Pre-print: to appear at PPSN 2018
The final authenticated version is available online at https://doi.org/[insert DOI]. (DOI not yet available)

Linear Combination of Distance Measures for Surrogate Models in Genetic Programming

Martin Zaefferer ✉[1], Jörg Stork[1], Oliver Flasch[2], and Thomas Bartz-Beielstein[1]

[1] TH Köln, Institute of Data Science, Engineering, and Analytics, Steinmüllerallee 6, 51643 Gummersbach, Germany, martin.zaefferer@th-koeln.de, joerg.stork@th-koeln.de, thomas.bartz-beielstein@th-koeln.de
[2] sourcewerk GmbH, Roseggerstraße 59, 44137 Dortmund, Germany
oliver.flasch@sourcewerk.de

Abstract. Surrogate models are a well established approach to reduce the number of expensive function evaluations in continuous optimization. In the context of genetic programming, surrogate modeling still poses a challenge, due to the complex genotype-phenotype relationships. We investigate how different genotypic and phenotypic distance measures can be used to learn Kriging models as surrogates. We compare the measures and suggest to use their linear combination in a kernel.
We test the resulting model in an optimization framework, using symbolic regression problem instances as a benchmark. Our experiments show that the model provides valuable information. Firstly, the model enables an improved optimization performance compared to a model-free algorithm. Furthermore, the model provides information on the contribution of different distance measures. The data indicates that a phenotypic distance measure is important during the early stages of an optimization run when less data is available. In contrast, genotypic measures, such as the tree edit distance, contribute more during the later stages.

Keywords: genetic programming, surrogate models, distance measures

1 Introduction

Genetic programming (GP) automatically evolves computer programs that aim to solve a task. This idea goes back to fundamental work by John Koza [1] and follows the principles of evolutionary computation. The computer programs are individuals subject to an evolutionary process, which improves them based on their fitness, i.e., their ability to solve a problem. Examples for GP tasks are symbolic regression (SR), classification, and production scheduling [2, 3].

Expensive fitness functions pose a challenge to evolutionary algorithms, including GP. This occurs, e.g., when the fitness function requires laboratory experiments or extensive simulations. Frequently, Surrogate Model-Based Optimization (SMBO) is used to deal with expensive evaluations [4]. Most SMBO research focuses on problems with continuous variables, where many competitive

regression models are available. In the context of GP, the use of surrogates is not well researched. This might seem surprising, as the computational bottleneck of most GP applications is the evaluation of fitness cases. Unfortunately, surrogate modeling of GP tasks, such as SR, is difficult, because it subsumes modeling of a complex genotype-phenotype-fitness mapping. Recent work in deep learning suggests that this mapping can be approximated, at least in certain domains of program synthesis [5].

In the last years, combinatorial search spaces were treated successfully with SMBO, by using distance-based models [6, 7]. However, there is no generic way for choosing an adequate distance measure. For complex tree shaped structures, which occur in GP, it is challenging to select a suitable distance measure and find a feasible modeling approach. For that reason, we will focus on the following research questions regarding SMBO for GP and tree-shaped structures:

1. How do different distance measures compare to each other?
2. What impact do these distances have on the model?
3. How does SMBO based on a linear combination of these distances compare to a model-free Evolutionary Algorithm (EA) and random search?

To answer these questions, we will utilize bi-level optimization problems based on different SR tasks as test functions. While these test functions are not that expensive to evaluate (and hence are not a natural use-case for surrogate models), they present a challenging benchmark for the proposed models. They allow us to gain insights into the topics summarized by our research questions. We expect that our result can be transferred to other problems with tree shaped structures, such as program synthesis for general purpose or domain-specific languages.

2 Related Work

In the following, we will differentiate between two approaches, which will be further referred to as a) SMBO and b) SAEA.

a) Sequential SMBO generates new candidate solutions by performing a search procedure on the surrogate model, e.g., as described for the Efficient Global Optimization (EGO) algorithm by Jones et al. [8].
b) Approaches that utilize surrogates to assist an EA (SAEA), e.g., as described by Jin [9]. For example, the surrogate is utilized to support the selection process of an EA by predicting the fitness of proposed offspring.

