Discovery of Search Objectives in Continuous Domains

Paweł Liskowski
Poznan University of Technology
Poznan, Poland
pliskowski@cs.put.poznan.pl

Krzysztof Krawiec
Poznan University of Technology
Poznan, Poland
krawiec@cs.put.poznan.pl

ABSTRACT

In genetic programming (GP), the outcomes of the evaluation phase can be represented as an interaction matrix, with rows corresponding to programs in a population and columns corresponding to tests that define a program synthesis task. Recent contributions on Discovery of Objectives via Clustering (DOC) and Discovery of Objectives by Factorization of interaction matrix (DOF) show that informative characterizations of programs can be automatically derived from interaction matrices in discrete domains and used as search objectives in multidimensional setting. In this paper, we propose analogous methods for continuous domains and compare them with conventional GP that uses tournament selection, Age-Fitness Pareto Optimization, and GP with epsilon-lexicase selection. Experiments show that the proposed methods are effective for symbolic regression, systematically producing better-fitting models than the two former baselines, and surpassing epsilon-lexicase selection on some problems. We also investigate the hybrids of the proposed approach with the baselines, concluding that hybridization of DOC with epsilon-lexicase leads to the best overall results.

CCS CONCEPTS

•Software and its engineering → Genetic programming; •Theory of computation → *Evolutionary algorithms;*

KEYWORDS

Genetic Programming; Machine Learning; Nonnegative Matrix Factorization; Multiobjective optimization

ACM Reference format:
Paweł Liskowski and Krzysztof Krawiec. 2017. Discovery of Search Objectives in Continuous Domains. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071344

1 INTRODUCTION

Fitness function in GP is intended to reflect candidate program's conformance with the desired behavior, typically given as a set of training examples (fitness cases, tests). It is arguably convenient as a succinct yardstick of program's quality. On the other hand, it is also extremely crude in its aggregate characterization of the outcomes resulting from applying a program to particular tests. Even though some programs in a population may fare better than others on some tests but not on others (and vice versa), these individual differences are often not reflected in the overall fitness, which informs on the average performance only. Arguably, a richer characterization of program performance might help making search more effective, not least by providing natural means for differentiation in the population.

In this paper, we follow the works that aim at enriching program characteristics by describing them with multiple performance characteristics rather than one. Specifically, by this we do not mean conventional 'helper objectives' occasionally used in EC and GP and typically designed and/or engaged manually (like, e.g., program size in GP). It has been posited in recent works [10, 14, 16] that useful multi-objective characterization of programs can be achieved automatically, in a largely data-driven manner. The key observation behind those works is that m programs in the population confronted with n tests result in an $m \times n$ interaction matrix, which can be automatically 'datamined' for *derived objectives*. Such objectives may, but do not have to, correspond to *underlying objectives* of a problem, a concept devised in the area of coevolutionary algorithms [1, 3, 7]. Either way, they offer a richer characterization of program behavior than scalar fitness.

Empirical evidence brought in previous works suggests significant practical utility of methods of this kind, which not only prove better than conventional GP that navigates the search space using scalar fitness function, but can also successfully compete with efficient alternative techniques like Lexicase selection [5, 12]. The methods proposed to date are however limited in being applicable only to problems with binary interaction outcomes, i.e. domains where programs either pass a test or not. Given that a significant fraction of GP practice revolves around continuous domains (viz. symbolic regression), it becomes natural to seek for means of extending such approaches in that direction.

The novel contribution of this paper is thus a (largely universal) approach for transforming interaction matrices generated in continuous domains to a form that is appropriate for the existing methods that construct derived objectives from interaction matrices. Apart from presenting and motivating this particular method, we thoroughly assess its performance on a range of uni- and multivariate symbolic regression benchmarks. Additionally, we hybridize our approach with Lexicase selection and compare it against it. Empirical evidence indicates high effectiveness of the proposed approach and points to possibilities of interesting extensions.

2 BACKGROUND

For conformance with its 'mother field' of evolutionary computation, it is rather common in GP to characterize programs with scalar fitness. As argued in Introduction, there is nothing wrong with this perspective, as long as it does not stop one from considering

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071344

the alternatives. In contrast to, e.g. black-box optimization, where nothing or little more than candidate solution's fitness is available, fitness in GP stems from program's interaction with multiple tests. Crucially, the outcomes of those interactions are easily available, at no extra cost, to a search algorithm.

Formally, the fitness $f(p)$ of a program p assessed on a set of n tests T of the form (in, out) is for discrete domains usually defined as the number of passed tests:

$$f(p) = \sum_{(in,out)\in T} [p(in) = out], \tag{1}$$

where $p(in)$ is the output produce by p for in, and $[\cdot]$ is the Iverson bracket. For continuous domains, f will usually compute some form of error, e.g. the mean absolute deviation:

$$f(p) = \frac{1}{m} \sum_{(in,out)\in T} |p(in) - out|. \tag{2}$$

In these and most other cases, individual interactions of p with tests matter only *en masse*. In the context of the current population P, they can be conveniently gathered in an interaction matrix $G = [g_{ij}]$, where g_{ij} is the outcome of interaction of ith program in P with jth test in T. In the discrete case, $g_{ij} = [p_i(in_j) = out_j]$ (cf. (1)). In the continuous case, $g_{ij} = |p_i(in_j) - out_j|$ (cf. (2)).

Even for relatively small population size $m = |P|$ and a moderate number of tests n, the $m \times n$ interaction matrix G may become quite large. However, the elements of G are routinely computed also in conventional GP, as evidenced by (1) and (2) – it is just they are usually not memorized, as in conventional GP there is no need for that. Thus, other than significant memory footprint, obtaining interaction matrices do not bring any computational overheads. On the positive side, they offer rich information on the behavior of individual programs on individual tests. On one side, an interaction matrix may convey useful information on the structure of the current population – for instance the most straightforward distribution of fitness. On the other hand, it reveals a great deal about the characteristics of tests, not least about their difficulty – exploited in past approaches like implicit fitness sharing [19]. These 'marginal' perspectives on G are however still rather limited, compared to the more recent approaches that focus on individual interactions, which we cover in the next section.

