Automatic Surrogate Modelling Technique Selection based on Features of Optimization Problems

Bhupinder Singh Saini
University of Jyvaskyla
Faculty of Information Technology,
P.O. Box 35 (Agora)
FI-40014 University of Jyvaskyla,
Finland
bhupinder.s.saini@jyu.fi

Manuel López-Ibáñez
Alliance Manchester Business School,
University of Manchester, UK
manuel.lopez-ibanez@manchester.ac.uk

Kaisa Miettinen
University of Jyvaskyla
Faculty of Information Technology,
P.O. Box 35 (Agora)
FI-40014 University of Jyvaskyla,
Finland
kaisa.miettinen@jyu.fi

ABSTRACT

A typical scenario when solving industrial single or multiobjective optimization problems is that no explicit formulation of the problem is available. Instead, a dataset containing vectors of decision variables together with their objective function value(s) is given and a surrogate model (or metamodel) is build from the data and used for optimization and decision-making. This data-driven optimization process strongly depends on the ability of the surrogate model to predict the objective value of decision variables not present in the original dataset. Therefore, the choice of surrogate modelling technique is crucial. While many surrogate modelling techniques have been discussed in the literature, there is no standard procedure that will select the best technique for a given problem.

In this work, we propose the automatic selection of a surrogate modelling technique based on exploratory landscape features of the optimization problem that underlies the given dataset. The overall idea is to learn offline from a large pool of benchmark problems, on which we can evaluate a large number of surrogate modelling techniques. When given a new dataset, features are used to select the most appropriate surrogate modelling technique. The preliminary experiments reported here suggest that the proposed automatic selector is able to identify high-accuracy surrogate models as long as an appropriate classifier is used for selection.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by regression**;

KEYWORDS

surrogate modelling, automatic algorithm selection, exploratory landscape analysis

ACM Reference Format:
Bhupinder Singh Saini, Manuel López-Ibáñez, and Kaisa Miettinen. 2019. Automatic Surrogate Modelling Technique Selection based on Features of Optimization Problems. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 8 pages.
https://doi.org/10.1145/3319619.3326890

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326890

1 INTRODUCTION

Increasingly, data is being used as the starting point of analysis of problems and optimization. Alternatives, such as running simulations or conducting physical experiments, may be computationally expensive, financially expensive or both. Data, on the other hand, can be analyzed cheaply and efficiently, while leading to equally relevant insights. One of the ways to use data is to create metamodels or surrogate models that try to replicate the behaviour of the simulations or real-life phenomena. Creating these surrogate models is comparatively cheap and the models can be used for descriptive, predictive or prescriptive analysis of the problem. Using surrogate models for prescriptive analysis, such as optimization of the problem, is particularly beneficial as optimization may require multiple calls to an expensive objective function, which can be replaced with a cheap surrogate model. However, accurate modelling of data is important for such analysis.

In many papers, there is no reasoning behind the selection of a surrogate modelling technique apart from the popularity of the technique in the research community and the researcher's familiarity with it. In the absence of an evidence-based guide to select a surrogate modelling technique, choices made based on experience or popularity may be far from optimal [2]. This may lead to inaccurate analysis, which at best is a waste of time and resources and, at worst, may lead to ill-informed decisions. A possible approach for selecting surrogate modelling techniques would be to train a large number of techniques in the given dataset, which may require considerable time and, in the worst case, overfit to the particular dataset. Cross-validation approaches may somewhat overcome this over-fitting. However, they require setting aside part of the available data for validation of the models. Not only the validation data could otherwise have been used for training, leading to a better modelling of the problem; but also a selection based on cross-validation only makes use of the available dataset when making a decision, without any knowledge about similar datasets or optimization problems. Hence, a better selection approach is desirable, which can help us choose a good surrogate modelling technique for a problem instance, without sacrificing useful data, while being computationally efficient.

A similar problem is faced when choosing an optimization algorithm to tackle an optimization problem, due to the large number of algorithms available and the lack of precise guidelines for choosing the most appropriate algorithm for a given problem. In the field of automatic algorithm selection [10], an optimization algorithm is selected for a given problem instance by computing features of the problem instance that help to predict the performance of the available algorithms. In particular, features generated using exploratory landscape analysis (ELA) have been used to automatically select optimization algorithms for continuous black-box optimization problems with promising results [6]. Similar techniques have been applied to combinatorial optimization problems [13]. Similar to the assumptions made by automatic algorithm selection methods, we expect that there are classes of optimization problems for which certain surrogate modelling techniques perform better than others.