Most studies on GP and surrogate modeling focus on SAEA. Kattan and Ong [10] describe an SAEA approach with two distinct Radial Basis Function Network (RBFN) models (semantic and fitness). The conjunction of both models is used to evolve a subset of the population. They report superiority of their approach over standard GP for three different tasks, including SR.

Hildebrandt and Branke [11] present a phenotypic distance. They optimize job dispatching rules with an SAEA approach. Their surrogate model is a nearest neighbor regression model based on the phenotypic distance. They demonstrate that their model allows for a faster evolution of good solutions. This approach is also discussed and extended by Nguyen et al. [12, 13].

To the best of our knowledge, only Moraglio and Kattan [14] describe an SMBO approach to GP where a very limited number of function evaluations is allowed. They use an RBFN with appropriate distance measures. Their results did not indicate a significant improvement over the use of a model-free optimization approach.

In contrast to these works, we aim to learn Kriging models (following the idea of EGO [8]) and employ them in an SMBO framework with a severely limited number of 100 fitness function evaluations. Our models are based on a linear combination of three diverse distances. Like several of the above described studies, we use SR as a test case. We want to show that the relation between complex structures and their associated fitness can be learned and exploited for optimization purposes. Although SR is not particularly expensive, we argue that it presents a difficult and challenging test case to investigate whether our proposed models are able to learn such a complex search landscape.

3 A Test Case for SMBO-GP: Bi-level Symbolic Regression

In SR, a regression task is solved by evolving symbolic expressions. In essence, SR searches for a formula that best represents a given data set. The formulas can be represented by trees. Each tree consists of nodes and leaves, as well as the discrete labels on the nodes (mathematical operators, e.g., $+, -, *, /$) and leaves (variables and real-valued constants). Figure 1 shows the tree structure of the symbolic expression $\sqrt{c_1 - z_2} + (z_1 c_2)$. Our goal is to develop models that learn the relation between discrete tree structures and their fitness. For now, we are not interested in the influence of the real-valued constants. Hence, we suggest a bi-level problem definition.

3.1 Problem Definition

The upper level is the optimization of the discrete tree structure. For each fitness evaluation of the upper level, the lower level optimization problem has to be solved, which comprehends the optimization of the constants. Therefore, the upper level problem is defined by

$$\min_x F(x, c) \qquad \text{subject to} \quad c \in \arg\min_c f(x, c),$$

where x is the tree structure representation, $c \in \mathbb{R}^d$ is the set of d_c constant values, and $f(x, c)$ is the lower level objective function. Note, that the number of constants d_c depends on x. In extreme cases, the tree x may not contain any constants ($d_c = 0$), which eliminates the lower level problem. The fitness will be determined as

$$f(x, c) = 1 - |\text{cor}(\hat{y}(x, c), y)|, \tag{1}$$

where $\hat{y}(x, c)$ denotes the output of the symbolic expression for the data set, y is the corresponding vector of true observations, and $\text{cor}(\cdot, \cdot)$ is the Pearson

correlation coefficient. If $\hat{y}(x, c)$ becomes infeasible (e.g., due to a negative square root or division by zero), we assign a penalty value. To that end, we use the upper bound of our fitness function, $f_{\text{penalty}}(x, c) = 1$. An example of an upper level candidate's evaluation is visualized in Fig. 1. If not stated otherwise, fitness evaluations refer to evaluations of the upper level function F.

