the quality of a candidate program is usually assessed by confronting it with a set of tests (fitness cases). The outcomes of program's interactions with individual tests are then aggregated by a fitness function. In discrete domains, this usually boils down to counting the number of passed tests.

Although employing a fitness function defined in this way may appear natural at first sight, there are several drawbacks of driving the search purely by the number of passed tests. Starting from not necessarily the most severe one, for n tests, fitness will take on $n + 1$ possible values, and once a search process identifies good and thus similarly fit solutions, ties become likely. Next, this quality measure is oblivious to the fact that some tests can be inherently more difficult to pass than others. But most importantly, aggregation of interaction outcomes into a single scalar implies compensation: two programs that perform very differently on particular tests may receive the same fitness and thus become indiscernible in a subsequent selection phase.

Furthermore, conventional fitness in GP is known to exhibit low fitness-distance correlation [22], i.e., it does not reflect well the number of search steps

P. Machado et al. (Eds.): EuroGP 2015, LNCS 9025, pp. 53–65, 2015.
DOI: 10.1007/978-3-319-16501-1_5

required to reach the optimal solution. As a result, guiding search by a fitness function defined in this way may be not particularly efficient. In other words, fitness function, despite embodying the *objective* quality of candidate solutions (considered as prospective outcomes of program synthesis process), is not necessarily the best *driver* to guide the search. Alternative *search drivers,* meant as substitutes for objective function, should be sought that correlate better with distance, possibly by reflecting other aspects of program behavior.

As we argued in [11], the habit of using scalar objective functions in domains like GP, where more detailed information on solutions' characteristic is easily available, seems particularly wasteful. The information on the outcomes of individual interactions can and should be exploited more efficiently wherever possible. In GP, search drivers could be evaluation measures that capture program's performance only on a *subset* of tests.

Various means, reviewed in Sect. 4 of this paper, have been proposed in the past to address the weaknesses of conventional fitness measure in GP. The method we propose here and describe in Sect. 3 is inspired by previous work in coevolutionary algorithms, and builds upon the approach we designed for test-based problems in [15]. In every generation, the algorithm identifies the groups of tests on which the programs in the current population behave *similarly.* Each such group gives rise to a separate *derived objective.* Typically, a few such objectives emerge from this process, and we employ them to perform selection on the current population. We propose two selection procedures that exploit the derived objectives, one of them involving the NSGA-II method [4]. In an experimental assessment reported in Sect. 5, the method performs significantly better than conventional GP and implicit fitness sharing.

2 Background

The task of automated program synthesis by means of genetic programming can be conveniently phrased as an optimization problem in which the search objective is to find a candidate solution $p^* = argmax_{p \in \mathcal{P}} f(p)$ that maximizes the objective function f, where $\mathcal{P}$ is the space of all candidate programs. In non-trivial problems, $\mathcal{P}$ is large or even infinite, and grows exponentially with the length of considered programs. Searching the entire space is therefore computationally infeasible, and one needs resort to a heuristic algorithm that is not guaranteed to find p^*. In GP, it is common to drive the search process using f as fitness function. As motivated earlier, this is not always the best approach.

A program to be evolved is typically specified by a set of tests (fitness cases). Each test is a pair $(x, y) \in T$, where x is the input fed into a program, and y is the desired outcome of applying it to x. From the machine learning perspective, T forms the training set. While in general the elements of $t \in T$ can be arbitrary objects, for the purpose of this study, we limit our interest to Boolean and integer-valued inputs and outputs.

In many problems, fitness cases do not enumerate all possible pairs of program inputs and outputs. Ideally, the synthesized program is expected to generalize beyond the training set which bears resemblance to *test-based problems*

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'

Fig. 1. Example of deriving search objectives from interaction matrix G (a) using clustering (b), resulting in the derived objectives shown in (c).

originating from the field of coevolutionary algorithms [1,3]. In test-based problems, candidate solutions interact with multiple environments – tests. Typically, the number of such environments is very large, making it infeasible to evaluate candidate solutions on all of them. Depending on problem domain, tests may take on the form of, e.g., opponent strategies (when evolving a game-playing strategy) or simulation environments (when evolving a robot controller).

In this light, it does not take long to notice that also the program synthesis task can be formulated as a test-based problem, in which passing a test requires a program to produce the desired output for a given input. In general, we will assume that an *interaction* between a program p and a test t produces a scalar outcome $g(p,t)$ that reflects the capability of the former to *pass* the latter. In this paper, we assume that interaction outcome is binary, i.e., $g : \mathcal{P} \times \mathcal{T} \to \{0,1\}$.