Instead of selecting an optimization algorithm for a given optimization problem, we propose here to select a surrogate modelling technique for a given dataset by using classification techniques from machine learning. Moreover, by assuming that the dataset originates from an underlying but unknown optimization problem, we propose that the classifier learns from ELA features, among others. Our working hypothesis is that, due to common properties of the optimization problems underlying the datasets, ELA features may give enough information to identify a good surrogate modelling technique, if we can collect enough training data to characterize new, i.e., unseen problems.

The classifier is trained on known optimization benchmark functions from which it is easy to sample datasets with diverse features and evaluate the accuracy of various surrogate modelling techniques. The computational cost of this training phase may be large, but it is incurred only once "offline". When the automatic selection system is applied to an unseen dataset, the system only has to compute dataset features in order to select the appropriate surrogate modelling technique to model the dataset.

In this paper, we evaluate the above proposal using a collection of datasets generated from benchmark and real-world engineering problems from the literature. We also compare the performance of several classifiers when used as the selector of the proposed system. Our preliminary results show that the proposed system is able to select good surrogate modelling techniques for almost all engineering problems.

The paper is structured as follows. Section 2 describes the general idea of our proposal for automatic selection of surrogate modelling techniques. In Section 3, we evaluate this proposal experimentally on benchmark and real-world engineering problems by studying a proof-of-concept system that automatically chooses among ten surrogate modelling techniques. In addition, we compare the performance of nine classifiers as the selection method used by our automatic selection system. Finally, we summarise our conclusions in Section 4 and point out ongoing and future work.

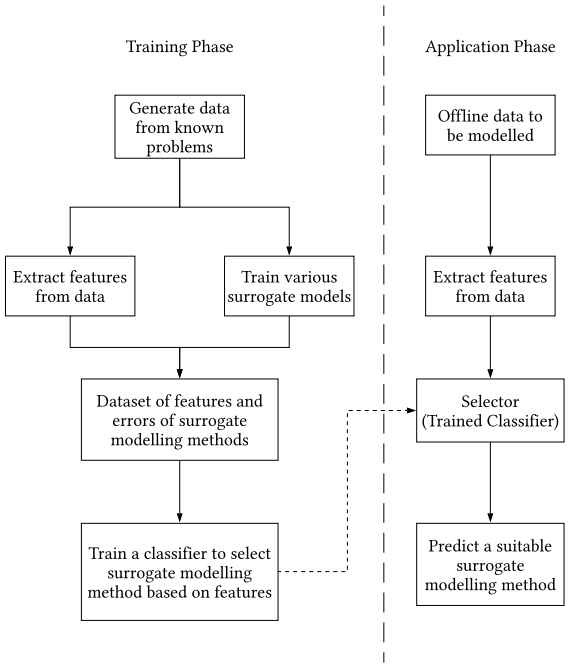

Figure 1: Automatic selection of surrogate modelling techniques based on optimization features.

2 AUTOMATIC SELECTION OF SURROGATE MODELLING TECHNIQUES

We propose here to automatically select the most appropriate technique for building a surrogate model of a given optimization problem based on features that have been traditionally used for landscape analysis of optimization algorithms. Surrogate modelling technique selection (SMTS) uses machine learning techniques to learn and predict which surrogate modelling techniques would perform best on given data based on certain "features" of the dataset rather than individual samples. In our SMTS framework, a classifier that we will call a "*selector*" is trained on landscape features of optimization problems and on the performance of various surrogate modelling techniques. Later, in an application phase, given a dataset to be modelled from an underlying optimization problem, the trained selector will select just one surrogate modelling technique based on features of the given data. By assuming a data-driven context, the application phase is constrained to the data already available and there is no possibility of generating new data from the same underlying problem to evaluate the performance of alternative surrogate modelling techniques.

An outline of the proposed automatic SMTS framework is shown in Figure 1. There are two clearly delimited training and application phases. The training phase proceeds as follows:

(1) **Data generation**: Training datasets are generated from well known optimization problems. Each training dataset corresponds to a sample of solutions and corresponding objective function values. To ensure the applicability of the SMTS to

a wide range of real-life problems, the individual datasets should differ not only in the characteristics of the problems, e.g., number of decision variables, but also in the features of the sampling: different numbers of samples, uniform vs. non-uniform distributions of samples in the decision space, presence of noise and/or missing data, etc.

(2) **Surrogate model training**: A previously chosen set of surrogate modelling techniques is trained on each dataset. These techniques will form the choices available to the selector. Then, the accuracy of each surrogate model is evaluated by using a sample of solutions and objectives values from the same problem but different from those used for training, and calculating its R^2 coefficient.