3 METHODS

3.1 Discovery of Objectives by Clustering

Discovery of Search Objectives by Clustering (DOC) [10] is a method that autonomously derives new search objectives by clustering the outcomes of interactions between programs in the population and the tests. These objectives characterize the programs in P and form the basis for selecting the most promising programs for the next generation. Technically, DOC applies clustering to n columns of G that are treated as points in m-dimensional space. For each resulting cluster T_j, the corresponding columns in G are averaged row-wise, giving rise to $m \times k$ *derived interaction matrix* G' with the elements defined as follows:

$$g'_{ij} = \frac{1}{|T_j|} \sum_{t\in T_j} g(p_i, t), \tag{3}$$

where p_i is the program corresponding to the ith row of G, and $j = 1, \ldots, k$. Each derived objective is intended to capture a subset of 'capabilities' exhibited by the programs in the context of other individuals in a population. The k derived objectives replace the conventional fitness function (Eq. 1) and are subsequently used to drive the selection process in a multi-objective fashion. DOC builds upon the approach designed for coevolutionary algorithms in [13, 15].

DOC is designed to be sensitive to inherent difficulty of tests by avoiding the problem of compensation deeply rooted in aggregating fitness functions (cf. Section 1). For this purpose, DOC transforms a single objective-problem given by the original objective function into a multi-objective one to facilitate the use of *dominance relation*. The dominance relation in the original space of interaction outcomes (where each test is treated as a separate objective) tends to be sparse. DOC compresses (in a lossy way) the interaction outcomes into a low number of derived objectives, so that the dominance relation induced in their space is more likely to be dense and lend itself to elicitation of useful search gradient.

3.2 Discovery of Objectives by Factorization

Discovery of Objectives via Factorization (DOF) proposed in [14] is another method that aims at scrutinizing the individual outcomes of programs' interactions and leveraging them for better performance. DOF employs a non-negative matrix factorization (NMF) to heuristically derive search objectives from an interaction matrix G, and similarly to DOC, uses these objectives to drive search using a multi-objective selection method. DOF searches for matrices W and H that together form a lower rank approximation of G, i.e.:

$$G \approx WH \;\; s.t. \;\; W, H \geq 0, \tag{4}$$

where $W \in \mathbb{R}^{m\times r}$ is called *weights matrix*, $H \in \mathbb{R}^{r\times n}$ is a *feature matrix*, and r is a desired factorization rank (typically $r \ll min(m, n)$). Each program $p \in P$ is associated with a row in W (a vector $w_p \in \mathbb{R}^r$), and each test $t \in T$ corresponds to a column in H (a vector $h_t \in \mathbb{R}^r$). To perform factorization, DOF solves the following optimization problem:

$$\min_{W,H} f(W, H) \equiv \frac{1}{2}||G - WH||_F^2 \;\; s.t. \;\; W, H \geq 0, \tag{5}$$

where $||\cdot||_F$ is the Frobenius norm. Once W and H are known, an estimate of an interaction outcome of a program p with a test t is given by the dot product of two vectors corresponding to p and t:

$$\hat{g}_{pt} = w_p^T h_t = \sum_{j=1}^{r} w_{pj} h_{jt}. \tag{6}$$

The central observation that motivates DOF is that NMF can be used to explain the interaction outcomes in G by characterizing both programs and tests in terms of factors inferred from the patterns observed in their interactions. These factors are used to recast the GP problem as multi- rather than single-optimization problem. In every generation, DOF feeds the factors from W directly into NSGA-II [4] selection procedure in order to select the parent programs and generate candidate solutions for the next generation.

The main difference between DOF and DOC is that the objectives in the former *cover* the interaction matrix, while they *partition* it in the latter. In DOC, a test contributes to exactly one objective, while

in DOF each derived objective (a column of W) depends in general on all program-test interactions. The space of derived objectives that can be designed by DOF is thus larger than for DOC. Moreover, DOC needs all interaction outcomes to be available, while DOF can work with a partial interaction matrix. Performing only a fraction of interactions and employing NMF to estimate the outcomes of remaining ones allows for substantial reduction of computational effort [16], or investing these savings in increased population [14].

4 DISCOVERY OF OBJECTIVES IN CONTINUOUS DOMAINS

DOC and DOF have been originally applied to discrete domains and can handle only binary interaction outcomes, where a program either passes a test or not. In order to apply them to domains with continuous interaction outcomes, we introduce a preprocessing step that maps the – in general arbitrary large – continuous error to interaction outcomes.

The method proceeds as follows:

(1) Calculate the interaction matrix G between the programs from the current population P and the tests from T. We assume G initially contains *errors*, i.e. $g_{ij} \in \mathbb{R}_{\geq 0} \cup \{\text{NaN}\}$, where NaNs signal execution errors (like division by zero etc.). For regression tasks considered in this paper, g_{ij} is the absolute deviation of program output w.r.t. desired output, i.e. $g_{ij} = |p_i(t_j) - y_j|$.
(2) Replace in G NaNs and values that are greater than a threshold φ with the positive infinity $+\infty$.
(3) Standardize the columns of G, omitting any infinities.
(4) Apply sigmoid 'squeezing' non-linearity $1/(1+e^{-x})$ to the elements of G.

