Feature-based Benchmarking of Distance-based Multi/Many-objective Optimisation Problems: A Machine Learning Perspective

Arnaud Liefooghe[1][0000−0003−3283−3122], Sébastien Verel[2][0000−0003−1661−4093], Tinkle Chugh[3][0000−0001−5123−8148], Jonathan Fieldsend[3][0000−0002−0683−2583], Richard Allmendinger[4][0000−0003−1236−3143], and Kaisa Miettinen[5][0000−0003−1013−4689]

[1] Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
arnaud.liefooghe@univ-lille.fr
[2] Univ. Littoral Côte d'Opale, LISIC, F-62100 Calais, France
verel@univ-littoral.fr
[3] University of Exeter, Exeter EX4 4QD, U.K.
t.chugh@exeter.ac.uk, j.e.fieldsend@exeter.ac.uk
[4] The University of Manchester, Manchester M15 6PB, U.K.
richard.allmendinger@manchester.ac.uk
[5] University of Jyvaskyla, Faculty of Information Technology, FI-40014 University of Jyvaskyla, Finland
kaisa.miettinen@jyu.fi

Abstract. We consider the application of machine learning techniques to gain insights into the effect of problem features on algorithm performance, and to automate the task of algorithm selection for distance-based multi- and many-objective optimisation problems. This is the most extensive benchmark study of such problems to date. The problem features can be set directly by the problem generator, and include e.g. the number of variables, objectives, local fronts, and disconnected Pareto sets. Using 945 problem configurations (leading to 28 350 instances) of varying complexity, we find that the problem features and the available optimisation budget (i) affect the considered algorithms (NSGA-II, IBEA, MOEA/D, and random search) in different ways and that (ii) it is possible to recommend a relevant algorithm based on problem features.

Keywords: Multi/many-objective distance problems · Feature-based performance prediction · Automated algorithm selection.

1 Introduction

Given a collection of problems and algorithms, the algorithm selection problem [24] is concerned with identifying an algorithm that is most suitable, in terms of some performance criteria, for a given problem at hand. An additional component is the availability of features characterising a problem. One can first extract problem features to generate a feature space, and then act on it as opposed to on the problem space. Significant research has been carried out on

algorithm selection during the past two decades. Amongst others, tackling more efficiently a variety of continuous and mixed discrete/continuous [12,27], combinatorial [26] and multi-objective (continuous) optimisation problems [19], as well as supervised learning [22]. For algorithm selection, extracting problem features that drive algorithm performance is critical; doing this efficiently is the focus of fitness and exploratory landscape analysis [20,21]. Furthermore, understanding which and how problem features drive algorithm performance is valuable information when designing artificial problems of different complexity for benchmarking and tuning of algorithms. This information can then be used, e.g., to develop problem generators that can create test problems that meet user-defined problem characteristics. Such generators exist, for example, for many-objective distance-based optimisation [10] and cluster analysis [25].

Our focus is to advance the area of feature design and algorithm selection for multi- and many-objective optimisation problems. This is motivated by the prominence of problems in practice, combined with our limited understanding on suitable multi-objective problem features [12,20]. The most relevant work is given in [19], where features from landscape analysis (adapted from [17]) are applied on a benchmark set of 1 200 randomly-generated bi-objective interpolated continuous optimisation problems [30]. The study concluded that combining a classification model with a range of landscape features used as predictors can deliver a similar accuracy to predicting algorithm performance based on parameters used to generate the problems. It also investigated the relative importance of features for performance prediction and algorithm selection.

In this paper, we follow a similar approach to investigate the predictive power of parameters (problem features) used to generate distance-based multi/many-objective point problems (DBMOPPs) proposed in [10]. More precisely, we (i) generate 945 problems with different characteristics (as defined by 7 problem features), then (ii) test the correlation between the problem features and the performance of three popular multi-objective evolutionary algorithms and one baseline approach (random search), and finally assess the problem features as predictors for (iii) algorithm performance prediction and (iv) algorithm selection on machine learning regression and classification, respectively. This is the first study of DBMOPPs and the generator/problem features proposed in [10].

The paper is organised as follows. Section 2 introduces DBMOPPs, together with the generator and its parameters (problem features) used to control the generation of such problems. Section 3 gives the experimental setup and discusses algorithm performance. Section 4 presents an experimental analyses of the problem features for automated performance prediction and algorithm selection. Finally, Section 5 concludes the paper and discusses further research.