A GP algorithm solving a test-based problem (program synthesis task) maintains a population of programs $P \subset \mathcal{P}$. In every generation, each program $p \in P$ interacts with every test $(x,y) \in T$, in which p is applied to x and returns an output denoted as $p(x)$. If $p(x) = y$, p is said to *solve* the test and $g(p(x),y) = 1$. If, on the other hand, $p(x) \neq y$, we set $g(p(x),y) = 0$ and say that p *fails* (x,y).

As it will become clear in the following, it is convenient to gather the outcomes of these interactions in an *interaction matrix* G. For a population of m programs and $|T| = n$, G is an $m \times n$ matrix where g_{ij} is the outcome of interaction between the ith program and jth test.

Given this test-based framework, the conventional GP fitness that rewards a program for the number of passed tests can be written as

$$f(p) = |\{t \in T : \; g(p,t) = 1\}|. \tag{1}$$

3 The DOC Algorithm

The proposed method of discovery of search objectives by clustering (DOC) addresses the shortcomings of conventional evaluation (cf. Sect. 1) by clustering the interaction outcomes into several *derived objectives*. Each derived objective is intended to capture a subset of 'capabilities' exhibited by the programs in the context of other individuals in population. The derived objectives replace then the conventional fitness function (Eq. 1).

Technically, DOC replaces the conventional evaluation stage of GP algorithm (cf. Sect. 2) in favor of the following steps:

1. Calculate the $m \times n$ interaction matrix G between the programs from the current population P, $|P| = m$, and the tests from T, $|T| = n$.
2. Cluster the tests. We treat every column of G, i.e., the vector of interaction outcomes of all programs from P with a 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. The outcome of this step is a partition $\{T_1, \ldots, T_k\}$ of the original n tests in T into k subsets/clusters, where $1 \leq k \leq n$ and $T_j \neq \emptyset$.
3. Define the derived objectives. For each cluster T_j, we average row-wise the corresponding columns in G. This results in an $m \times k$ *derived interaction matrix* G', with the elements defined as follows:

$$g'_{i,j} = \frac{1}{|T_j|} \sum_{t \in T_j} g(s_i, t) \tag{2}$$

   where s_i is the program corresponding to the ith row of G, and $j = 1, \ldots, k$.

The columns of G' implicitly define the k *derived objectives* that characterize the programs in P.

The derived objectives form the basis for selecting the most promising programs from P, which subsequently give rise to the next generation of programs. The natural avenue here is to apply a multiobjective evolutionary algorithm. Following our previous work, we employ NSGA-II [4], one of the most popular method of that sort. This allows programs that feature different behaviors, reflected in the derived objectives, to coexist in population even if some of them are clearly better than others in terms of conventional fitness. However, we will show in the experimental section that such multiobjective selection may involve certain undesired side-effects, and that driving selection by certain scalar aggregate of the derived objectives can be also an interesting option.

Properties of DOC. An important property of DOC is its contextual character manifested by the fact that the outcome of evaluation of any program in P depends not only on the tests in T, but also on the other programs in P. This is the case because all programs in P together determine the result of clustering and therefore influence the derived objectives. This quite direct interaction between the programs is not a common feature of GP.

An implication of contextual evaluation is that derived objectives are *adaptive* and driven by the current state of evolving programs. The process of their discovery repeats in every generation so that they reflect the changes in behaviors of the programs in population. The derived objectives are thus *subjective* in this sense, which makes them analogue to search drivers used in two-population coevolution [15], even though the tests does not change with time here.

As clustering *partitions* the set of tests T (rather than, e.g., *selecting* some of them), none of the original tests is discarded in the transformation process. The more two tests are similar in terms of programs' performance on them, the more likely they will end up in the same cluster and contribute to the same derived objective. In the extreme case, tests that are mutually redundant (i.e., identical columns in G) are guaranteed to be included in the same derived objective.

For $k = 1$, DOC degenerates to a single-objective approach: all tests form one cluster, and G' has a single column that contains solutions' fitness as defined by Eq. 1 (albeit normalized). On the other hand, setting $k = n$ implies $G' = G$, and every derived objective being associated with a single test.

4 Related Work

There are two groups of past studies related to this work, those originating in GP and those originating in research on coevolutionary algorithms. We review these groups in the following.