As a result of applying the above preprocessing steps, the values in G are now guaranteed to be in the range $[0, 1]$. Crucially, the smaller φ, the more sensitive the method becomes to small errors. For example, assuming $\varphi = 200$ and the following errors made by a single individual on six test cases in the initial G:

$$0.3 \quad 1.6 \quad 3.3 \quad 150.8 \quad 51.3 \quad 160.3$$

none of them would be omitted during standardization. Moreover, relatively small interaction outcomes for the second and third elements, become indiscernible after standardization and squeezing:

$$0.29 \quad 0.3 \quad 0.3 \quad 0.79 \quad 0.46 \quad 0.81$$

which may have consequences for further processing, in particular for selection. If, on the other hand, we set $\varphi = 50$, only the first three errors would be considered for computing mean and standard deviation necessary for standardization. As a result, the differences between the smaller errors are amplified:

$$0.24 \quad 0.47 \quad 0.78 \quad 1.0 \quad 1.0 \quad 1.0$$

The motivation for parameterizing this process with φ is that maintaining the differences between smaller errors may be more important than doing so for the bigger errors, as for the latter it may be justified to consider them 'equally bad'. In the experimental section, we propose to automate the choice of φ by setting it to the 95th percentile of the errors in an interaction matrix.

Other design choices such as standardization and sigmoid non-linearity are motivated by the importance of providing for the same magnitude of outcomes on tests while maintaining their individual capability of discrimination between programs, and the desire to map the outcomes in entire matrix to the same interval so that they are globally comparable. The former is particularly important for DOC, as it employs clustering based on Euclidean distances that is isotropic in all directions of space and therefore sensitive to unequal variances in different dimensions.

5 RELATED WORK

The concept of derived search objectives has a counterpart in co-evolutionary algorithms. De Jong [3] and Bucci [1] independently proposed to map interaction matrices to *coordinate systems*, with axes representing different *skills* exhibited by candidate solutions, and in this sense embodying *underlying* objectives. Such coordinate systems are guaranteed preserve dominance in the original interaction matrix, and their dimensionality reflects problem difficulty. However, because the problem of finding the coordinate system of minimal dimensionality is NP-hard [7], and system's dimensionality for even the simplest test-based problems turns out to be very high, using these formalisms to actually drive search is rare and usually does not lead to spectacular improvements.

Methods that that attempt to derive alternative search objectives are also related to semantic GP [20] and behavioral [9, 11] GP methods that define program semantics as the vector of outputs produced by a program for particular tests. A single row in an interaction matrix might be viewed as the outcome of confronting program's semantics with the vector of desired outputs. With the increasing popularity of semantic methods in GP, recent years have seen a large number of contributions that employ this characterization of program behavior to design new initialization, search, and selection operators. However, those methods are in general not designed to redefine search objectives, which is the primary goal of the methods that derive new search objectives.

Regarding other, less related approaches, the methods studied here can be likened to methods that redefine fitness function. The arguably best known approach of this type is implicit fitness sharing (IFS) introduced by Smith *et al.* [24] and further explored for genetic programming by McKay [19]. IFS estimates the difficulty of particular tests from an interaction matrix and weighs accordingly the rewards granted to programs. Higher rewards are provided for solving tests that are rarely solved by population members. The difficulties of tests change with evolution, which can help escaping local minima and diversifies population.

6 EXPERIMENTS

We conduct an extended experimental assessment of DOC and DOF by confronting them with reference approaches on a suite of 18 symbolic regression benchmarks. The objective of the experiment is to gauge the performance of the these methods in terms of typical metrics such as training set error, test set error and program size in the domain of tree-based GP.

As already signaled in Section 4, we set φ to 95th percentile of the errors in an interaction matrix. This allows us to ignore the top 5% of the biggest errors that would otherwise distort the standardization of G. In the preliminary experiments, we also considered median which turned out to perform slightly worse.

Table 1: Parameters of evolution.

Parameter	Value
population size	1000
termination condition	200 generation or fitness =0
initialization	ramped half-and-half
instruction set	{+,-,×,/,exp,log,sin,cos}
tournament size	7
crossover probability	0.9
mutation probability	0.1
number of runs	50

All compared methods implement generational evolutionary algorithm and share the same parameter settings, with initial population of size $|P| = 1000$ 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. Search lasts up to 200 generations and terminates when the assumed number of generation elapses or an ideal program is found. Details concerning these and other GP settings are summarized in Table 1.

6.1 Compared algorithms

DOC employs x-means [22] clustering algorithm that extends k-means by autonomously adjusting 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 $k \in [1, 5]$ and employ the Euclidean metric to measure the distances between the observations (columns of G).

DOF employs NMF that is realized by the stochastic gradient descent algorithm. The factorization rank r is set to 2 since NSGA-II tends to work best with few objectives. Following [14], we set the fraction of calculated interactions $\alpha = 0.4$ and increase the population size by the factor $(1 - \alpha)$ to align the computational effort. The remaining parameters are set according to [14].

Both DOC and DOF rely on NSGA-II [4] selection procedure in order to select the parent programs and generate candidate solutions for the next generation. NSGA-II uses the default value of tournament size, i.e., 2.

We confront the above methods 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 other control methods are shortly introduced in the following subsections.

6.1.1 Lexicase selection. ϵ-lexicase selection (LEX) has been recently proposed for symbolic regression problems [12], and builds upon lexicase selection that has been originally designed for 'uncompromising' problems [5]. In each parent selection event, the method starts by shuffling the test cases, and proceeds by removing the individuals in the selection pool that do not satisfy a pass condition c_t on the first test. In its original definition, c_t filters all individuals with a fitness worse than the best fitness on the current test. If more than one individuals remains in the pool, the first case is removed and the procedure is repeated for the next case until only one individuals remains and becomes a parent, or all fitness cases are used. In such a case, a parent is chosen randomly from the remaining individuals. ϵ-lexicase selection addresses poor performance of lexicase selection on continuous errors by modulating the pass condition c_t on test cases via ϵ, so that only individuals outside ϵ are filtered during selection. In the following, we use the variant of ϵ-lexicase selection that proved to be the most effective in [12] and uses pass condition:

$$e_t(p) < e_t^* + \lambda(\mathbf{e}_t), \tag{7}$$

where $e_t(p)$ is the error of program p on test t, e_t^* is the best error on t in P, and $\lambda(\mathbf{e}_t)$ is the median absolute deviation of the errors on test t across the population.