2 Distance-based Multi/Many-objective Problems

In multi-objective optimisation (with 2 or 3 competing objectives) and many-objective optimisation (with 4+ objectives), a *set* of solutions is typically sought that approximates the optimal trade-off combinations between the objectives,

given any constraints on *decision vectors*. Formally, the tuple of a decision vector $\mathbf{x}$, and its evaluation under the objective vector $\mathbf{f}()$ define a *solution* $s = (\mathbf{x}, \mathbf{f}(\mathbf{x}))$. A solution s is said to dominate another s' if s performs better than s' on at least one objective, and no worse on all others. The maximal set of non-dominated decision vectors is known as the Pareto set, and its image in the objective space is known as the Pareto front.

A wide range of scalable test problem frameworks have been developed for multi- and many-objective optimisation, which are used (together with set quality indicators) to assess the performance of optimisers. These encapsulate a range of known problem characteristics (e.g. [3,6,11]). In addition, means for generating *instances* of problems have been created to prevent "tuning" of optimisers to particular suites of tests, to the detriment of performance on practical problems.

Frameworks for DBMOPPs have been developed over the last decade. They incorporate the range of features exhibited in other test suites (constraints, neutrality, multi-modality, dominance resistance regions, local fronts, etc.) and enable direct visualising of the search space in a plane. Initial work in this area includes [13,14], and [10] includes a summary of the features incorporated into this test problem design approach over the last 15 years. Arbitrarily many objectives can be defined. If $|\mathbf{x}| > 2$ the decision vector is projected into two dimensions via a pair of orthogonal vectors prior to function evaluation. We now describe the construction of DBMOPPs, and the generator we recently developed. We will then extensively investigate its characteristics.

Properties and Features of DBMOPPs. Point-based distance problems are parameterised by sets of attractor vectors, where the minimum distance to a member of the ith set, V_i, defines the ith objective value:

$$f_i(\mathbf{x}) = \min_{\mathbf{v} \in V_i} \texttt{dist}(\mathbf{v}, \mathbf{x}). \tag{1}$$

Further complexity can be added by imposing regions of constraint violation which locally adjust the distance function, and thereby can introduce discontinuities and neutrality to particular objectives, amongst other modifications.

In [10], we introduced a DBMOPP[6] instance generator, where problems can incorporate a range of properties, a subset of which are listed in Table 1 which we consider here. An example 3-objective problem with local fronts and dominance resistance regions is illustrated in Figure 1, along with its local dominance landscape [9], PLOS-net [18] and PLON [8] network visualisations. In this work, we focus on box-constrained problems.

3 Experimental Setup

This section describes our experimental setup covering the approach adopted to generate problems, algorithms and their settings, and performance metrics.

[6] Available in `Matlab` (https://github.com/fieldsend/DBMOPP_generator), and in `Python` (https://github.com/industrial-optimization-group/desdeo-problem/tree/master/desdeo_problem/testproblems/DBMOPP).

Table 1. Problem features considered from the DBMOPP generator [10].

description	name	domain
number of variables	`n_var`	$[\![2, 20]\!]$
number of objectives	`n_obj`	$[\![2, 10]\!]$
non-identical Pareto sets	`nonident_ps`	{`no`, `yes`}
varying density	`var_density`	{`no`, `yes`}
number of disconnected Pareto sets	`n_discon_ps`	$[\![0, 6]\!]$
number of local fronts	`n_local_fronts`	$[\![0, 6]\!]$
number of dominance resistance regions	`n_resist_regions`	$[\![0, 6]\!]$