In the group of **methods that originate in GP**, a prominent example of addressing the issues outlined in Sect. 1 is implicit fitness sharing (IFS) introduced by Smith *et al.* [20] and further explored for genetic programming by McKay [16,17]. IFS lets the evolution assess the difficulty of particular tests and *weighs* the rewards granted for solving them. Given a set of tests T, the IFS fitness of a program p in the context of a population P is defined as:

$$f_{IFS}(p) = \sum_{t \in T:\, g(p,t)=1} \frac{1}{|P(t)|} \tag{3}$$

where $P(t)$ is the subset of programs in P that solve test t, i.e., $P(t) = \{p \in P : g(p,t) = 1\}$. IFS treats tests as limited resources: programs *share* the rewards for solving particular tests, each of which can vary from $\frac{1}{|P|}$ to 1 inclusive. Higher rewards are provided for solving tests that are rarely solved by population members (small $P(t)$), while importance of tests that are easy (large $P(t)$) is diminished. The assessed difficulties of tests change as P evolves, which can help escaping local minima.

Other methods that reward solutions for having rare characteristics have been proposed as well. An example is co-solvability [10] that focuses on individual's ability to properly handle *pairs* of fitness cases, and as such can be considered a 'second-order' IFS. Such pairs are treated as elementary competences (skills) for which solutions can be awarded. Lasarczyk *et al.* [14] proposed a method for selection of fitness cases based on a concept similar to co-solvability. The method maintains a weighted graph that spans fitness cases, where the weight of an edge reflects the historical frequency of a pair of tests being solved simultaneously. Fitness cases are then selected based on a sophisticated analysis of that graph.

Last but not least, the relatively recent research on semantic GP [12] can be also seen as an attempt to provide search process with richer information of programs' behavioral characteristics. Similarly, pattern-guided GP and behavioral evaluation [13] clearly set similar goals.

In the group of studies that **originate in coevolutionary algorithms**, Pareto coevolution [6,18] was initially proposed to overcome the drawbacks of an aggregating fitness function. In Pareto coevolution, aggregation of interaction outcomes has been abandoned in favor of using each test as a separate objective. As a result, a test-based problem can be transformed into a multi-objective

optimization problem. This, in turn, allows adoption of dominance relation — a candidate solution s_1 dominates a candidate solution s_2 if and only if s_1 performs at least as good as s_2 on all tests. Nevertheless, the number of such elementary objectives is often prohibitively large due to a huge number of tests present in typical test-based problems.

It was later observed that certain test-based problems feature an *internal structure* comprising groups of tests that examine the same *skill* of solutions. Based on this observation, Bucci [1] and de Jong [2] introduced *coordinate systems* that *compress* the elementary objectives into a multidimensional structure, while preserving the dominance relation between candidate solutions. Because of the inherent redundancy of tests, the number of so-called underlying objectives (dimensions) in such a coordinate system is typically lower than the number of tests. However, even with a moderately large number of tests, it is unlikely for a candidate solution to dominate 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. The coordinate systems introduced in the cited work do not help in this respect, as they perfectly preserve the dominance relation, and if the dominance in the original space is sparse, they need to feature very high number of dimensions. Also, the problem of their derivation is NP-hard [8].

The derived objectives constructed by DOC bear certain similarity to the underlying objectives studied in the above works. However, as shown by the example in Fig. 1, the derived objectives are not guaranteed to preserve dominance: given a pair of candidate solutions (p_1, p_2) that do not dominate each other in the original space of interaction outcomes, one of them may turn out to dominate the other in the space of resulting derived objectives. For instance, given the interaction matrix as in Fig. 1a, program c does not dominate d, however it does so in the space of derived objectives (Fig. 1c). As a result of clustering, some information about the dominance structure has been lost. This inconsistency buys us however a critical advantage: the number of resulting derived objectives is low, so that together they are able to impose an effective search gradient on the evolving population.

5 Experimental Verification

We examine the capabilities of DOC within the domain of tree-based GP. The compared algorithms implement generational evolutionary algorithm and vary only in the selection procedure. Otherwise, they share the same parameter settings, with initial population filled with the ramped half-and-half operator, subtree-replacing mutation engaged with probability 0.1 and subtree-swapping crossover engaged with probability 0.9. We run two series of experiments: one with runs lasting up to 200 generations and population size $|P| = 500$, and with runs up to 100 generations and population size $|P| = 1000$. The search process stops when the assumed number of generation elapses or an ideal program is found; the latter case is considered a success.