Similarly to DOC, in addition to rewarding the programs for solving test cases, ϵ-lexicase selection promotes diversified programs that pass randomly selected subset of tests. In this way, different tests are emphasized in each selection event. An individual that passes test(s) that are rarely passed by its competitors has substantial chance to propagate to the next generation even if it performs poorly on many other test. Note that in contrast to DOC and DOF, LEX does not explicitly define any objectives or alternative fitness functions. In this sense, it is 'natively' a selection method.

6.1.2 Age-Fitness Pareto Optimization. Age-Fitness Pareto Optimization (AFPO) [23] is a multi-objective method that assigns each individual an age equal to the number of generations since inception of its oldest ancestor. Each generation, AFPO selects random parents from the population and applies crossover and mutation operators to produce $|P| - 1$ offspring. The offspring and a single randomly initialized individual are then added to the population doubling its size. Next, Pareto tournament selection is iteratively applied by randomly selecting a subset of individuals and removing the dominated ones until the size of the population is reduced back to $|P|$. To determine which individuals are dominated, the algorithm identifies the Pareto front using two objectives: age and fitness.

6.1.3 Hybrid Approaches. In their original form, derived objectives identified by DOC and DOF drive the selection process in a multi-objective fashion to avoid aggregation of interaction outcomes with all tests into a single scalar value, which is characteristic for the traditional objective function. One of the main motivations for LEX is also to avoid such aggregation, and the decisions made by the algorithm regarding which programs to select are based on distinct tests. These observations encourage us to combine these methods into a hybrid approaches in which lexicase selection is performed on the derived objectives.

In hybrid approaches, DOCLEX and DOFLEX, we first derive new search objectives and subsequently, we apply ϵ-lexicase selection, using the particular derived objectives d_i as if they were test cases in ϵ-lexicase selection. In each iteration, a derived objective is drawn at random. Then, individuals in the population that do not satisfy the pass condition c_t on that objective are filtered. If more than one individual remains, the process repeats, filtering any remaining individuals that do not satisfy c_t on the next derived objective drawn at random. This process continues until only one individual remains and is selected, or until all derived objectives have been processed, in which case a random program is selected from the remaining programs. In DOCLEX, we set $k = [10, 100]$. Similarly, in DOFLEX, we set $r = [10, 100]$ and use all factors from W.

6.2 Benchmark problems

We compare the methods on 18 uni- and bi-variate symbolic regression problems that come from [18, 21]. In univariate problems, 20 Chebyshev nodes [2] are used from training, and 20 uniformly sampled points for testing. For bivariate problems, 10 values are picked in analogous way for each input variable. Their Cartesian product constitutes a data set. By varying in the number of variables, the required functions, and the characteristic of desired output, this selection of benchmarks forms good representation of problems considered in research on GP and its practical applications.

6.3 Results

Figure 1 shows the average best-of-generation fitness achieved by particular methods on different benchmark problems, with 95% confidence intervals marked as semi-transparent bands. Clearly, each of the considered methods that drive the search using derived objectives significantly improves the performance of the standard GP algorithm. The best performance is achieved either by DOC or DOCLEX, depending on the problem. LEX and DOCLEX tend to maintain the lowest training set error during evolution, with DOC eventually catching up. DOF performs slightly worse than the already mentioned methods, but still better than two other control methods AFPO and GP. DOFLEX, on the other hand, seems to be the weakest algorithm of the analyzed. It has the highest variance, and often achieves inferior results when compared to other methods. These observations are confirmed by Table 2 that demonstrates average and 95% confidence intervals of the best-of-run fitness.

To provide an aggregated perspective on performance, we employ the Friedman's test for multiple achievements of multiple subjects [8] on the best-of-run fitness. The p-value for Friedman test is $3.46{\times}10^{-12}$, which strongly indicates that at least one method performs significantly different from the remaining ones. To determine the significantly different pairs we conduct post-hoc analysis using symmetry test [6]. Table 5 presents the p-values for the hypothesis that a setup in a row is better than a setup in a column. The significant p-values are marked in bold. This comparison indicates that the performance improvement of DOC and DOCLEX relative to control methods GP and AFPO is significant. For a more detailed insight, we also rank all configurations in the bottom row of Table 2. The best overall average rank of 1.39 is achieved by DOCLEX which outperforms the other methods on 13/18 benchmarks. The second is DOC with the average rank of 2.22 and the lowest error on 3/18 benchmarks. Third place is taken by LEX with the average rank of 2.89 and the lowest error on 1/18 benchmarks.

Table 3 presents the median and 95% confidence interval of test set fitness of the best-of-run program. The results are mostly consistent with the results on the training set, and again we observe the positive effects of driving search using derived objectives. Across all problems, the median best fitness on the test sets is obtained by DOC which achieves the best overall average rank of 1.72. DOCLEX is second with the rank of 2.67, while LEX and DOF both rank third with the same average result of 3.33. The methods do not exhibit heavy overfitting to training data except for Kj8, Kj15 and Pg1, where higher test errors are apparent. Interestingly, despite this tendency, DOC and DOF manage to maintain the lowest errors on these benchmarks.

Friedman's test conducted on test set fitness from Table 3 results in p-value of 8.02×10^{-10} and Table 6 demonstrates its post-hoc analysis. Observations are largely confirmed: DOC and DOF are both better than GP, while DOC and DOCLEX also significantly outperform AFPO.