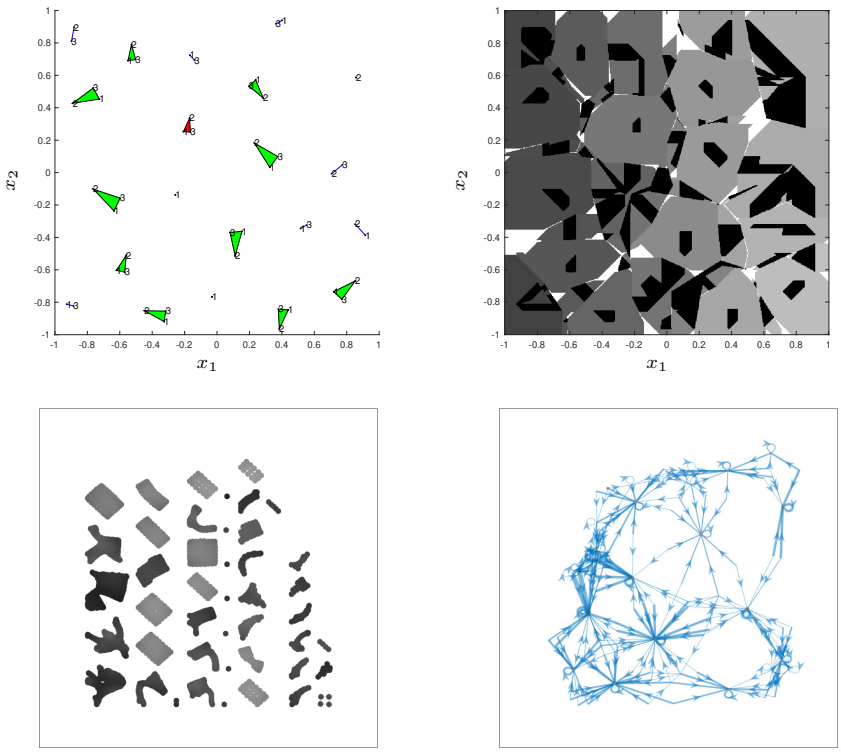

Fig. 1. *Top-left*: an example 3-objective problem with regions creating local fronts (green triangles), dominance resistance regions (points and lines) and Pareto set (red triangle). *Top-right*: its corresponding local dominance landscape – black regions are locally dominance neutral, shaded grey regions denote basins (for all basin members all neighbouring dominating moves lead to the same dominance neutral region), and white regions denote locations where immediate neighbours lead to different basins (saddles). *Bottom-left*: PLOS-net and *Bottom-right*: PLON network visualisations.

Table 2. Setting of the population size according to the number of objectives.

number of objectives	2	3	4	5	6	7	8	9	10
population size	100	105	120	126	132	112	156	90	275

3.1 Dataset

We generated 945 problems with different parameter settings. A random latin hypercube sample [4] of size 1 000 was generated, among which 55 problems were discarded due to restrictions in the benchmark generator. For each parameter setting, 30 instances (folds) were independently created using the generator and thus the total number of instances was $945 \times 30 = 28\,350$. Features used to create the problems are provided in Table 1. All 30 folds for a given problem had the same complexity in terms of features. However, folds could be different from each other because of the randomness in the generator. For each fold, we ran different algorithms (one run per instance): NSGA-II [5], IBEA [32] with the ε indicator, MOEA/D [31] with the Chebyshev scalarising function and random search for up to 50 000 evaluations. For a fair comparison, we kept the same initial population for a fold and used the same population size. It was selected based on the number of objectives and is given in Table 2. We employed an out of the shelf implementation with simulated binary crossover and polynomial mutation with probability of 0.8 and $\frac{1}{\texttt{n\_var}}$ and distribution index of 20 and 20, respectively; $\kappa = 0.5$ for IBEA. For each algorithm, we calculated the normalised hypervolume (hypervolume of final solutions/hypervolume of the Pareto front[7]) after 5 000, 10 000, 30 000 and 50 000 evaluations for all 30 folds of each problem. Normalised hypervolume values were then averaged for each of the 945 problems and 4 algorithms. The code is available at: https://github.com/tichugh/Feature_Analysis_DBMOPP_EMO_2023, and the corresponding dataset at: https://doi.org/10.5281/zenodo.7155803.

3.2 Algorithm Performance

We analysed results with `R` [23] using the `caret` [15], `rpart` [28] and `ggplot2` [29] packages. In Figure 2 (left), we show the average normalised hypervolume (and the 95% confidence interval) for each algorithm with respect to the search budget over all considered 945 problems. Figure 2 (right) gives the proportion of problems where each algorithm obtained the best average performance, over the 30 folds, for the considered budgets. We observe that IBEA was consistently the best-performing algorithm, whatever the budget, and outperformed others in at least 50% of problems (for a budget of 5 000 evaluations), and at most 80% (10 000 evaluations). It was followed by NSGA-II (almost as good as IBEA for

[7] 1 000 members drawn from the Pareto front plus all non-dominated points found by the union of the algorithms' approximation sets for each instance. The reference point for hypervolume was 1.1 × maximum of objective values on the Pareto front and estimated via Monte Carlo [7] with 50 000 samples for 4+ objectives.

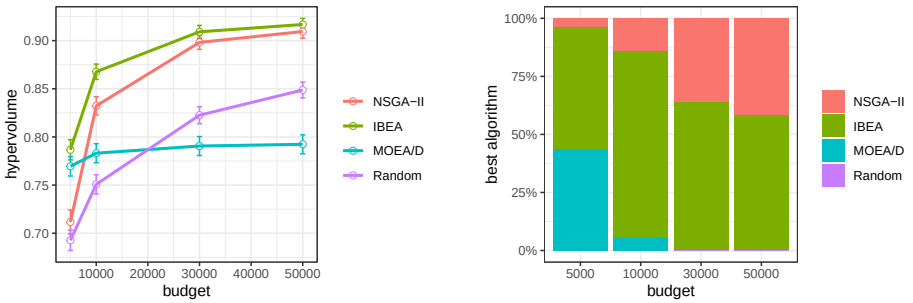

Fig. 2. Hypervolume (left) and proportion of problems where each algorithm obtains the best average hypervolume (right) with respect to the search budget.