(3) **Feature calculation**: Independently of the training of each surrogate model, we also calculate a relatively large number of features for each training dataset that hopefully help to characterize the landscape of the optimization problem that underlies the dataset.

(4) **Training of the selector**: The features of each training dataset together with the R^2 value of each surrogate modelling technique on the problem forms the training data for a classifier that aims to predict the best surrogate modelling technique (i.e., with the highest R^2) for each dataset based on its features.

(5) **Custom cost function:** We have observed that there are large differences among various surrogate modelling techniques and, often, the second-best technique is almost as good as the best one, while both of them are significantly better than the worst one. In other words, when failing to select the best surrogate modelling technique, selecting instead the second-best is much better than selecting the worst one. Therefore, we use the following cost function to quantify the performance of the selector relative to the best-performing surrogate modelling technique:

$$Loss(Dataset_j) = \max_{k \in K}\{R^2_{k,j}\} - R^2_{\text{selected},j} \tag{1}$$

$$Cost = \frac{\sum_{j=1}^{J} Loss(Dataset_j)}{J}, \tag{2}$$

where

$Dataset_j$ = Dataset with index j
$R^2_{i,j}$ = R^2 value of surrogate modelling technique (SMT) i applied to dataset j
$R^2_{\text{selected},j}$ = R^2 value of the SMT selected by the selector for dataset j
K = Total number of SMTs available for selection
J = Total number of datasets included in cost evaluation.

The best-performing surrogate modelling technique on a particular dataset produces the maximum R^2 value and, if selected, it results in a loss value of zero. Selecting any other surrogate modelling technique will produce a higher loss value. The cost metric aggregates the loss values over multiple datasets.

Once the selector has been trained using the training datasets, it may be used to select a surrogate modelling technique on unseen data. In this application phase, we have access to some features of the sampling, such as the number of samples, presence of noise or missing data, etc. Since we assume that the unseen data arises from an unknown optimization problem, we can also calculate landscape features of the underlying optimization problem from the data available. However, we do not have the possibility of creating new data and we do not have access to the underlying optimization problem. Moreover, we do not attempt to train each surrogate modelling technique on the available data. Instead, we use the selector trained in the previous phase to select one surrogate modelling technique according to the features that can be computed on the given sample data. The selected surrogate modelling technique will then be trained on the available data and used for analysis, prediction and, possibly, for data-driven optimization.

3 EXPERIMENTAL STUDY

We evaluate here a proof-of-concept of our proposed automatic selection of surrogate modelling techniques based on features of optimization problems. The first goal of our experimental analysis is to understand whether there are sufficient differences among standard surrogate modelling techniques to justify the computational cost of training a selector. To answer this question, we consider 10 popular surrogate modelling techniques (Table 2) and evaluate them on datasets generated from standard numerical optimization benchmark functions.

A second question is whether training a selector in the manner proposed above can identify a good surrogate modelling technique for a given dataset, which is at least better than a random selection. Of course, an important component of the proposed selector is the classification method used for selection. Hence, we evaluate here nine different classifiers (Table 3). Moreover, a key characteristic of our proposal is that, given an unseen dataset, we wish to select a surrogate modelling technique based on features of the underlying optimization problem extracted from the available dataset, without training any of the surrogate models, not even on a sub-sample of the dataset. The latter could in principle help to inform the selection, but it may obscure the contribution of the landscape features, which is what we are testing here.

Finally, we also want to assess whether a selector trained on data from benchmark functions is better than chance in identifying a good surrogate modelling technique for real-world engineering problems.

3.1 Experimental Setup

Benchmark Datasets (Training). For the training phase, we generate datasets from well-known benchmark problems. In particular, we consider separately single functions from several multiobjective benchmark test suites (DTLZ [3], WFG [4] and ZDT [14]). The number of decision variables, i.e., the dimensionality of the decision space, was varied between 10, 20 and 30 for the DTLZ and WFG sets, and the recommended dimensionality for ZDT (30 in most cases).

Data from real-life problems may have characteristics that are not usually found in uniformly distributed datasets generated from benchmark functions. These may include skewed distributions of samples in the data, presence of noise, and chunks of the decision

Table 1: Characteristics of datasets generated from benchmark problems.