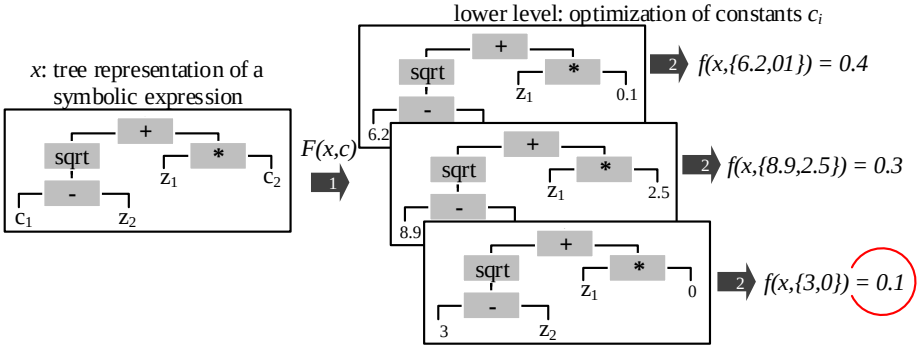

Fig. 1. Example for the upper level candidate $x = \sqrt{c_1 - z_2} + (z_1 c_2)$. To estimate its fitness $F(x, c)$, a lower level optimizer (step 1) estimates the fitness $f(x, c)$ for different constants (step 2) and returns the best to $F(x, c)$ (red circle).

3.2 Surrogate Model-Based Optimization

The SMBO approach we employ for the upper level optimization is loosely based on the EGO algorithm [8]. Initially, the search space is randomly sampled. The resulting data is used to learn a suitable regression model. This surrogate model is subject to a search via an optimization algorithm (e.g., an EA), which optimizes an infill criterion based on the model. An iteration ends with evaluating the actual (upper level) fitness of the new individual. Then, the surrogate model is updated with the new data and the procedure iterates.

As in standard EGO, we utilize a Kriging regression model, which assumes that the observed data is derived from a Gaussian process [15]. One reason for the popularity of Kriging in SMBO is that it allows to estimate its own uncertainty. The uncertainty estimate can be used to calculate the expected improvement (EI) infill criterion, which allows to balance exploitation and exploration in an optimization process [16, 8].

Importantly, Kriging is based on correlation measures or kernels, which describe the similarity of samples. Exponential kernels, e.g., $\mathrm{k}(x, x') = \exp(-\theta||x - x'||_2)$, with the parameter θ determined by Maximum Likelihood Estimation (MLE), are often used. It is straightforward to extend kernel-based models to combinatorial search spaces [6, 7]. The core idea is to replace the distance measure, e.g., in the exponential kernel $\mathrm{k}(x, x') = \exp(-\theta \mathrm{d}(x, x'))$. The distance

measure $\mathrm{d}(x, x')$ can be some adequate measure of distance between candidate solutions, such as an edit distance. Our study follows this idea. We will compare different distance measures and test how much they can contribute to Kriging models in an SMBO algorithm.

4 Kernels for Bi-level Symbolic Regression

We investigate four distance measures between trees or symbolic expressions, that will embedded into an exponential kernel.

4.1 Phenotypic Distance

The Phenotypic Distance (PhD) estimates the dissimilarity of two individuals (trees) based on their program output / phenotype, instead of using their code / genotype. This idea has been suggested by Hildebrandt and Branke for evolving dispatching rules via GP [11]. They defined a phenotypic dissimilarity by comparing the outcome of a decision rule based on a small set of test situations. Our SR tasks require a different definition of the phenotypic distance. We propose to measure the correlation between the outcomes of two symbolic expressions, with all numeric constants set to one. Hence, we save the effort of the optimization of the constants and compare the outputs of the expressions $\hat{y}(x, \mathbf{1})$ with

$$\mathrm{d}_{\mathrm{PhD}}(x, x') = 1 - |\mathrm{cor}(\hat{y}(x, \mathbf{1}), \hat{y}(x', \mathbf{1}))|.$$

If either of the two expressions is infeasible (e.g., due to division by zero), the distance will be set to one. Setting all constants to one is of course arbitrary. A random sample would also be possible but potentially problematic. A difference in phenotype could be perceived due to a different assignment of the constants on the leaves, rather than an actually different behavior of the symbolic expressions.

4.2 Tree Edit Distance

As an alternative to the PhD, we will also employ genotypic distances, i.e., distances between trees. One possible definition of distance between trees is the minimal number of edit operations required to transform one tree into another. This approach is denoted as the Tree Edit Distance (TED). We use the TED implementation that was introduced by Pawlik and Augsten [17]. It is available in the APTED library version 0.1.1 [18]. The APTED implementation counts the following edit operations: node deletion, node insertion, and node relabeling.