50 000 evaluations). However, one should note that NSGA-II was not significantly better than random search for 5 000 evaluations. MOEA/D was efficient for the budget 5 000, but the increase of budget did not improve its performance as much as for the others. Indeed, the average hypervolume obtained by MOEA/D went from 0.77 for a budget of 5 000 to 0.79 for 50 000. A similar observation was reported in [10]. We conjecture that MOEA/D is significantly impacted by an increase in local Pareto fronts and multi-modality. Random search was dominated by other algorithms for the two lowest budgets. However, surprisingly, it surpassed MOEA/D for the highest budgets. In fact, random search was even the best for 1 problem for a budget of 30 000, and 2 problems for 50 000.

Interestingly, whatever the budget, there is no algorithm that outperforms the others for all problems. For the smallest budgets, IBEA and MOEA/D share the success almost equally on more than 95% of the 945 problems. For the largest budgets, NSGA-II and IBEA share the success on more than 99% of problems.

4 Experimental Study

This section uses a machine learning perspective to investigate the relationship of problem features and algorithm performance, the predictive power of features for performance prediction, and classification for feature-based algorithm selection.

4.1 Problem Features *vs.* Algorithm Performance

We first investigate how problem features impact search performance. Figure 3 shows how the normalised hypervolume of algorithms is individually impacted by each of the 7 problem features. Due to space restrictions, we report only two budgets. In addition, Figure 4 gives the Spearman's rank correlation coefficient between each problem feature and algorithm performance for all budgets. A larger hypervolume indicates a better performance and thus a positive correlation means that the problem feature has a favourable effect on algorithm performance.

For a given problem feature, the trend is similar for all algorithms and budgets (there is no feature with a positive effect for one algorithm and a negative effect for another algorithm). The same goes for the budgets. However, the strength

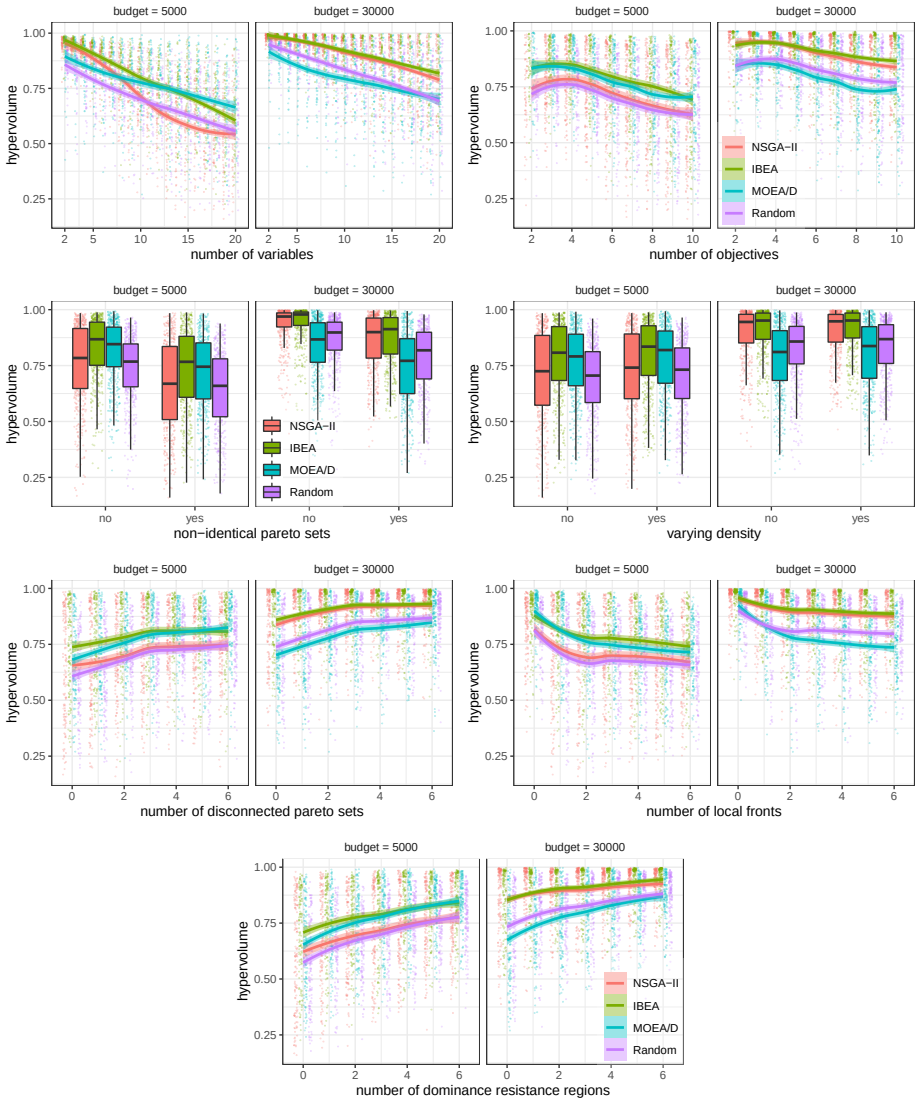

Fig. 3. Hypervolume vs. each problem feature and 5 000 and 30 000 evaluations.