Benchmark problem family:	DTLZ functions {1–5}, ZDT funcs. {1–9}, WFG funcs. {1–4, 6}
Decision variables:	{10, 20, 30} (ZDT: only recommended values)
Number of samples:	{100, 150, 200, 250, 300, 350, 400, 500, 600, 700, 800, 900, 1000, 1200, 1500, 2000}
Distribution of samples (decision space):	{Uniform, Normal with $\mu = 0.5$, sd $= 0.25$}
Missing data:	{None, Missing}

space may be missing from the data. Datasets may also have too few samples for a surrogate modelling technique to work correctly, or some surrogate modelling techniques may benefit more than others from having a large number of samples. We account for these characteristics by sampling several training datasets from each benchmark problem. In particular, we sample datasets with various sizes ranging from 100 to 2 000 solutions. Samples are also generated either from a uniform random distribution in the decision space or they are normally distributed with a mean equal to 0.5 and standard deviation equal to 0.25 for each decision variable. Some datasets have chunks of data missing, which is done by creating the dataset as previously described, and then removing data which lie within a rectangular hyperbox one-tenth of the size of the decision space. Table 1 summarizes all the variations of training datasets included in our training phase.

Engineering Datasets (Validation). For evaluating the performance in the application phase, we generated datasets from several box-constrained engineering problems from the literature:

(1) Kursawe test function [7]
(2) Four bar plan truss problem [1]
(3) Gear train design [11]
(4) Pressure vessel design [11]
(5) Speed reducer problem [12]
(6) Welded beam design problem [11]
(7) Unconstrained function problem 1 [8]

In the case of multiobjective optimization problems, each objective function was treated as a separate problem, leading to a total of 12 problems. For each problem, we sample several datasets of sizes {50, 100, 150, 200, 400, 700, 1000}, sampling from the decision space uniformly at random. This resulted in a total of 84 datasets created from the engineering problems. The number of decision variables ranges from 3 to 7 and there is no missing data.

Dataset Features. In order to calculate landscape features of each dataset that hopefully characterize the underlying optimization problem, we used the R package `FLACCO` [5]. The features considered here belong to the set of "simple" or ELA metamodel features, which are calculated by creating linear and quadratic models on the dataset. The parameters and accuracy of these models, such as the intercept of the linear model or the adjusted R-square of the models, form the features of the dataset, resulting in a total of 10 features. In addition to the landscape features, three other features were included per dataset: the dimensionality of the decision space, the number of samples, and a Boolean variable representing whether the sampling was uniformly distributed or not. Table 4 describes all features.

Table 2: Surrogate modelling techniques available for automatic selection.

Surrogate modelling technique	Keyword
Support vector machine	SVR
Neural networks	NN
Adaptive boosted regression	Ada
Gaussian process regression	GPR
Stochastic gradient descent	SGD
K Nearest neighbour	KNR
Decision trees	DTR
Random forest	RFR
Extra trees regression	ExTR
Gradient boosted regression	GBR

Table 3: Classification methods evaluated for the selector.

Classifier method	Keyword
Bagging	BC
Support vector machine	SVC
K-nearest neighbor	KNC
Nearest centroid	NC
Gaussian process	GPC
Decision tree	DTC
Neural network	NNC
Extremely randomized tree	ExTC1
Extra-trees	ExTC2

Surrogate Modelling Techniques. We consider 10 popular surrogate modelling techniques (Table 2), from which the selector must choose one. In particular, we use the standard implementations of these techniques available in the Python package `scikit-learn` [9] with their default hyper-parameters. Surrogate models are trained and evaluated on one dataset at a time by randomly splitting the dataset into 70% solutions used for training and 30% solutions used to compute an R^2 value to measure the performance of the surrogate model.

Selector. In our proposal, the selector that chooses a surrogate modelling technique is a classifier. Since we do not have any intuition about which classifier may perform best for this task, we evaluate here nine alternative classifiers (see Table 3). We use the standard implementation of these classifiers available from `scikit-learn` with default hyper-parameters. To compare the classifiers, we use

Table 4: Dataset features used for selector training and application.

Feature Name	Description
ela_meta.lin_simple.adj_r2	Adjusted R^2 of a simple linear model
ela_meta.lin_simple.intercept	Intercept of a simple linear model
ela_meta.lin_simple.coef.min	Smallest non-intercept coefficient of the linear simple model
ela_meta.lin_simple.coef.max	Largest non-intercept coefficient of the linear simple model
ela_meta.lin_simple.coef.max_by_min	Ratio of the largest and smallest non-intercept coefficient of the linear simple model
ela_meta.lin_w_interact.adj_r2	Adjusted R^2 of a simple linear model with interactions
ela_meta.quad_simple.adj_r2	Adjusted R^2 of a simple quadratic model
ela_meta.quad_simple.cond	The ratio of the biggest and smallest cofficients of the simple quadratic model
ela_meta.quad_w_interact.adj_r2	Adjusted R^2 of a quadratic model with interactions
ela_meta.costs_runtime	Runtime for the computation of the features
numsamples	Number of samples in the problem instance
dimensionality	Dimensionality of the decision space of the problem instance
is_uniform	True if data is uniformly distributed in decision space, otherwise false

the custom cost function described by Eq. 2. The dataset features explained above together with the R^2 values of each surrogate modelling technique on each dataset are the input data for the classifier during the training phase, while only the dataset features are available to the classifier on the application phase when selecting a surrogate model.