Table 1. Success rate (percent of successful runs) of best-of-run individuals, averaged over 30 evolutionary runs. Bold marks the best result for each benchmark

	$\|P\| = 500$						$\|P\| = 1000$					
	GP	IFS	RAND	DOC	DOC-P	DOC-D	GP	IFS	RAND	DOC	DOC-P	DOC-D
Cmp6	20	**100**	50	21	83	78	26	**97**	48	22	64	77
Cmp8	0	**56**	0	0	21	29	0	**7**	0	0	4	5
Disc1	0	0	0	7	3	**13**	0	0	0	**10**	**10**	7
Disc2	0	4	0	10	14	**37**	0	0	0	0	21	**40**
Disc3	0	0	0	18	53	**62**	0	0	0	56	71	**77**
Disc4	0	0	0	0	0	**7**	0	0	0	**4**	0	0
Disc5	0	0	0	0	**7**	3	0	0	0	0	**4**	**4**
Maj6	22	**100**	60	40	83	90	52	**100**	71	81	96	89
Malcev1	0	18	24	18	70	**76**	14	27	33	25	69	**93**
Malcev2	3	3	0	7	27	**30**	0	0	11	17	**32**	27
Malcev3	0	7	8	23	**83**	**83**	0	3	8	43	**93**	75
Malcev4	0	0	4	7	**10**	7	0	0	0	**25**	20	10
Malcev5	17	30	25	54	47	**57**	17	23	44	**100**	68	60
Mux6	77	**100**	83	73	**100**	**100**	90	**100**	96	**100**	**100**	**100**
Par5	0	14	14	**18**	7	12	4	6	0	**18**	3	0

Compared algorithms. The particular implementation of DOC used in this work employs X-MEANS [19], an extension of the popular k-means algorithm that autonomously adjusts k. Given an admissible range of k, X-MEANS picks the k that leads to clustering that maximizes the Bayesian Information Criterion. In this experiment, we allow X-MEANS consider $k \in [1, 4]$ and employ the Euclidean metric to measure the distances between the observations (the columns of G).

We confront DOC with several control setups. The first baseline is the conventional Koza-style GP (**GP** in the following), which employs tournament of size 7 in the selection phase. The second control is implicit fitness sharing (**IFS** [17]) presented in Sect. 4, with fitness defined as in Formula 3 and also with tournament of size 7. The last control configuration, **RAND**, is a crippled variant of DOC. In that configuration, the tests, rather than being clustered based on interaction outcomes as described in Sect. 3, are partitioned into k subsets at random with k randomly drawn from the interval $[2, 4]$. RAND is intended to control for the effect of multiobjective selection performed by NSGA-II (which is known to behave very differently from the tournament selection).

Benchmark problems. In its current form presented in Sect. 3, DOC can handle only binary interaction outcomes, where a program either passes a test or not. Because of that, we compare the methods on problems with discrete interaction outcomes. The first group of them are **Boolean benchmarks**, which employ instruction set $\{and, nand, or, nor\}$ and are defined as follows. For an v-bit comparator $Cmp\,v$, a program is required to return *true* if the $\frac{v}{2}$ least significant input bits encode a number that is smaller than the number represented by the $\frac{v}{2}$ most significant bits. In case of the majority $Maj\,v$ problems, *true* should be returned if more that half of the input variables are *true*. For the multiplexer

Mul v, the state of the addressed input should be returned (6-bit multiplexer uses two inputs to address the remaining four inputs). In the parity *Par* v problems, *true* should be returned only for an odd number of *true* inputs.

The second group of benchmarks are the **algebra problems** from Spector *et al.*'s work on evolving algebraic terms [21]. These problems dwell in a ternary domain: the admissible values of program inputs and outputs are $\{0, 1, 2\}$. The peculiarity of these problems consists of using only one binary instruction in the programming language, which defines the underlying algebra. For instance, for the a_1 algebra, the semantics of that instruction is defined as in (a) below (see [21] for the definitions of the remaining four algebras). For each of the five algebras considered here, we consider two tasks (of four discussed in [21]). In the *discriminator term* tasks (*Disc* in the following), the goal is to synthesize an expression that accepts three inputs x, y, z and is semantically equivalent to the one shown in (b) below. There are thus $3^3 = 27$ fitness cases in these benchmarks. The second tasks (*Malcev*), consists in evolving a so-called *Mal'cev term*, i.e., a ternary term that satisfies the equation (c) below. This condition specifies the desired program output only for some combinations of inputs: the desired value for $m(x, y, z)$, where x, y, and z are all distinct, is not determined. As a result, there are only 15 fitness cases in our *Malcev* tasks, the lowest of all considered benchmarks.