Table 4 shows the average and 95% confidence interval of the number of nodes in the best-of-run programs. We are not surprised to see AFPO produce the smallest programs on average, as one of its motivations is to address the issue of bloat in GP. DOC comes second, with the rank of 2.27, and produces much smaller programs than the control methods. LEX, which ranked next after DOC in terms of fitness, turns out to produce much larger programs, even though its average run lengths do not diverge much from those of DOC (cf. Table 2). This observation is confirmed by the Friedman's test - DOC produces significantly smaller programs than LEX, DOCLEX, DOF and DOFLEX. LEX also seems to have slightly detrimental effect on program size when used in a hybrid approach with DOC.

In terms of pure wall clock times (cf. Table 2), AFPO takes the least mean time to run a single trial. DOC is more computationally expensive than tournament selection in GP, but slightly cheaper than LEX. DOF and DOFLEX are among the most expensive methods, most likely due to employing NMF.

7 DISCUSSION

The overall positive results corroborate the findings from previous works and extend them to continuous domains. The alternative, transient objectives derived automatically from interaction matrices by DOC, DOCLEX and DOF turn out to surpass a range of other methods on key performance indicators. Crucially, this holds even for LEX, which showed superior performance in multiple previous studies [5, 12, 17]. Acceptable runtimes that do not diverge much from those of more conventional methods strengthen this claim.

This outcome suggests also that the decisions we made when designing the preprocessing method described in Section 4, were largely appropriate. Indeed, preliminary experiments not reported here suggested that all key components of that method are essential: the 'capping' of maximum error with φ in Step 2 to emphasize the differences in low ranges of error, the standardization in Step 3 to provide for the same magnitude of outcomes on tests while maintaining their individual capability of discrimination between programs, and the sigmoid squeezing in Step 4 to map the outcomes in entire matrix to the same interval so that they are globally comparable for DOC and DOF. Sigmoid squeezing also reduces the impact of outliers on clustering in DOC and factorization in DOF.

It is encouraging to see that the low errors on the training set in most cases translate to test sets. Given that smaller programs often generalize better, this result can be in part attributed, particularly for DOC and DOCLEX, to moderate sizes of programs evolved by these methods. This is interesting, given that, except for AFPO, none of the setups considered here explicitly rewards smaller programs. It would be interesting to find out whether there are any other (than size) characteristics of the programs evolved with derived objectives that make them perform so well, and we consider this one of interesting follow-up directions.

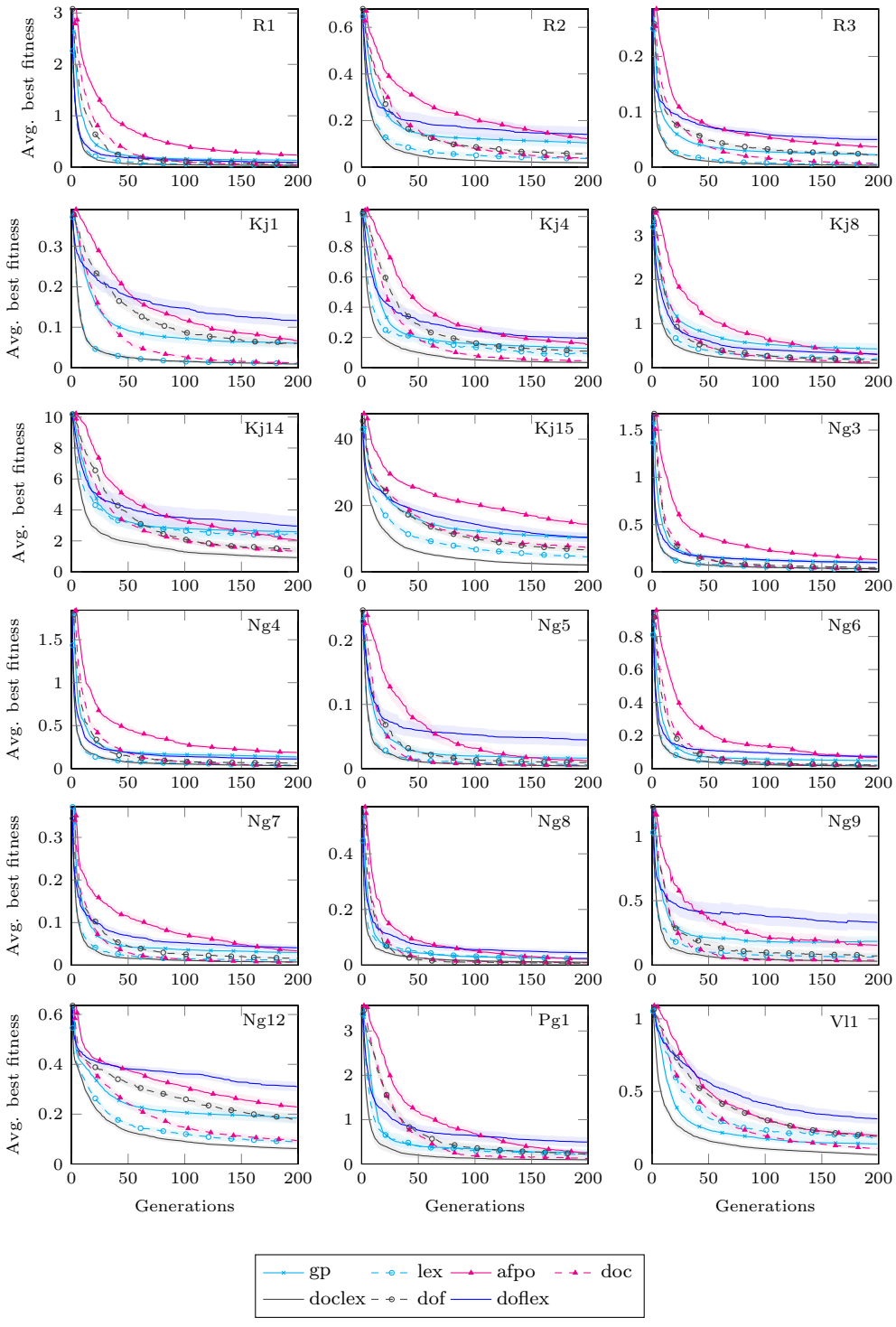

Figure 1: Average and .95-confidence interval of the best-of-generation fitness.