Source Code. The SMTS code can be found at https://github.com/industrial-optimization-group/SurrogateAgents2/releases/tag/GECCO2019

3.2 Experimental Results

As the first experiment, we train each of the classifiers in Table 3 on 70% of the benchmark datasets and perform automatic selection of a surrogate modelling technique on the remaining 30% datasets to calculate a cost value.[1] We repeat this step 50 times with different random splits of the benchmark datasets, thus obtaining 50 cost values per classifier, which are shown as boxplots in Figure 2.

The plot shows that most classifiers were able to select a high-performing surrogate model, with cost values very close to zero. Of particular note are the BC, SVC, KNC, DTC, ExTC1 and ExTC2 classifiers, all of which had a cost value below 0.05 consistently. Given the random 70%/30% split of the datasets, the same exact dataset is not available for training and selection. However, the same underlying optimization problem may appear in both phases. Nevertheless, some classifiers performed very poorly, which suggests that selecting the best-performing classifier is not trivial.

In the second experiment, we focus on the best trained variant of each classifier from the first experiment and evaluate it on the engineering datasets. In other words, we choose as a selector for the application phase the trained version of each classifier with the lowest cost out of the 50 repetitions of the training phase performed above. We then use this selector to select surrogate modelling techniques for each engineering dataset. More specifically, a random (training) 70% subset of each engineering dataset is used to compute features, the classifier uses these features to select a surrogate model, the selected surrogate model is trained on the same training subset and its R^2 value is computed on the remainder 30% data of the dataset. We also calculate the R^2 value of all other surrogate modelling techniques so that we can identify the best-performing

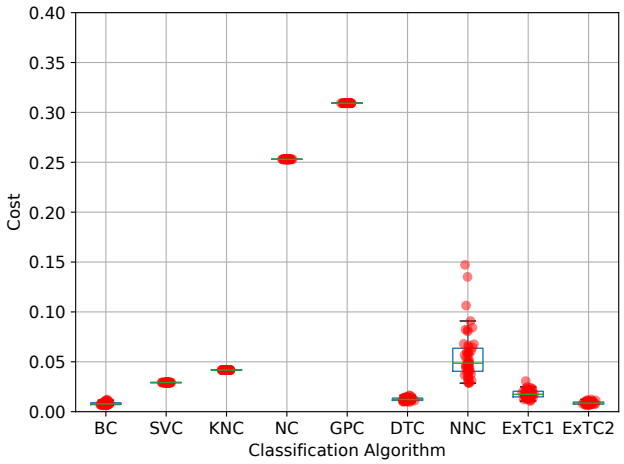

Figure 2: Cost values of each classifier on the benchmark datasets. Each point is the cost value of the surrogate modelling technique selected by the corresponding classifier for one random 70%/30% split of the benchmark datasets. Each boxplot contains 50 cost values.

[1] As explained above, when training and evaluating each surrogate modelling technique on each dataset, the dataset itself is also randomly split into 70% points used for training and 30% points used for computing R^2 values.

technique and compute the loss function (Eq. 1). Figure 3 shows the loss values of all the selectors for all 84 engineering datasets. In this case, obtaining a perfect score is much more difficult, as the selector was trained in samples of features generated from the benchmark datasets that are quite different from the feature samples generated from the engineering datasets. Nevertheless, the performance of the various classifiers follows a similar pattern as in the first experiment. In particular, BC, KNC, SVC, ExTC1 and ExTC2 again perform better than the rest.

Table 5 shows the means and sample standard deviations of the loss values of each classifier over all engineering datasets. By looking at the mean loss values we can say that the SVC selector performed the best with a mean value of 0.052 and a standard deviation of 0.096. This means that the classifier is selecting good surrogate modelling techniques, among the ones available, for most of the engineering problems. However, we noticed that the SVC selector always selected the same surrogate modelling technique (GBR) for all datasets. This technique belongs to the group of ensemble based techniques, all of which performed very well for most of the problems. By contrast, the second best selector, ExTC2, selected different surrogate models for different engineering datasets, and still achieved excellent loss values.