of correlation is at times different. For instance, although a larger number of objectives (`n_obj`) means a worse performance for all algorithms and budgets, it is more impactful for NSGA-II than for other algorithms for large budgets.

The more variables (`n_var`), the worst the performance. However we see in Figure 3 (top-left) that, for a budget of 5 000, NSGA-II is good for a small number of variables, and becomes worse than random search as the number of variables increases. Correspondingly, while the number of objectives and hav-

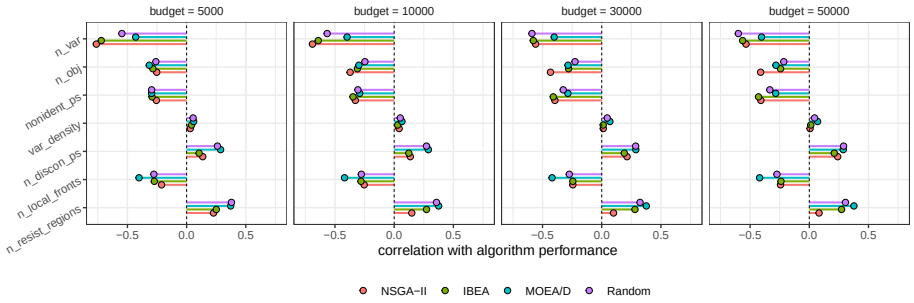

Fig. 4. Correlation between problem features and algorithm performance.

ing non-identical Pareto sets (`nonident_ps`) negatively impact performance, the number of dominance resistance regions (`n_resist_regions`) has a positive effect. For a budget of 30 000, NSGA-II is as good as IBEA with few objectives and few dominance resistance regions, but IBEA gets better as both numbers increase. Besides, the fewer local fronts (`n_local_fronts`) and disconnected Pareto sets (`n_discon_ps`), the better the performance of all algorithms with all budgets. The varying density (`var_density`) has a minor impact on performance. Overall, problem features in the 945 problems often imply a significant difference between algorithms independently of other features.

4.2 Performance Prediction by Regression

The previous results concerned the *individual* effect of problem features on performance. We now investigate their *combined* effect by constructing a regression model for predicting the hypervolume reached by the algorithms under different budgets. We thus end up with 4 (algorithms) $\times$ 4 (budgets) = 16 models aiming at predicting performance separately for each algorithm and budget, using the problem features as predictors. We consider random forest [1,16] with default parameters, a well-established state-of the-art ensemble learning method that constructs multiple decision trees for regression. We start by evaluating the prediction accuracy of the trained models using 30 independent replicates of 10-fold cross-validation. We report the repeated cross-validated coefficient of determination (R^2) for each algorithm and budget in Figure 5. We observe that the smallest R^2 obtained over all folds and repetitions is above 0.7, and the median R^2 is always above 0.85, whatever the algorithm and budget. This suggests that more than 85% of the variance in hypervolume values across all problems is explained by the model, and thus by problem features. The slight drop in the prediction accuracy for IBEA and NSGA-II as the budget increases is not significant and the R^2 values remain quite satisfactory. The prediction accuracy for random search is particularly high, regardless of the budget.

For each regression model, we also compute the importance of predictors, commonly measured as the mean decrease of prediction accuracy with random

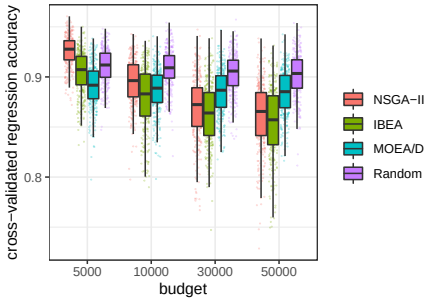

Fig. 5. Coefficient of determination (R^2) of regression models trained by algorithm and budget, calculated using 10-fold cross-validation with 30 repetitions.

forest [1,16]. The higher the value, the more important the predictor. Figure 6 shows the relative importance of problem features, scaled between 0 and 100. Overall, the most important problem features remain quite consistent with their correlation with algorithm performance reported in Section 4.1. The numbers of variables, objectives, dominance resistance regions and the presence of non-identical Pareto sets appear on top of the list, whereas the varying density only has a marginal contribution to the prediction accuracy. Interestingly, the presence of non-identical Pareto sets and the number of disconnected Pareto sets get more important as the budget increases. However, noticeable differences appear for MOEA/D, for which the numbers of local fronts and dominance resistance regions are highly important, regardless of the budget. They even surpass the number of variables as the most important features for larger budgets. Thus, it is interesting to note that even though the problem features coming from the problem generator have a similar effect on performance overall, their strength may be quite different depending on the considered algorithm and budget.