a_1	0	1	2
0	2	1	2
1	1	0	0
2	0	0	1

a)

$$t^A(x, y, z) = \begin{cases} x & if\ x \neq y \\ z & if\ x = y \end{cases}$$

b)

$$m(x, x, y) = m(y, x, x) = y$$

c)

Performance. Table 1 reports the success rates of particular algorithms, resulting from 30 runs of each configuration on every benchmark. The methods clearly fair differently on particular benchmarks. To provide an aggregated perspective on performance, we employ the Friedman's test for multiple achievements of multiple subjects [9]. Compared to ANOVA, it does not require the distributions of variables in question to be normal.

Friedman's test operates on average ranks, which for the considered methods are as follows, for $|P| = 500$ (left) and $|P| = 1000$ (right):

DOC	IFS	RAND	GP	DOC	IFS	RAND	GP
1.93	2.20	2.50	**3.36**	1.76	2.33	2.60	**3.30**

The p-value for Friedman test is $\ll 0.001$, which strongly indicates that at least one method performs significantly different from the remaining ones. We conducted post-hoc analysis using symmetry test [7]: bold font marks the methods that are outranked at 0.05 significance level by the *first* method in the ranking.

Analysis. Although DOC ranks first for both population sizes, it does not seem to be much better than IFS, a substantially simpler method. We hypothesize that this may be an effect of *overspecialization*, which may be likened to *focusing*, one of so-called coevolutionary pathologies [5,23]. Even though evolving a program

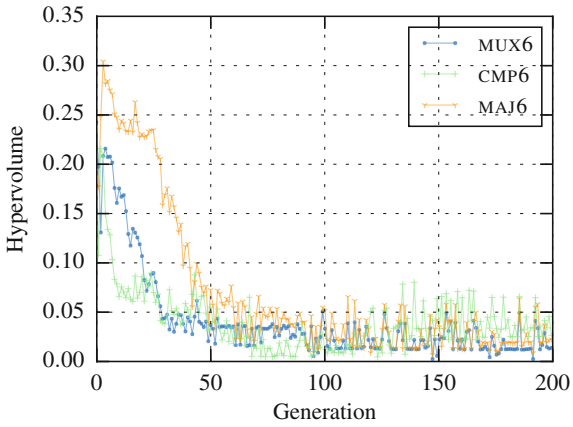

Fig. 2. Average hypervolume of programs in population across generations for the Mux6, Cmp6 and Maj6 benchmarks.

that passes all tests is hard, it may be relatively easy to find programs that perform well on a certain *subset* of tests while failing the other tests. For instance, in the Boolean benchmark *Cmp6*, the task is to determine whether the number encoded by the three least significant input bits b_0, b_1, b_2 is smaller than the number encoded by the three most significant bits b_3, b_4, b_5. A program that checks if b_2 is off and simultaneously b_5 is on solves the quarter of $2^6 = 64$ tests in this task. This can be expressed with a mere few instructions from the assumed instruction set, e.g., as $(b_2\ nor\ b_2)\ and\ b_5$. It is possible that evolution exploits this opportunity by synthesizing programs that focus on such easy subproblems.

To verify this hypothesis, we define the *hypervolume* of program's performance as characterized by the k derived objectives $o_1, \ldots, o_k$, i.e.,

$$h(p) = \prod_{i=1}^{k} o_i(p). \tag{4}$$

The key property of hypervolume is that it increases as the scores on o_is become more balanced. Consider two programs p_1, p_2 with the same overall fitness, i.e., $\sum_i o_i(p_1) = \sum_i o_i(p_2)$. Assume the scores of p_1 on o_is vary, while those of p_2 are all the same, i.e., $o_i(p_2) = \sum_i o_i(p_1)/k$. In such a case, $h(p_2) > h(p_1)$. $h(p_2)$ is the maximum hypervolume for all possible distributions of the same scalar fitness across the derived objectives.