Table 2: Average and .95-confidence interval of the best-of-run fitness. Last rows present the averaged ranks of setups and the average time to run a single trial of each algorithm.

Problem	gp		lex		afpo		doc		doclex		dof		doflex	
R1	0.133	±0.027	0.039	±0.013	0.235	±0.025	0.046	±0.008	**0.034**	±0.011	0.076	±0.013	0.093	±0.013
R2	0.104	±0.020	0.037	±0.007	0.122	±0.014	0.040	±0.006	**0.019**	±0.003	0.057	±0.008	0.141	±0.032
R3	0.022	±0.004	0.005	±0.001	0.037	±0.003	0.007	±0.001	**0.003**	±0.001	0.022	±0.003	0.050	±0.006
Kj1	0.060	±0.013	0.010	±0.003	0.067	±0.009	0.012	±0.003	**0.009**	±0.002	0.060	±0.010	0.117	±0.015
Kj4	0.128	±0.021	0.092	±0.021	0.152	±0.022	0.045	±0.009	**0.034**	±0.006	0.110	±0.011	0.195	±0.040
Kj8	0.427	±0.128	0.208	±0.048	0.297	±0.059	0.126	±0.030	**0.099**	±0.020	0.174	±0.051	0.306	±0.056
Kj14	2.586	±0.463	2.468	±0.415	2.031	±0.242	1.354	±0.182	**0.923**	±0.153	1.457	±0.196	2.957	±0.630
Kj15	10.229	±1.380	4.517	±0.793	14.236	±1.189	7.426	±0.921	**2.069**	±0.324	6.629	±0.946	10.428	±1.432
Ng3	0.103	±0.022	**0.029**	±0.016	0.128	±0.016	0.030	±0.005	0.031	±0.010	0.045	±0.007	0.097	±0.024
Ng4	0.135	±0.035	0.040	±0.016	0.184	±0.018	**0.031**	±0.007	0.033	±0.010	0.064	±0.012	0.111	±0.019
Ng5	0.016	±0.006	0.006	±0.002	0.012	±0.003	0.005	±0.001	**0.004**	±0.001	0.009	±0.002	0.045	±0.010
Ng6	0.047	±0.012	0.021	±0.007	0.066	±0.012	**0.015**	±0.005	0.016	±0.004	0.022	±0.004	0.073	±0.013
Ng7	0.030	±0.010	0.012	±0.004	0.034	±0.006	**0.007**	±0.001	0.007	±0.002	0.016	±0.003	0.041	±0.007
Ng8	0.025	±0.005	0.024	±0.006	0.022	±0.005	0.009	±0.003	0.011	±0.003	**0.007**	±0.005	0.045	±0.009
Ng9	0.184	±0.042	0.061	±0.022	0.154	±0.048	0.038	±0.030	**0.030**	±0.011	0.072	±0.029	0.331	±0.065
Ng12	0.184	±0.021	0.089	±0.010	0.228	±0.020	0.094	±0.011	**0.062**	±0.008	0.180	±0.017	0.310	±0.019
Pg1	0.259	±0.079	0.221	±0.069	0.247	±0.046	0.121	±0.026	**0.094**	±0.042	0.223	±0.075	0.495	±0.112
Vl1	0.138	±0.018	0.189	±0.042	0.196	±0.018	0.107	±0.009	**0.063**	±0.011	0.191	±0.018	0.311	±0.036
Rank:	5.444		2.889		5.778		2.222		1.389		3.722		6.556	
Time:	00:08:50		00:18:32		00:01:58		00:12:41		00:17:18		00:23:04		00:41:43	

Table 3: Median and .95-confidence interval of test set fitness of the best-of-run programs.

Problem	gp		lex		afpo		doc		doclex		dof		doflex	
R1	0.192	±0.042	0.046	±0.017	0.266	±0.032	0.055	±0.009	**0.042**	±0.016	0.071	±0.012	0.124	±0.043
R2	0.127	±0.022	0.059	±0.017	0.141	±0.018	0.047	±0.008	**0.033**	±0.017	0.071	±0.012	0.184	±0.045
R3	0.043	±0.006	0.010	±0.003	0.052	±0.005	0.011	±0.002	**0.009**	±0.002	0.028	±0.004	0.068	±0.009
Kj1	0.126	±0.021	0.071	±0.027	0.108	±0.018	**0.031**	±0.009	0.044	±0.014	0.098	±0.017	0.180	±0.027
Kj4	0.555	±0.097	0.329	±0.068	0.291	±0.058	**0.164**	±0.052	0.282	±0.076	0.274	±0.052	0.684	±0.131
Kj8	210.659	±405.680	29.085	±37.352	1.029	±0.307	0.882	±0.273	3.235	±3.101	**0.639**	±0.237	1.218	±0.410
Kj14	3.785	±0.647	3.040	±0.541	2.516	±0.340	**1.548**	±0.179	1.672	±0.364	2.151	±0.425	4.264	±0.855
Kj15	11.380	±1.391	5.466	±1.063	14.601	±1.108	7.367	±0.831	**3.005**	±0.471	6.885	±0.910	12.243	±1.545
Ng3	0.120	±0.025	**0.033**	±0.017	0.146	±0.018	0.034	±0.006	0.034	±0.010	0.043	±0.006	0.099	±0.023
Ng4	0.173	±0.035	0.049	±0.019	0.205	±0.019	**0.035**	±0.010	0.047	±0.015	0.071	±0.014	0.133	±0.020
Ng5	0.024	±0.010	0.018	±0.009	0.011	±0.003	**0.005**	±0.001	0.005	±0.002	0.009	±0.002	0.053	±0.021
Ng6	0.046	±0.013	0.025	±0.011	0.037	±0.011	**0.008**	±0.004	0.016	±0.005	0.014	±0.004	0.070	±0.015
Ng7	0.040	±0.012	0.018	±0.007	0.038	±0.008	**0.012**	±0.005	0.066	±0.045	0.025	±0.008	0.051	±0.011
Ng8	0.129	±0.018	0.135	±0.032	0.077	±0.024	0.056	±0.016	0.114	±0.038	**0.024**	±0.015	0.176	±0.041
Ng9	0.211	±0.055	0.058	±0.032	0.079	±0.038	**0.012**	±0.012	0.131	±0.088	0.085	±0.041	0.501	±0.107
Ng12	0.265	±0.044	**0.187**	±0.029	0.282	±0.020	0.188	±0.031	0.220	±0.064	0.297	±0.023	0.448	±0.048
Pg1	2.085	±0.277	1.746	±0.272	1.580	±0.217	**1.170**	±0.168	1.475	±0.198	1.717	±0.222	2.252	±0.217
Vl1	0.636	±0.076	0.532	±0.079	0.608	±0.086	0.427	±0.043	**0.427**	±0.075	0.562	±0.052	0.831	±0.081
Rank:	5.667		3.333		4.889		1.722		2.667		3.333		6.389	