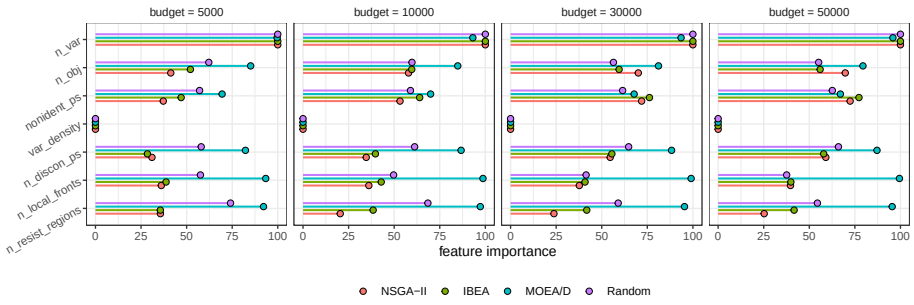

Fig. 6. Relative importance of problem features for regression models.

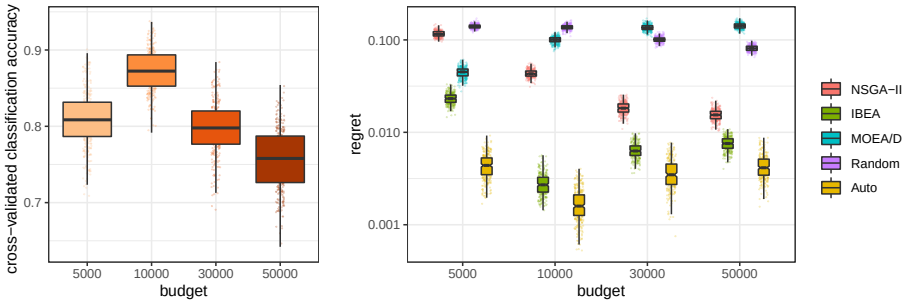

Fig. 7. Accuracy (left) and regret (right) of classification models trained by budget, calculated using 10-fold cross-validation with 30 repetitions.

4.3 Algorithm Selection by Classification

Random Forest Classification. We now focus on feature-based algorithm selection to answer the following question: Given a problem, what is the recommended algorithm for solving it under a particular budget? The interest is no longer in which problem features have more impact on the performance of algorithms, but which features best distinguish algorithms from each other. We still apply random forest, this time for classification. We construct 4 classification models (one per budget) to predict the best algorithm using problem features as predictors. Their classification accuracy, based on 30 repetitions of 10-fold cross-validation, is reported in Figure 7 (left). The lowest accuracy obtained over all folds and repetitions is 0.64, and the median accuracy is always above 0.75 for all budgets. This means that the classifier is able to predict the best algorithm in at least 75% of the cases, which is significantly better than a random classifier (with an accuracy of 25%), or a dummy classifier that would always select IBEA, the most frequent best algorithm for any budget, which outperformed other algorithms in 53%, 80%, 63%, and 58% of problems respectively, for budgets of 5 000, 10 000, 30 000, and 50 000 evaluations, as reported in Section 3.2. The accuracy of random forest is slightly higher for a budget of 10 000. We attribute this to the fact that IBEA outperforms other approaches more often under this budget, which makes the classification problem easier. We also report in Figure 7 (right) the relative hypervolume deviation of the feature-based random forest classifier (`Auto`) from the virtual best algorithm, that is the ideal method, an oracle, that always selects the best algorithm. This measure is termed *regret*, as it indicates how far the obtained hypervolume is from an ideal classifier. It is compared against always selecting each one of the considered algorithms. Notice the log scale in the plot. The regret obtained by the feature-based classifier ranges from 0.0016 to 0.0044 and deviates from an ideal classifier by less than 0.5%. This is less than IBEA, the most frequent best algorithm for all budgets, by an order of magnitude. Compared to an ideal classifier, the relative performance of NSGA-II increases with the budget, while MOEA/D moves away from it.