Table 5: Mean and standard deviation of loss values of selection on engineering data.

Classification method	Mean loss	Standard deviation of loss
BC	0.1072	0.2369
SVC	**0.0518**	**0.0963**
KNC	0.1798	0.3407
NC	0.3887	0.4153
GPC	0.7351	0.3723
DTC	0.2516	0.3972
NNC	0.1553	0.2730
ExTC1	0.0979	0.1895
ExTC2	**0.0853**	**0.1689**

Finally, we compare the actual R^2 values of the selected surrogate models versus the rest. As described above, all surrogate modelling techniques were trained on a random 70% subset of each engineering dataset and their R^2 value is calculated on the remainder 30%. Figure 4 shows the mean R^2 values (as lines, with a 95% confidence interval as a shaded area) of each surrogate modelling technique over all datasets generated from each engineering problem (x-axis). Good surrogate models have R^2 values close to 1 with a low variance. Interestingly, popular surrogate modelling techniques such as support vector machines (SVM), neural networks (NN), and Gaussian processes (GPR), had a significantly worse accuracy than ensemble methods such as extra-trees regression (ExTR) and gradient boosted regression (GBR).

The black line in Figure 4 shows the mean R^2 values obtained when the SVC classifier selects the surrogate modelling technique for each dataset. Figure 5 shows the same data and, instead, the black line indicates the mean R^2 values obtained by the surrogate modelling technique selected by ExTC2 classifier. These plots show

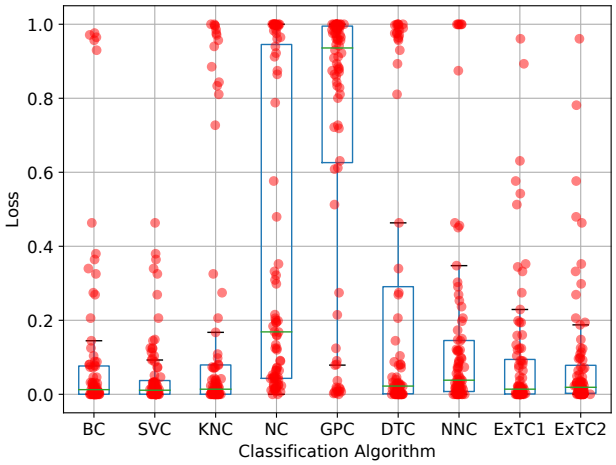

Figure 3: Loss values of each classifier on the benchmark datasets. Each point is the loss value of the surrogate modelling technique selected by the corresponding classifier for one engineering dataset after training the classifier on the benchmark datasets. Each boxplot contains 84 loss values.

that the SVC classifier has a lower mean loss value than ExTC2 because it performs significantly better in the `pressure_f1` and the `unconstrained_f_f1` problem. Although neither classifier is able to always select the surrogate model with the highest R^2 value, both are always able to select surrogate models with R^2 larger than 0.8, except for problems `gear_train_f1` and `unconstrained_f_f1`. Given the behavior of the surrogate models on these two problems, it seems likely that either our training phase does not have benchmark problems with the appropriate features, or we are lacking the features required to characterize these problems.

4 CONCLUSIONS

In this paper, we have proposed the automatic selection of surrogate modelling techniques for a given dataset by using features that aim at characterizing the underlying optimization problem. We describe here a proof-of-concept system that uses exploratory landscape features provided by the `FLACCO` package in addition to the dimensionality of the data, the number of points in the dataset and whether the dataset is a uniform sample or not. These features are used to select among ten available surrogate modelling techniques. For evaluating our proposal, we have generated a diverse set of datasets from popular benchmark functions as well as real-world engineering problems. In addition, we have compared nine different classifiers to be used as the selector of the proposed automatic surrogate model selection technique.

