Landscape-Aware Selection of Constraint Handling Techniques in Mulitobjective Optimisation

Jordan N. Cork
Andrejaana Andova
Tea Tušar
Bogdan Filipič
Jožef Stefan Institute and
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
{jordan.cork,andrejaana.andova,tea.tusar,bogdan.filipic}@ijs.si

Pavel Kromer
Technical University of Ostrava
Ostrava, Czechia
pavel.kromer@vsb.cz

Abstract

Constrained multiobjective optimisation problems (CMOPs) are common in real-world optimisation. They often involve expensive solution evaluations and, therefore, it is helpful to know the best methods to solve them prior to actually solving them. These problems also tend to be relatively difficult for algorithms compared to the majority of test problems. This difficulty often presents itself in the infeasible region, calling for a focus on the constraint handling technique (CHT). The purpose of this work is to select the best CHT for problems with difficult constraint functions. This first involves the collection of a set of such problems. CHT selection is then conducted using problem characterisation and machine learning. The outcomes are positive in that prediction achieved a high accuracy. Additionally, further insights are provided into the features that describe CMOPs.

Keywords

constrained multiobjective optimisation, algorithm selection, problem selection, constraint handling techniques

1 Introduction

Real-world optimisation problems very often have multiple objectives and are subject to one or more constraints. This is the domain of constrained multiobjective optimisation (CMO). These problems are generally demanding to solve and have restrictions to the available computational budget. These restrictions make it all the more important to know the best method for solving the problem prior to actually attempting to solve it. This calls for an algorithm selection methodology.

One approach to algorithm selection is to first characterise the problem before conducting the algorithm run [2]. Characterisation involves the calculation of features used to describe the objectives and constraints, as well as their interaction. This is done using a small set of sampled solutions. Once the problem is characterised, knowledge of similar problems can be used to determine the best approach to solving it. This approach is taken in this study and applied to constraint handling techniques (CHTs).

There are three primary contributions from this paper, all within the CMO domain. The first is related to the set of problems used to train the algorithm selection model. Real-world optimisation problems are often difficult to solve, particularly when they include constraints. The field requires a methodology for selecting a subset of problems with difficult constraint functions from the larger set of known problems. This is the first contribution. The CHT selection methodology is then tested on these problems. This methodology is the second contribution. Here, problem characterisation and machine learning are used to predict the best-performing CHT. The final contribution is a set of insights into the features used. The decision tree output by the CHT selection methodology provides significant insights into both which features are useful and what the features reveal about the problems.

The paper is further structured as follows. In Section 2, CMO is described, providing the required background. Section 3 describes the two methodologies, as well as the validation method used. Section 4 describes the experimental setup. In Section 5, the results from the experiments are presented. Finally, in Section 6, the work is summarised and future work is outlined.

2 Constrained Multiobjective Optimisation

Constrained multiobjective optimisation (CMO) involves the optimisation of two or more objective functions given one or more constraint functions. The constraints may be of the equality or inequality forms, however, for this study, only inequality constraints will be considered. Such a CMO problem (CMOP) may be formulated as follows:

$$\begin{aligned} &\text{minimize} \quad f_m(\mathbf{x}), \quad m = 1, \ldots, M, \\ &\text{subject to} \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, J, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$ is a D dimensional *solution vector*, $f_m(\mathbf{x})$ are the *objective functions*, and $g_j(\mathbf{x})$ the inequality *constraint functions*. M is the number of objectives and J the number of inequality constraints.

CMO requires an indicator for assessing solution quality. This indicator is I^{CMOP}. It is used to represent the value of a set of points and is a hypervolume-based indicator, built off of $I^{\text{HV+}}$ from [8]. It was proposed in [19] as a value to be minimised. However, it is commonly maximised based on the `moarchiving` package implementation [9]. On top of I^{CMOP}, the maximised area under the runtime profile curve is used to measure the anytime performance of the algorithm [8]. Here, the runtime profile is the proportion of performance targets attained with respect to the evaluation number.

Many methodologies in CMO use a I^{CMOP}value with normalised function values. For this, the function values of the problems' optimal solution set are required. Together, these are known as the Pareto front. The Pareto front may be obtained empirically,

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

https://doi.org/10.XXXXX/is.2025.scai.YYYYY

through knowledge of the problems construction. Often this is not possible, however, and, therefore, algorithm runs are used to construct an approximation of the front.

In [4], there are 13 benchmark suites listed, consisting of 139 test problems. These test problems are able to be instantiated in various numbers of dimensions and objectives. This then allows for a substantially larger number of test problem instances to be generated based on these 139 base test problems.

Problem characterisation is conducted using exploratory landscape analysis (ELA) features [16]. Work done in [1] has listed 80 such features for CMO. These come from three landscapes: the multiobjective, violation and multiobjective-violation landscapes. There are also two methods of computing them: sampling and random walks.