In Figure 8, we report the relative importance of problem features for the random forest classifiers. As suggested earlier, the importance of the number of variables decreases with the budget. By contrast, the importance of the numbers of objectives and of dominance resistance regions increases with the budget, and even exceeds the number of variables for larger budgets. The numbers of disconnected Pareto sets and local fronts are less important for algorithm selection than for performance regression. In fact, they have a low importance similar to the presence of non-identical Pareto sets and to the varying density for some budgets. Thus, although they have a significant effect on algorithm performance, the impact on all algorithms is the same. The feature analysis for classification shows that it is possible to recommend an algorithm based on problem features with a fairly high accuracy. Moreover, the features have a different impact on the choice of the algorithm depending on the budget.

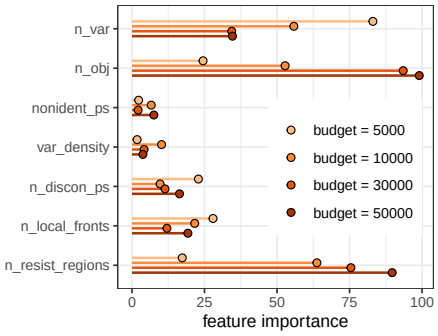

Fig. 8. Relative importance of problem features for classification models.

Decision trees. We conclude with a basic classifier for algorithm selection based on a decision tree. Its construction follows the well-established CART algorithm [2,28], whose segmentation criterion is the Gini diversity index and which generates binary decision trees (i.e. a node has two children at most). In Figure 9, we show decision trees for a budget of 5 000 (left) and 30 000 evaluations (right). Numbers below each node are the number of times NSGA-II, IBEA, MOEA/D and random search are each the best algorithm, respectively, followed by the proportion of problems covered by the node. There are only three values on the first rows for a budget of 5 000 since random search is never the best. Although the accuracy is slightly lower than that of a random forest (0.75 and 0.79, respectively, for a budget of 5 000 and 30 000), we argue that this can provide a useful recommendation system for algorithm selection, with only three levels of decision in these examples.

The tree levels are consistent with the importance of features depicted by random forest, but their joint effect appears more explicitly. Under the smaller budget, the first decision is based on the number of variables. For example, with less than 13 variables, more than 2 local fronts, and less than 10 objectives, IBEA is the best. For a larger budget, and as expected from our previous comments, the feature that appears on top of the tree is the number of objectives. In this case, NSGA-II is the best with few objectives and few variables or few local fronts. Conversely, looking at the rightmost branch of the tree, IBEA is clearly the best with 4+ objectives and 3+ dominance resistance regions. This covers 48% of the problems. Such decision trees justify why an algorithm is recommended based on feature values. In addition, it points out the problem characteristics for which search mechanisms should be refined to improve performance.

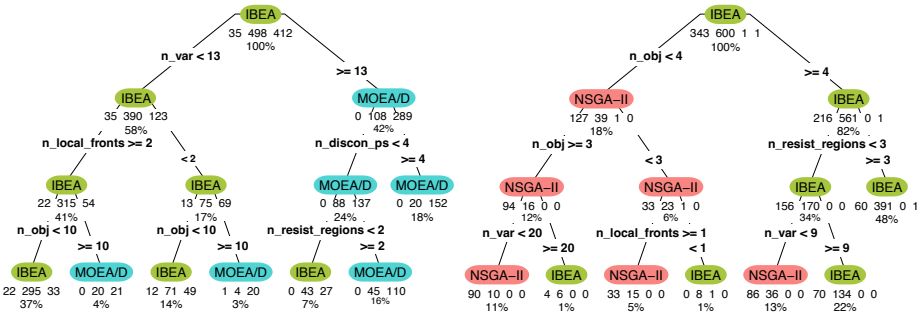

Fig. 9. Algorithm selection for budgets of 5 000 (left) and 30 000 evaluations (right).

5 Conclusions

We adopted a machine learning perspective to carry out the most extensive feature-based benchmark study of distance-based multi/many-objective optimisation problems to date. We generated 28 350 instances based on 945 problem configurations by varying the complexity controlled by 7 features. Random forests and decision trees were then used to understand correlations between the problem features and algorithm performance, predict algorithm performance, and automate the task of algorithm selection for a given problem and budget at hand. We find that, although the considered problem features affect the performance of algorithms in distinctive ways, when used as predictors in a random forest classifier we can predict the best algorithm with an accuracy of 75% or more. Thus, problem features can control the complexity of a problem, and lend themselves to selecting an algorithm when faced with a previously unseen problem. This is the first automated algorithm selection study for continuous problems with more than 2 objectives. We observed that the number of objectives is (i) negatively correlated with algorithm performance, (ii) one of the most important problem features for predicting algorithm performance, especially for larger budgets, and (iii) a key feature to making an accurate algorithm selection when faced with an unseen problem. Future work could investigate if considering additional problem and landscape features can help increase prediction accuracy further. Some expected algorithm behaviours with respect to specific problem features could also be corroborated based on a fine-grained analysis of the data produced in this work. At last, it would be worth studying the sensitivity of algorithm parameters, and their joint impact with problem features on performance.