Figure 2 plots the hypervolume of programs in population across generations for the *Mux*6, *Cmp*6 and *Maj*6 benchmarks, averaged over population and over 90 evolutionary runs. We observe dramatic decline of this measure with evolution time. With the other benchmarks exhibiting similar characteristics, we can conclude that indeed the programs evolved by DOC tend to overspecialize.

Promoting uniform progress. The NSGA-II selection procedure operates on Pareto ranks and as such is agnostic to a more detailed location of a given point in the multiobjective space that spans o_is. As long as two programs have the same Pareto rank, they will be equally valuable (unless differentiated by sparsity). This holds even if one of them is on the very extreme of Pareto front, i.e., attains zero value of one or more objectives. In other words, NSGA-II lacks mechanisms that would promote achieving balanced performance on *all* derived objectives simultaneously.

This observation, combined with the above demonstration of overspecialization, immediately points to a remedy. If hypervolume is a natural measure of balanced performance on all objectives, why not use it as a search driver? To verify this idea, we come up with a straightforward variant of DOC, called **DOC-P** in the following. DOC-P aggregates the scores on derived objectives using Formula 4, and uses the resulting hypervolume as fitness in combination with tournament selection of size 7, as in the other control configurations.

We also propose a second variant of this idea, **DOC-P**, which additionally *weights* the objectives by the number of tests (columns in G) included in each objective, i.e.,

$$h_D(p) = \prod_{i=1}^{k} |T_i| o_i(p). \tag{5}$$

In effect, $h_D(p)$ is based on the *number* of tests passed by p on each derived objectives, while h relied on the raw values of o_j, i.e., *mean* test outcomes in clusters.

The columns in Table 1 marked DOC-P and DOC-D report the results of these methods. Below, we present the average ranks of all methods, including these extensions:

DOC-D	DOC-P	IFS	DOC	RAND	GP
1.70	2.43	3.56	**3.63**	**4.33**	**5.33**

DOC-P	DOC-D	DOC	IFS	RAND	GP
2.20	2.43	3.10	3.66	**4.50**	**5.10**

We observe both setups dramatically improving the performance compared to the original DOC. For $|P| = 500$ (left), the DOC-D ranks the best, outperforming GP, RAND and the multiobjective variant of DOC in a statistically significant way. The difference is statistically insignificant for IFS, but both DOC-D and DOC-P score higher success rates more often and manage to solve two problems that remained unsolved by other algorithms, i.e., *Disc4* and *Disc5*.

The result are quite similar when $|P| = 1000$ (right), however this time DOC-P stands out as the best, albeit its rank is only slightly higher than that of DOC-D. Larger population is also beneficial for multiobjective DOC allowing it to achieve lower rank than IFS and beat GP in a statistically significant way. We speculate that this effect is directly related to the Pareto-fronts becoming densely populated, and thus decreasing the risk of over-specialization.

The experimental results clearly indicate that both DOC-P and DOC-D are more likely to find an ideal solution than the traditional GP and prove capable of solving problems that GP struggles with. If a larger population size is

admissible, multiobjective DOC also emerges as a viable alternative to IFS and conventional GP.

6 Conclusions

In this paper we proposed a method that heuristically derives new search objectives by clustering the outcomes of interactions between the programs in population and the tests. The derived search objectives, either combined with the NSGA-II or combined into a hypervolume of program's performance, effectively enhance conventional GP. DOC manages to produce a low number of objectives that approximately capture the capabilities of evolving programs. Once identified, DOC maintains the presence of such skills in the population, even if the programs featuring them are inferior according to the conventional fitness. In this study, the capabilities in question concerned program output; in general, they may correspond to program *behaviors* in a broader sense, or reflect whether they satisfy certain *conditions*. Such generalizations deserve investigation in the future work.

When seen from the perspective of the overall evolutionary workflow, DOC broadens the 'bottleneck of evaluation' described in Introduction in characterizing the candidate solutions with multiple objectives rather than with a single one. Objectives derived by DOC constitute alternative search drivers that replace the conventional fitness function and guide the search in a single- or multiobjective fashion. Ultimately, capabilities elaborated by particular individuals have the chance of being fused in their offspring and so ease reaching the search goal. In this context, there is an interesting relationship between the derived objectives and the *intermediate* results produced by programs studied in behavioral evaluation [12] and pattern-guided genetic programming [13].

Acknowledgments. P. Liskowski acknowledges support from grant no. 09/91/DSPB/0572.