Last but not least, let us note that the overall underperformance of DOFLEX should be mainly attributed to the fact that for large r weights matrix becomes sparse. In such setting LEX tends to use only one case for each parent selection, resulting in poor performance. For DOF, no obvious means for automatic setting of factorization rank have been proposed to date, so we also experimented with the minimal value of $r = 2$. This proves to work reasonably well when combined with conventional DOF, as the NSGA-II selection is known to operate well in low-dimensional spaces. However for LEX this setting is also sub-optimal, as there is substantial conceptual and empirical evidence that this selection method yields the best results when the number of tests is at least in dozens.

8 CONCLUSIONS

In this study, we empirically generalized previous findings concerning methods that derive search objectives from interaction matrices. We may thus claim now that derived objectives, though heuristically derived and transient, are effective means of search not only for discrete domains, but also for the continuous ones. This elevates derived objectives to a fully-fledged concept for metaheuristics applied to test-based problems, including, but not limited to, GP. Given that many such methods are practically parameter-free, and so is the preprocessing method proposed in this paper, one may seriously consider including these solutions as out-of-the-box features in metaheuristic toolkits.

In a broader perspective, the results presented here form yet another argument for the quest for alternative means of driving search in heuristic algorithms [9]. Indeed, in many domains there are no conceptual nor technical obstacles for distilling more precise and useful information from candidate solutions. To that aim, we employed here the interaction matrix, but potential other approaches abound. Given the potential benefits evidenced in this paper, such opportunities should be exploited more often in research and practice of GP and other test-based problems, and we strongly encourage the readers to consider this path.

Acknowledgments P. Liskowski acknowledges support from grant 2014/15/N/ST6/04572 funded by the National Science Centre, Poland. K. Krawiec acknowledges support from grant 2014/15/B/ST6/05205 funded by the National Science Centre, Poland. The computations were performed in Poznan Supercomputing and Networking Center. The authors would like to thank Tomasz P. Pawlak for his advice and valuable comments.

Table 4: Average and .95-confidence interval of number of nodes in the best-of-run program.

Problem	gp		lex		afpo		doc		doclex		dof		doflex	
R1	92.437	±9.833	100.008	±11.078	**30.170**	±1.387	81.442	±7.024	101.085	±12.206	89.206	±4.571	87.903	±14.132
R2	99.776	±17.784	101.452	±9.665	**25.597**	±1.353	84.166	±6.028	110.236	±14.055	109.446	±6.626	90.809	±8.168
R3	112.906	±14.584	122.057	±10.600	**32.255**	±1.636	103.675	±7.762	123.575	±13.857	134.024	±9.053	122.974	±9.672
Kj1	111.203	±16.660	134.972	±16.185	**32.499**	±1.593	96.185	±9.052	109.394	±12.523	144.953	±10.278	148.799	±13.634
Kj4	133.811	±15.045	130.992	±9.754	**40.949**	±2.224	108.622	±9.683	134.536	±13.969	142.368	±10.577	146.039	±13.207
Kj8	150.517	±16.563	140.043	±11.708	**35.363**	±1.815	95.444	±7.903	147.541	±12.499	117.642	±8.253	131.048	±10.252
Kj14	89.812	±7.699	86.670	±7.108	**23.324**	±1.247	71.025	±3.976	98.103	±9.044	95.905	±9.338	110.179	±8.010
Kj15	110.263	±11.370	116.492	±8.230	**26.498**	±1.362	108.840	±7.486	134.830	±9.140	104.770	±6.949	136.477	±10.615
Ng3	89.343	±11.995	99.310	±12.007	**27.755**	±1.867	65.468	±5.011	100.422	±12.846	100.974	±8.165	94.524	±8.106
Ng4	91.692	±11.229	91.805	±9.541	**27.099**	±1.329	86.014	±9.434	98.851	±9.930	94.838	±6.475	93.499	±7.341
Ng5	60.719	±9.520	66.786	±10.729	**19.564**	±1.637	51.837	±6.405	67.796	±10.529	99.257	±12.233	84.411	±10.612
Ng6	82.674	±11.563	80.698	±7.670	**24.255**	±2.011	69.064	±8.436	83.287	±10.081	113.484	±16.390	96.361	±9.407
Ng7	66.389	±9.146	75.943	±6.520	**19.454**	±1.475	67.261	±5.456	90.062	±10.930	129.159	±15.665	89.042	±9.225
Ng8	63.756	±8.558	80.615	±10.483	**22.530**	±1.268	54.663	±6.226	75.417	±8.860	89.167	±10.359	81.057	±7.619
Ng9	90.664	±14.549	98.202	±13.235	**20.756**	±2.439	64.692	±11.441	87.062	±11.478	104.855	±12.879	98.217	±10.351
Ng12	83.121	±13.884	91.343	±6.956	**22.000**	±1.443	95.799	±6.730	116.942	±11.983	142.845	±14.956	104.046	±15.909
Pg1	64.462	±7.809	87.612	±7.157	**24.062**	±1.121	64.629	±3.858	90.902	±10.217	91.096	±8.086	110.165	±12.050
Vl1	107.003	±10.537	111.494	±10.925	**29.776**	±1.473	100.707	±6.986	126.004	±10.708	120.169	±7.465	121.183	±13.613
Rank:	3.556		4.278		1.000		2.278		5.556		5.833		5.500	