4.3 Structural Hamming Distance

The Structural Hamming Distance (SHD) [19] has been used to express genotypic dissimilarity for model-based GP in several studies [14, 10, 11]. Roughly speaking, it compares two trees by recursively checking each node that the two trees have

in common. To compare nodes, it uses the Hamming Distance (HD), which is one if two labels are different and zero otherwise. The original SHD (SHD1) is defined as

$$\mathrm{d}_{\mathrm{SHD1}}(x, x') = \begin{cases} 1, & \text{if arity}(x_0) \neq \text{arity}(x'_0) \\ \mathrm{HD}(x_0, x'_0), & \text{if arity}(x_0) = \text{arity}(x'_0) = 0 \\ \Delta(x, x'), & \text{if arity}(x_0) = \text{arity}(x'_0) = m, \end{cases}$$

with

$$\Delta(x, x') = \frac{1}{m+1} \left(\mathrm{HD}(x_0, x'_0) + \sum_{i=1}^{m} \mathrm{d}_{\mathrm{SHD1}}(x_i, x'_i) \right). \tag{2}$$

Here, x and x' are trees, x_0 indicates a root node of x, x_i with $i \geq 1$ is the i-th subtree of x, and arity(x_0) implies the number of subtrees linked to the corresponding node. We use a slight variation, which we refer to as SHD2. For the sake of simplicity, we define it for trees with a maximum arity of two. SHD1 and SHD2 are identical, except for the case arity(x_0) = arity$(x'_0) = m > 1$. Then, eq. (2) becomes

$$\begin{aligned} \Delta(x, x') = & \frac{1}{m+1} \Big(\mathrm{HD}(x_0, x'_0) + \\ & \min \left\{ \mathrm{d}_{\mathrm{SHD2}}(x_1, x'_1) + \mathrm{d}_{\mathrm{SHD2}}(x_2, x'_2), \mathrm{d}_{\mathrm{SHD2}}(x_1, x'_2) + \mathrm{d}_{\mathrm{SHD2}}(x_2, x'_1) \right\} \Big). \end{aligned}$$

That means, when two subtrees x_1, x_2 are compared with their counterparts x'_1, x'_2, we use the pairing or alignment between x and x' which yields the smaller distance. Potentially, this is more accurate, since it does not depend on the (arbitrary) initial alignment of the two trees. But SHD2 requires additional computational effort, even more so for larger arities.

The reason for using this modified variant lies in the nature of our SMBO algorithm. SAEAs yield datasets where some individuals will have common ancestors (or are ancestors of each other), and hence, are inherently more likely to be aligned with each other. Contrarily, SMBO generates new trees via a randomly initialized search that avoids direct ancestor relationships among individuals. This implies that two trees are more likely to have different alignments. Then, SHD2 is a potentially more accurate (but costly) measure.

4.4 Comparison and Linear Combination of Distances

For the comparison of the four different distance measures, we first calculated the distance matrices for 100 randomly generated trees (symbolic expressions). We used the same random tree-generation method as in Sec. 5. We computed the Pearson correlation between the different distance matrices. For this sample, the SHD variants yielded a strong correlation of 0.99, which indicates that they reflect very similar information. For the remaining samples, the correlation was 0.51 (PhD, SHD2), 0.29 (PhD, TED), and 0.37 (TED, SHD2). That is, the largest diversity was observed between PhD and TED. Figure 2 visualizes the