References

1. Bucci, A., Pollack, J.B., de Jong, E.: Automated extraction of problem structure. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 501–512. Springer, Heidelberg (2004)
2. de Jong, E.D., Bucci, A.: DECA: dimension extracting coevolutionary algorithm. In: Cattolico, M., et al., (eds.) GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 313–320. ACM Press, Seattle, Washington, USA (2006)
3. de Jong, E.D., Pollack, J.B.: Ideal evaluation from coevolution. Evol. Comput. **12**(2), 159–192 (2004)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

5. Ficici, S.G., Pollack, J.B.: Challenges in coevolutionary learning: arms-race dynamics, open-endedness, and mediocre stable states. In: Proceedings of the Sixth International Conference on Artificial Life, pp. 238–247. MIT Press (1998)
6. Ficici, S.G., Pollack, J.B.: Pareto optimality in coevolutionary learning. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, p. 316. Springer, Heidelberg (2001)
7. Hollander, M., Wolfe, D.A., Chicken, E.: Nonparametric Statistical Methods, vol. 751. John Wiley & Sons, Weinheim (2013)
8. Jaśkowski, W., Krawiec, K.: Formal analysis, hardness and algorithms for extracting internal structure of test-based problems. Evol. Comput. **19**(4), 639–671 (2011)
9. Kanji, G.K.: 100 Statistical Tests. Sage, London (2006)
10. Krawiec, K., Lichocki, P.: Using co-solvability to model and exploit synergetic effects in evolution. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 492–501. Springer, Heidelberg (2010)
11. Krawiec, K., O'Reilly, U.M.: Behavioral programming: a broader and more detailed take on semantic GP. In: Igel, C. (ed.) GECCO 2014: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, pp. 935–942. ACM, Vancouver, BC, Canada, 12–16 July 2014
12. Krawiec, K., O'Reilly, U.-M.: Behavioral search drivers for genetic programing. In: Nicolau, M., Krawiec, K., Heywood, M.I., Castelli, M., García-Sánchez, P., Merelo, J.J., Rivas Santos, V.M., Sim, K. (eds.) EuroGP 2014. LNCS, vol. 8599, pp. 210–221. Springer, Heidelberg (2014)
13. Krawiec, K., Swan, J.: Pattern-guided genetic programming. In: Blum, C. (ed.) GECCO 2013: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, pp. 949–956. ACM, Amsterdam, The Netherlands, 6–10 July 2013
14. Lasarczyk, C.W.G., Dittrich, P., Banzhaf, W.: Dynamic subset selection based on a fitness case topology. Evol. Comput. **12**(2), 223–242 (2004)
15. Liskowski, P., Krawiec, K.: Discovery of implicit objectives by compression of interaction matrix in test-based problems. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 611–620. Springer, Heidelberg (2014)
16. McKay, R.I.B.: Committee learning of partial functions in fitness-shared genetic programming. In: Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE Third Asia-Pacific Conference on Simulated Evolution and Learning 2000. vol. 4, pp. 2861–2866. IEEE Press, Nagoya, Japan, 22–28 October 2000
17. McKay, R.I.B.: Fitness sharing in genetic programming. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), pp. 435–442. Morgan Kaufmann, Las Vegas, Nevada, USA, 10–12 July 2000
18. Noble, J., Watson, R.A.: Pareto coevolution: using performance against coevolved opponents in a game as dimensions for pareto selection. In: Spector, L., et al., (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 493–500. Morgan Kaufmann, San Francisco, California, USA, 7–11 July 2001
19. Pelleg, D., Moore, A.W., et al.: X-means: extending k-means with efficient estimation of the number of clusters. In: ICML, pp. 727–734 (2000)
20. Smith, R.E., Forrest, S., Perelson, A.S.: Searching for diverse, cooperative populations with genetic algorithms. Evol. Comput. **1**(2), 127–149 (1993)

21. Spector, L., Clark, D.M., Lindsay, I., Barr, B., Klein, J.: Genetic programming for finite algebras. In: Keijzer, M. (ed.) GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 1291–1298. ACM, Atlanta, GA, USA, 12–16 July 2008
22. Tomassini, M., Vanneschi, L., Collard, P., Clergue, M.: A study of fitness distance correlation as a difficulty measure in genetic programming. Evol. Comput. **13**(2), 213–239 (2005)
23. Watson, R.A., Pollack, J.B.: Coevolutionary dynamics in a minimal substrate. In: Spector, L., et al., (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 702–709. Morgan Kaufmann, San Francisco, California, USA, 7–11 July 2001