The preliminary experimental results presented in this work show that the choice of a classifier to be used as a selector is crucial, with significant differences in performance between classifiers. In addition, the best classifiers for benchmark datasets turned out to also perform well on engineering datasets. Another interesting result was that, despite the fact that no single surrogate modelling

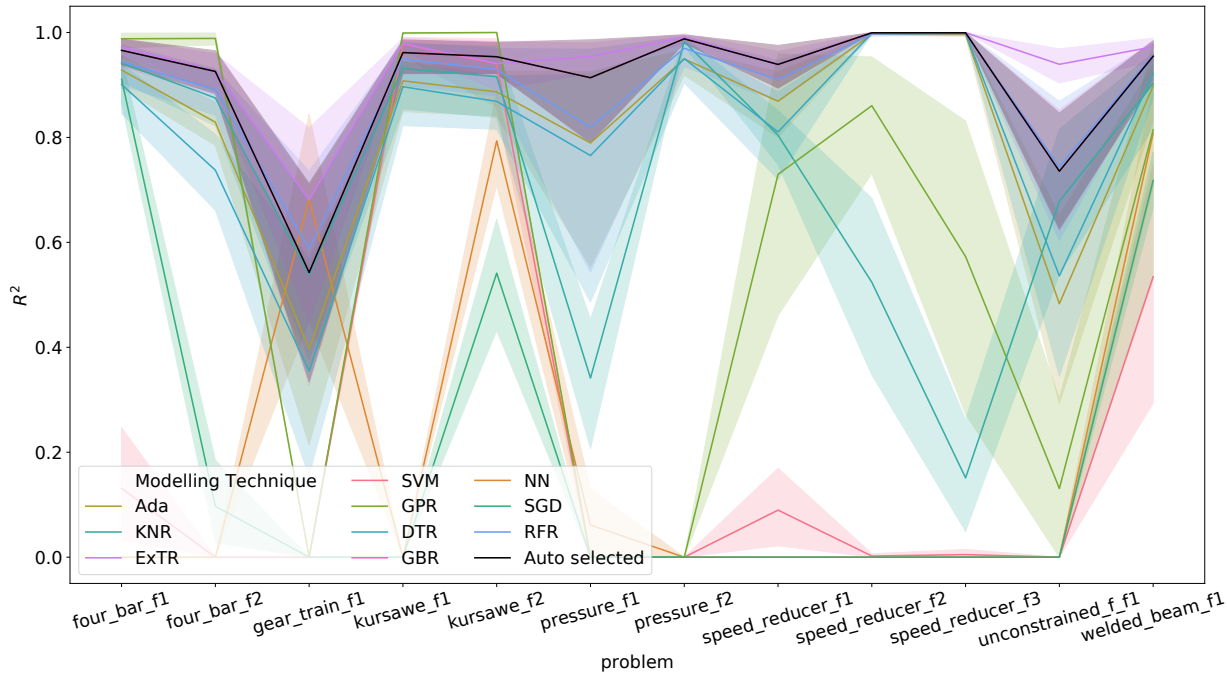

Figure 4: Mean R^2 values (and 95% confidence interval as shaded area) of each surrogate modelling technique when applied to all datasets generated from each engineering problem (x-axis). The black line indicates the mean R^2 value when SVC is used to select a surrogate modelling technique for each dataset.