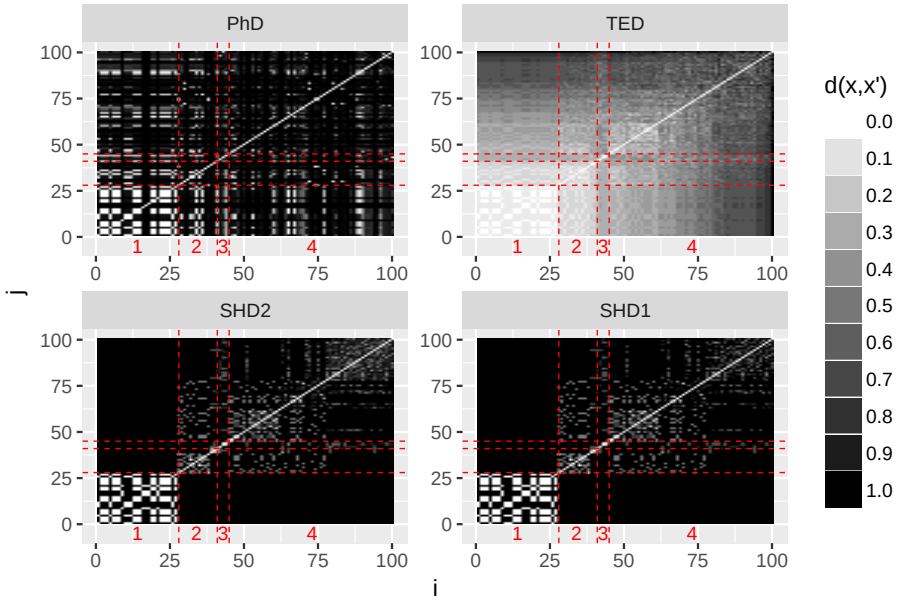

Fig. 2. Image plot of the four different tree distance measures. Each image cell is an element of a distance matrix. The trees are sorted by their complexity (tree depth and number of nodes). Trees in the lower left corner are less complex than those in the upper right. The tree depth is annotated in red at the bottom of each plot.

corresponding distance matrices. It shows that the SHD does have problems with differentiating between trees of different complexity. Several large blocks of the SHD matrices have a value of one, indicating that the respective trees are at maximum distance. This lack of perceiving a more fine-grained difference is problematic. It implies that any model based on SHD is potentially inaccurate for trees of a complexity that has not been observed so far. TED and PhD tend to see larger distances for more complex trees. This is obvious for TED, as complex trees require more operations to be transformed into each other. For PhD it is clear that complex trees can produce more diverse phenotypic behavior.

With regards to the computational effort, we note that TED is by far the most expensive measure. It is followed by the PhD and the cheapest measure is SHD1. While the specifics strongly depend on the implementation, we note that the TED required at least an order of magnitude more computation time than the others. This is not surprising, as determining the minimal number of edit operations requires to solve an optimization problem.

The PhD measure seems most promising in terms of generalizability. Most GP problems involve some phenotypic behavior that may be measured/compared. SHD and TED are limited to problems with tree structures and discrete labels.

The diversity of the different distances suggests that it is promising to combine them. We propose a linear combination of the PhD, TED, and SHD2. We

decided to focus on one of the SHD variants due to their similarity and chose the SHD2 variant due to its potentially increased accuracy. Also, its increased computational cost disappears compared to the larger costs of the TED. The linear combination in the kernel is

$$\mathrm{k}(x, x') = \exp\left\{-\beta_1 \mathrm{d}_{\mathrm{SHD2}}(x, x') - \beta_2 \mathrm{d}_{\mathrm{PhD}}(x, x') - \beta_3 \mathrm{d}_{\mathrm{TED}}(x, x')\right\}. \quad (3)$$

Each distance receives a weight $\beta_i \in \mathbb{R}^+$ that is determined by MLE. The linear combination allows for a potentially more accurate Kriging model. As we do not know a-priori which distance measure is appropriate for a certain problem (or whether they complement each other), the combination shifts this decision problem to the model. Furthermore, the weights provide insights into when and how much each distance contributes to the model.

5 Case Study

We performed a case study, testing the SMBO algorithm with six SR tasks.

Symbolic Regression Test Problems: We chose the Newton, sine-cosine, Kotanchek2D, and Salustowicz1D problems as used in [2] and the sqr and sqr+log problem as used in [10]. All problem configurations remained unchanged, i.e., operator set, data set size, and bounds for variables. We did not evaluate the derived symbolic expressions on an additional test set since our goal was to determine the ability of the SMBO algorithm to learn the connection between candidate solutions and fitness.