Acknowledgements

This research is part of the thematic research area DEMO, Decision Analytics utilising Causal Models and Multiobjective Optimisation, jyu.fi/demo, at the University of Jyvaskyla.

References

1. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
2. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. Taylor & Francis, Andover, UK (1984)
3. Brockhoff, D., Tusar, T., Auger, A., Hansen, N.: Using well-understood single-objective functions in multiobjective black-box optimization test suites. arXiv CoRR (2016). https://doi.org/10.48550/arxiv.1604.00359
4. Carnell, R.: lhs: Latin hypercube samples (2022), r package version 1.1.5
5. Deb, K., Prarap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**, 182–197 (2002)
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, pp. 105–145. Springer (2005)
7. Everson, R.M., Fieldsend, J.E., Singh, S.: Full elite sets for multi-objective optimisation. In: Adaptive Computing in Design and Manufacture V, pp. 343–354. Springer (2002)
8. Fieldsend, J.E., Alyahya, K.: Visualising the landscape of multi-objective problems using local optima networks. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1421–1429 (2019)
9. Fieldsend, J.E., Chugh, T., Allmendinger, R., Miettinen, K.: A feature rich distance-based many-objective visualisable test problem generator. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 541–549 (2019)
10. Fieldsend, J.E., Chugh, T., Allmendinger, R., Miettinen, K.: A visualizable test problem generator for many-objective optimization. IEEE Transactions on Evolutionary Computation **26**(1), 1–11 (2022)
11. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5), 477–506 (2006)
12. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: Survey and perspectives. Evolutionary Computation **27**(1), 3–45 (2019)
13. Köppen, M., Vicente-Garcia, R., Nickolay, B.: Fuzzy-Pareto-dominance and its application in evolutionary multi-objective optimization. In: Proceedings of Evolutionary Multi-Criterion Optimization. pp. 399–412. Springer (2005)
14. Köppen, M., Yoshida, K.: Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In: Proceedings of Evolutionary Multi-Criterion Optimization. pp. 727–741. Springer (2007)
15. Kuhn, M.: Building predictive models in R using the caret package. Journal of Statistical Software, Articles **28**(5), 1–26 (2008)
16. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002)
17. Liefooghe, A., Daolio, F., Verel, S., Derbel, B., Aguirre, H., Tanaka, K.: Landscape-aware performance prediction for evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation **24**(6), 1063–1077 (2019)
18. Liefooghe, A., Derbel, B., Verel, S., López-Ibáñez, M., Aguirre, H., Tanaka, K.: On Pareto local optimal solutions networks. In: Proceedings of Parallel Problem Solving from Nature. pp. 232–244. Springer (2018)

19. Liefooghe, A., Verel, S., Lacroix, B., Zăvoianu, A.C., McCall, J.: Landscape features and automated algorithm selection for multi-objective interpolated continuous optimisation problems. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 421–429 (2021)
20. Malan, K.M.: A survey of advances in landscape analysis for optimisation. Algorithms **14**(2), 40 (2021)
21. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 829–836 (2011)
22. Muñoz, M.A., Villanova, L., Baatar, D., Smith-Miles, K.: Instance spaces for machine learning classification. Machine Learning **107**(1), 109–147 (2018)
23. R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2020)
24. Rice, J.R.: The algorithm selection problem. In: Advances in Computers, vol. 15, pp. 65–118. Elsevier (1976)
25. Shand, C., Allmendinger, R., Handl, J., Webb, A., Keane, J.: HAWKS: Evolving challenging benchmark sets for cluster analysis. IEEE Transactions on Evolutionary Computation **26**(6), 1206–1220 (2022)
26. Smith-Miles, K., Lopes, L.: Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research **39**(5), 875–889 (2012)
27. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys **41**(1), 1–25 (2009)
28. Therneau, T., Atkinson, B.: rpart: Recursive partitioning and regression trees (2022), R package version 4.1.16
29. Wickham, H.: ggplot2: Elegant graphics for data analysis. Springer (2016)
30. Zăvoianu, A.C., Lacroix, B., McCall, J.: Comparative run-time performance of evolutionary algorithms on multi-objective interpolated continuous optimisation problems. In: Proceedings of Parallel Problem Solving from Nature. pp. 287–300. Springer (2020)
31. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**, 712–731 (2007)
32. Zitzler, E., Kunzli, S.: Indicator-based selection in multiobjective search. In: Proceedings of Parallel Problem Solving from Nature. pp. 832–842. Springer (2004)