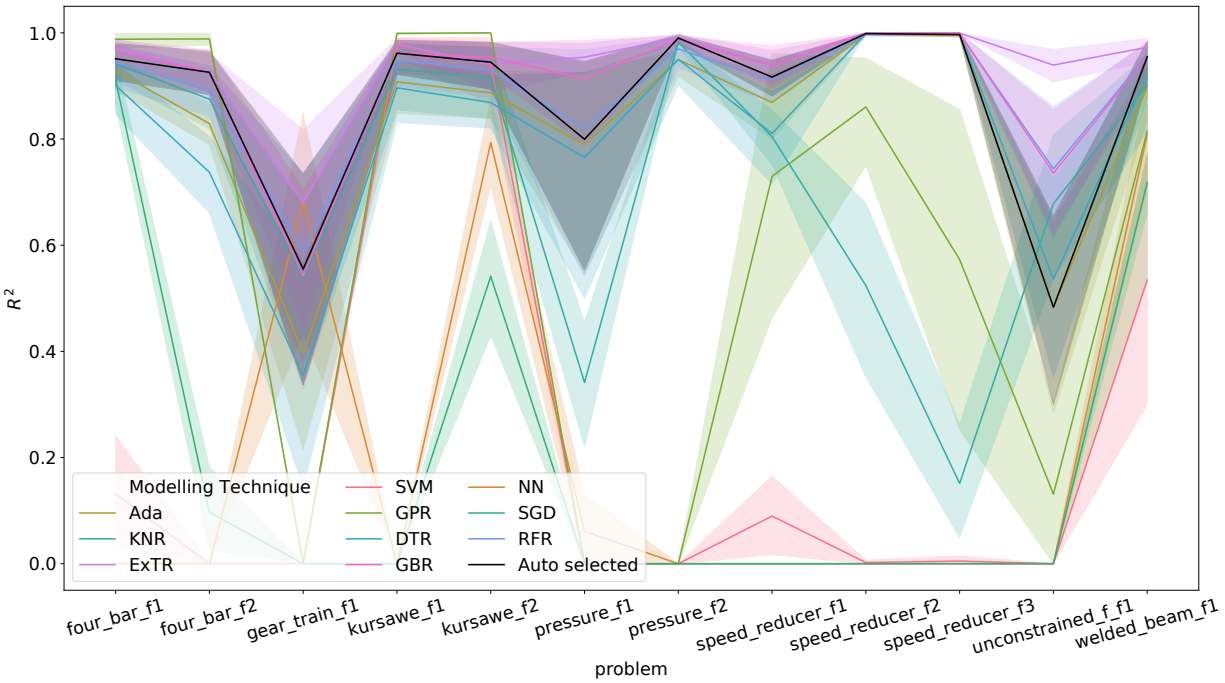

Figure 5: Mean R^2 values (and 95% confidence interval as shaded area) of each surrogate modelling technique when applied to all datasets generated from each engineering problem (x-axis). The black line indicates the mean R^2 value when ExTC2 is used to select a surrogate modelling technique for each dataset.

technique was always the best for all datasets, the automatic selection method was able to select surrogate modelling techniques with R^2 values higher than 0.8 in almost all datasets, when using the best or second-best classifier as a selector.

Although this research is at a very preliminary stage and there is much that could be improved, we believe that our results already show the potential of automatically selecting surrogate models for unseen data by training on datasets generated from known optimization problems.

Among the obvious improvements, we plan to simplify the creation and use of cross-validation sets for training and validating the surrogate models as well as the selector to make better use of the available data while keeping the benefits of separating training and validation data. In addition, we evaluated the various classifiers using a custom loss function. However, the training of the classifier could be itself cost-sensitive. Moreover, as surrogate modelling techniques are frequently used in conjunction with optimization algorithms, the quality of the optimal solutions obtained can also be used as a metric for training the Selector. The Selector, thus, is trained to select surrogate modelling techniques that perform well when used along with the considered optimization algorithms. We also considered "only" ten surrogate modelling techniques with default hyperparameter settings. Although this is a much larger number than what is usually considered in the data-driven surrogate modelling literature, it should be possible to select not only from an even larger pool, but also from various sets of hyperparameter settings. Moreover, we plan to analyze the importance of the features used by the selector. This will give us a better understanding of which characteristics of the problem instances are desirable as features, and will enable us to explore a much larger landscape of features productively. Finally, we plan to include more diverse training datasets, as our results suggest that our current training phase does not capture the characteristics of some real-world engineering problems.

ACKNOWLEDGMENTS

This research was partly supported by the Academy of Finland (grant number 287496, project DESDEO). This related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyvaskyla.

REFERENCES

[1] F. Y. Cheng and X. S. Li. 1999. Generalized center method for multiobjective engineering optimization. *Engineering Optimization* 31, 5 (1999), 641–661.
[2] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing* 23, 9 (2019), 3137–3166.
[3] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 2005. Scalable Test Problems for Evolutionary Multiobjective Optimization. In *Evolutionary Multiobjective Optimization*, A. Abraham et al. (Eds.). Springer, London, UK, 105–145.
[4] S. Huband, P. Hingston, L. Barone, and L. While. 2006. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation* 10, 5 (2006), 477–506.
[5] P. Kerschke and H. Trautmann. 2016. The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *Proceedings of the 2016 Congress on Evolutionary Computation (CEC 2016)*. IEEE Press, Piscataway, NJ, 5262–5269.
[6] P. Kerschke and H. Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation* 27, 1 (2019), 99–127.
[7] F. Kursawe. 1991. A variant of evolution strategies for vector optimization. In *Proceedings of PPSN-I, First International Conference on Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer (Eds.). Springer, Berlin, Heidelberg, 193–197.
[8] M. Mahdavi, M. Fesanghary, and E. Damangir. 2007. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* 188, 2 (2007), 1567–1579.
[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[10] J. R. Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65–118.
[11] E. Sandgren. 1990. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design* 112, 2 (1990), 223–229.
[12] J. N. Siddall. 1982. *Optimal Engineering Design: Principles and Applications*. Marcel Dekker Inc., New York.
[13] K. Smith-Miles. 2008. Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection. *Comput. Surveys* 41, 1 (2008), 1–25.
[14] E. Zitzler, L. Thiele, and K. Deb. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8, 2 (2000), 173–195.