Lower level optimization of the constants: To optimize the lower level objective function, we decided to use the locally biased version of the the Dividing RECTangles (DIRECT) algorithm [20] for a global search. DIRECT uses $1000 \times d_c$ evaluations of the objective function. The result of the DIRECT run is further refined with a Nelder-Mead local search [21] (also $1000 \times d_c$ evaluations).

Upper level optimization of the structure: All algorithms received a budget of 100 upper-level objective function evaluations to emulate an expensive optimization problem. We used Random Search (RS) and a model-free EA as baselines. All operators were taken from the `rgp` package [22]. For creating new individuals, both baselines used `randfuncRampedHalfAndHalf`, parameterized with a maximum tree depth of 4 and a probability to generate constants of 0.2. Furthermore, the EA employed `crossoverexprFast` for recombination, which randomly exchanges subtrees. For mutation, `mutateSubtreeFast` was used. The parameters of the mutation operator are as follows: 0.1 (probability to insert a subtree), 0.1 (probability to delete a subtree), 0.1 (probability of creating a subtree instead of a leaf), 0.2 (constant generation probability), and 4 (maximum tree depth). Since constant values were not considered at the upper level, the respective bounds in the operator are both set to one. We employed a standard EA (based on `optimEA` in the `CEGO` package [23]) that used the above described operators. The EA used truncation selection, and a fixed number of children in each generation. The population size and number of children were tuned (see Sec. 5.1).

The upper level problem was also solved by the SMBO algorithm. We used the Kriging model from the `CEGO` package, with the kernel given in eq. (3). The model was trained within $1,000$ likelihood evaluations (via DIRECT). The EA searched on the surrogate model with $10,000$ evaluations of the EI criterion in each iteration. The SMBO search was initialized with 20 random trees.

For the analysis, we recorded the best individual for each run. In addition, we recorded the weights used for linear combination of the distances in each iteration, to evaluate the contribution of each distance function over time. Each algorithm run was repeated 20 times.

5.1 Algorithm Tuning

We decided to tune some potentially sensitive parameters to allow for a more fair comparison between the model-based and model-free algorithm. The model-free GP algorithm's population size μ and number of children λ produced in each iteration were tuned. All combinations of $\mu = \{5, 10, 15, 20\}$ and $\lambda = \{1, 2, 3, 4, 5\}$ were tested. The optimization performance was expected to be sensitive to these parameters, due to the extremely small fitness evaluation budget.

For the SMBO algorithm, we did not tune μ and λ. Due to the overall larger complexity we decided to set the parameters based on experience only, without a detailed tuning. In fact, due to the larger number of evaluations (of the surrogate model) the algorithm should be less sensitive to μ and related parameters. Since $10,000$ evaluations of the surrogate model were allowed, a (relative to the model-free EA) large $\mu = 200$ was given to the EA and correspondingly larger $\lambda = 10$.

We also performed preliminary experiments with the mean square error (MSE) instead of the correlation-based fitness measurement in equation (1). The MSE-based experiments yielded rather poor results with SMBO. This may be explained by the penalty for infeasible candidates. The penalty value is very difficult to set for the MSE case. A poor choice may severely impair the ability to train a good Kriging model because of strong jumps or plateaus in the fitness landscape. While our preliminary experiments were not very detailed, they can be counted as additional tuning effort, since they influenced the choice of the correlation measure used in the phenotypic distance.