Table 5: Post-hoc analysis of Friedman's test on Table 2. Significant values ($\alpha = 0.05$) are in bold.

	gp	lex	afpo	doc	doclex	dof	doflex
gp			0.999				0.719
lex	**0.007**		**0.001**			0.910	**0.000**
afpo							0.934
doc	**0.000**	0.969	**0.000**			0.363	**0.000**
doclex	**0.000**	0.363	**0.000**	0.910		**0.020**	**0.000**
dof	0.201		0.065				**0.002**
doflex							

Table 6: Post-hoc analysis of Friedman's test on Table 3. Significant values ($\alpha = 0.05$) are in bold.

	gp	lex	afpo	doc	doclex	dof	doflex
gp							0.953
lex	**0.020**		0.317				**0.000**
afpo	0.934						0.363
doc	**0.000**	0.275	**0.000**		0.847	0.275	**0.000**
doclex	**0.001**	0.969	**0.033**			0.969	**0.000**
dof	**0.020**	1.000	0.317				**0.000**
doflex							

REFERENCES

[1] A. Bucci, J. B. Pollack, and E. de Jong. Automated extraction of problem structure. In K. D. et al., editor, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 501–512, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[2] R. Burden and J. Faires. Numerical analysis, cengage learning, 2010. Technical report, ISBN 978-0-538-73351-9, 1989.

[3] E. D. de Jong and A. Bucci. DECA: dimension extracting coevolutionary algorithm. In M. C. et al., editor, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 313–320, Seattle, Washington, USA, 2006. ACM Press.

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

[5] T. Helmuth, L. Spector, and J. Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, Oct. 2015.

[6] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[7] W. Jaśkowski and K. Krawiec. Formal analysis, hardness and algorithms for extracting internal structure of test-based problems. *Evolutionary Computation*, 19(4):639–671, 2011.

[8] G. K. Kanji. *100 statistical tests*. Sage, 2006.

[9] K. Krawiec. *Behavioral Program Synthesis with Genetic Programming*, volume 618 of *Studies in Computational Intelligence*. Springer International Publishing, 2015.

[10] K. Krawiec and P. Liskowski. 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, and K. Sim, editors, *18th European Conference on Genetic Programming*, volume 9025 of *LNCS*, pages 53–65, Copenhagen, 8-10 Apr. 2015. Springer.

[11] K. Krawiec and U.-M. O'Reilly. Behavioral programming: a broader and more detailed take on semantic GP. In C. Igel, editor, *GECCO '14: Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 935–942, Vancouver, BC, Canada, 12-16 July 2014. ACM. Best paper.

[12] W. La Cava, L. Spector, and K. Danai. Epsilon-lexicase selection for regression. In T. Friedrich, editor, *GECCO '16: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 741–748, Denver, USA, 20-24 July 2016. ACM.

[13] P. Liskowski and K. Krawiec. Discovery of implicit objectives by compression of interaction matrix in test-based problems. In *Parallel Problem Solving from Nature–PPSN XIII*, pages 611–620. Springer, 2014.

[14] P. Liskowski and K. Krawiec. Non-negative matrix factorization for unsupervised derivation of search objectives in genetic programming. In T. Friedrich, editor, *GECCO '16: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 749–756, Denver, USA, 20-24 July 2016. ACM.

[15] P. Liskowski and K. Krawiec. Online Discovery of Search Objectives for Test-based Problems. *Evolutionary Computation*, mar 2016.

[16] P. Liskowski and K. Krawiec. Surrogate fitness via factorization of interaction matrix. In M. I. Heywood, J. McDermott, M. Castelli, E. Costa, and K. Sim, editors, *EuroGP 2016: Proceedings of the 19th European Conference on Genetic Programming*, volume 9594 of *LNCS*, pages 68–82, Porto, Portugal, 30 Mar.–1 Apr. 2016. Springer Verlag.

[17] P. Liskowski, K. Krawiec, T. Helmuth, and L. Spector. Comparison of semantic-aware selection methods in genetic programming. In C. Johnson, K. Krawiec, A. Moraglio, and M. O'Neill, editors, *GECCO 2015 Semantic Methods in Genetic Programming (SMGP'15) Workshop*, pages 1301–1307, Madrid, Spain, 11-15 July 2015. ACM.

[18] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, et al. Genetic programming needs better benchmarks. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 791–798. ACM, 2012.

[19] R. I. B. McKay. Fitness sharing in genetic programming. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 435–442, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[20] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In C. A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31, Taormina, Italy, Sept. 1-5 2012. Springer.

[21] T. P. Pawlak, B. Wieloch, and K. Krawiec. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines*, 16(3):351–386, Sept. 2015.

[22] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[23] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, chapter 8, pages 129–146. Springer, Ann Arbor, USA, 20-22 May 2010.

[24] R. E. Smith, S. Forrest, and A. S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary computation*, 1(2):127–149, 1993.