5.2 Analysis and Discussion

Boxplots of the best observed fitness after 50 and 100 evaluations of the objective function F are shown in Fig. 3. We report results of the tuned, model-free EA that achieved the best mean rank on all problems ($\mu = 15$, $\lambda = 1$). The minimal λ makes sense, as it allows to perform a large number of iterations despite the small budget. For each problem and number of evaluations, we tested for statistical significance of the observed differences via the non-parametric Kruskal-Wallis rank sum test and Conover posthoc test, with a significance level of 0.05. The SMBO was significantly better than its two competitors in most cases, except for Salustowicz1D and Kotanchek2D after 100 evaluations, where no evidence

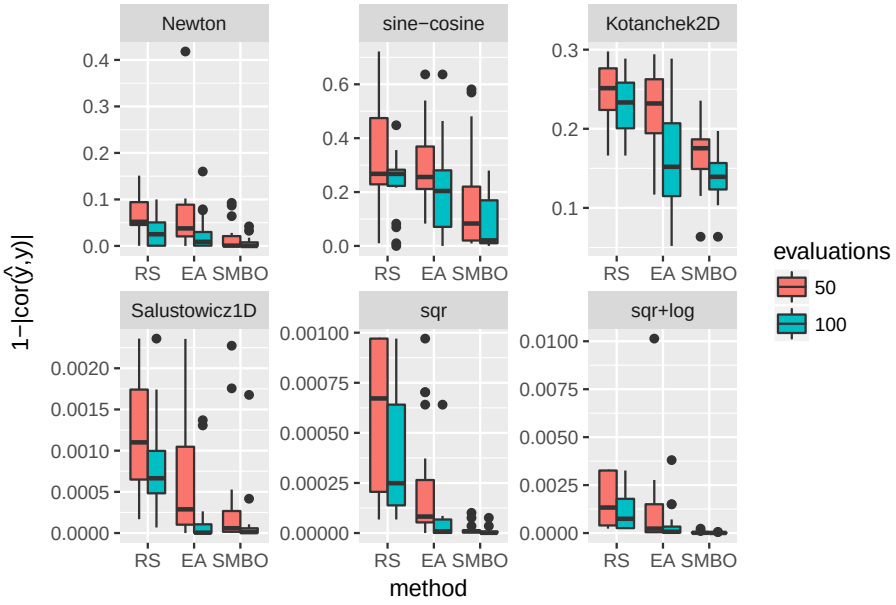

Fig. 3. Boxplot of best found values after 50 and 100 evaluations respectively.

for significant differences to the model-free EA is found. The EA was significantly better than the plain RS, except for Newton and sine-cosine (50 and 100 evaluations) as well as Kotanchek2D (50 evaluations).

To determine which distance measures contributed to these results, the weights of the linear combination are shown in Fig. 4. The weights are normalized so that they sum up to one. We show results for two problems, since they are similar in the other four cases. Usually, the PhD received the largest weights in the beginning, whereas the importance of the TED increased throughout the run, sometimes overtaking the PhD. SHD usually does not contribute as much, except for the sqr problem instance. Here, SHD overtakes both other distances at the end of the run. The generally larger importance of the PhD compared to SHD is in agreement with previous results by Hildebrandt and Branke [11], where a similar distance achieved better results than SHD.

We confirmed these results by additional optimization experiments for each single distance (i.e., without a linear combination). Runs with PhD tended to suggest good candidate solutions early, whereas TED and SHD performed better later on. The linear combination performed at least as well as the best of the single-distance models.

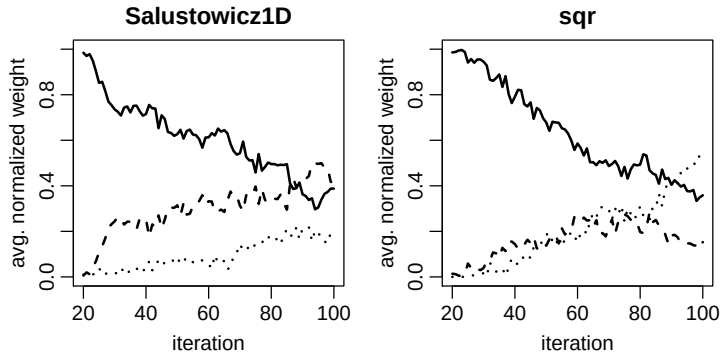

Fig. 4. Average normalized weights for the different kernels/distances. Solid line: PhD, dashed line: TED, dotted line: SHD2.

6 Conclusion and Outlook

We investigated whether three distance measures can be employed in an SMBO algorithm based on a Kriging model. We tested the algorithm with SR tasks. With respect to the research questions stated in Sec. 1, our results can be summarized as follows:

1. The distance measures PhD, SHD and TED are quite diverse. The SHD differentiates poorly between trees with different complexities. Especially the TED seems to be much more fine grained, but it requires the most computational effort. On the other hand, the PhD is comparatively cheap to evaluate and independent of the genotype.
2. Interestingly, the PhD seemed to contribute most, followed by the TED. This was especially true for small data sets at the beginning of an optimization run. Later on, TED and to a lesser extent SHD gained importance.
3. A Kriging model based on a linear combination of the three distances seems to be beneficial for SMBO. The SMBO algorithm outperformed a model-free algorithm and random search. All algorithms used no more than 100 fitness evaluations.

In future work, we would like to determine how well these results apply to other problem classes. Furthermore, alternatives to the linear combination of distances should be investigated.

References

1. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and computing **4**(2) (1994) 87–112
2. Flasch, O.: A modular genetic programming system. PhD thesis, TU Dortmund (2015)

3. Nguyen, S., Mei, Y., Zhang, M.: Genetic programming for production scheduling: a survey with a unified framework. Complex & Intelligent Systems **3**(1) (feb 2017) 41–66
4. Bartz-Beielstein, T., Zaefferer, M.: Model-based methods for continuous and discrete global optimization. Applied Soft Computing **55** (2017) 154 – 167
5. Parisotto, E., Mohamed, A., Singh, R., Li, L., Zhou, D., Kohli, P.: Neuro-symbolic program synthesis. ArXiv e-prints **1611.01855** (November 2016)
6. Moraglio, A., Kattan, A.: Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In: Proceedings of the 11th European Conference on Evolutionary Computation in Combinatorial Optimization. EvoCOP'11, Berlin, Heidelberg, Germany, Springer (2011) 142–154
7. Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., Bartz-Beielstein, T.: Efficient global optimization for combinatorial problems. In: Proceedings of the 2014 Genetic and Evolutionary Computation Conference. GECCO '14, New York, NY, USA, ACM (2014) 871–878
8. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4) (1998) 455–492
9. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2) (2011) 61–70
10. Kattan, A., Ong, Y.S.: Surrogate genetic programming: A semantic aware evolutionary search. Information Sciences **296** (2015) 345–359
11. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary Computation **23**(3) (Jun 2015) 343–367
12. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In: Simulated Evolution and Learning, 10th International Conference, SEAL, Springer Science + Business Media (2014) 656–667
13. Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE Transactions on Cybernetics (2016) 1–15
14. Moraglio, A., Kattan, A.: Geometric surrogate model based optimisation for genetic programming: Initial experiments. Technical report, University of Birmingham (2011)
15. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modelling. Wiley (2008)
16. Mockus, J., Tiesis, V., Zilinskas, A.: The application of Bayesian methods for seeking the extremum. In: Towards Global Optimization 2. North-Holland (1978) 117–129
17. Pawlik, M., Augsten, N.: Tree edit distance: Robust and memory-efficient. Information Systems **56** (mar 2016) 157–173
18. Pawlik, M., Augsten, N.: APTED release 0.1.1. GitHub. (2016) https://github.com/DatabaseGroup/apted, last accessed: 2017-06-01.
19. Moraglio, A., Poli, R.: Geometric landscape of homologous crossover for syntactic trees. In: 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, IEEE (2005)
20. Gablonsky, J., Kelley, C.: A locally-biased form of the direct algorithm. Journal of Global Optimization **21**(1) (September 2001) 27–37
21. Nelder, J.A., Mead, R.: A simplex method for function minimization. The Computer Journal **7**(4) (Jan 1965) 308–313
22. Flasch, O., Mersmann, O., Bartz-Beielstein, T., Stork, J., Zaefferer, M.: rgp: R genetic programming framework. (2014) R package version 0.4-1.

23. Zaefferer, M.: Combinatorial efficient global optimization in R - CEGO v2.2.0. online: https://cran.r-project.org/package=CEGO (2017) accessed: 2018-01-10.