There are several constraint handling techniques. Four of these were used in our study. The first is the constrained-domination principle (CDP), proposed along with the NSGA-II algorithm [5]. This is a feasibility first approach, where feasible solutions are preferred over infeasible ones. The penalty CHT is a classic method and applies a penalty value to the objective values [20], either statically or dynamically. The Improved-Epsilon (I-Epsilon) CHT was designed to work with the MOEA/D algorithm [7]. It dynamically adjusts the ϵ value based on the number of feasible solutions. Solutions are considered feasible if they are less than the ϵ value. Finally, stochastic ranking (SR) uses a probability value to switch between comparing solutions based on objectives or constraints [18].

3 Methodology

This section presents the methodologies used in the study. First, the methodology for selecting the hard test problems is presented, followed by the methodology for selecting the appropriate CHT and the means for testing the model.

3.1 Difficult Problem Selection

Testing the CHT selection methodology requires test problems. Test problems with too easy constraint functions are less likely to show differences among the CHTs, as algorithms will spend less time dealing with infeasible solutions. More difficult constraint functions, on the other hand, will force the algorithm to deal with infeasible solutions longer and, therefore, give the CHTs time to show their differences. Test problems with difficult constraint functions are then desired for our testing.

As mentioned in Section 2, anytime performance is measured using the area under the runtime profile curve (AUC), with the maximised I^{CMOP} as the indicator. In this study, difficulty is determined based on the anytime performance of a set of algorithms, $\mathcal{A}$. Each of the algorithms is run on the problem R times and the average AUC is taken. This is to ensure robustness. It should be noted that when recording the runs, an archive of all non-dominated solutions is kept and the I^{CMOP} value from this archive is recorded at each solution evaluation. The budget must also be chosen, with budgets allowing algorithm convergence preferred. The maximum average AUC is then used as the problem difficulty, with lower values signifying harder problems. This is formulated as follows:

$$\text{Difficulty}(p) \;=\; 1 - \max_{a \in \mathcal{A}} \left(\frac{1}{R} \sum_{r=1}^{R} \text{AUC}(p, a, r) \right) \qquad (2)$$

This problem difficulty is calculated for each of the problems in the set of problems, $\mathcal{P}$.

Within the current selection, there will still be cases where all CHTs perform roughly the same on the problem. These problems are removed using statistical and practical threshold tests on the final I^{CMOP} values from the 30 runs. Given a normal distribution cannot be ensured in the 30 values from each of the algorithm runs, the Kruskal-Wallis test is used [11]. It determines if independent samples come from the same distribution. However, this still leaves problems with no practical differences in their scores. To filter these out, the mean scores are tested for if they vary more or less than a small delta and those that vary less are removed.

Following the filtering out of problems where no meaningful differences are observed, the $\mathcal{N}$ most difficult problems from the remaining set are selected. This leaves one with a suite of difficult problems upon which at least one of the algorithms from $\mathcal{A}$ performs differently.

3.2 Constraint Handling Technique Selection

The general concept for CHT selection is as follows. First, a machine learning model is trained using the features from each problem in the training set. The labels are the best-performing CHTs on each problem. At inference time, features are calculated on the problem in question (note: this consumes a portion of the available budget). These features are used as input to the machine learning algorithm. The resulting model then predicts the best-performing CHT for use during the run.

Each step will now be described in more detail. The first step is to choose a base algorithm and a set of algorithm-relevant CHTs. The preferred approach would be to select the most appropriate algorithm for the problem to be solved at inference time.

The second step is generating the training data for the machine learning model. First, the features for each of the problems in the training set are gathered. The labels must then be computed, which requires algorithm runs; 30 for each CHT. For this, the budget must be selected carefully. The model, at inference, can be expected to work well only if the budget is the same as it was in training. The average final values from the 30 runs are then taken for each CHT. In CMO, these are the average final I^{CMOP} values, which are being maximised. The CHT with the highest value is then selected as the best-performing CHT. This is used as the label. Once this has been done for each of the problems in the training set, the training data is complete.

The third step is to train the model. A decision tree is preferred for its explainability properties. To enhance the explainability of the model, the depth of the tree should be kept at a minimum. Testing is described in the next subsection. Once complete, i.e. trained with all training data, the model is available for inference.

3.3 Cross Validation Testing

Testing the model involves a leave-one-problem-out cross validation approach. Here, a problem is taken out of the training set and left as the test problem. The model is then trained on the data from the remaining problems in the training set. To predict the best-performing CHT, the features from the test problem are used as input to the model. The model then makes a prediction for the best-performing CHT. This is compared to the actual result.

The methodology makes allowances for when two or more CHTs perform similarly well on the same problem. The prediction made by the algorithm is then correct if it selects any of these. Determining if two or more CHTs are statistically the same is achieved through the use of a statistical test, which in this case was the Mann-Whitney U test [15]. Again, this test was chosen

because a normal distribution cannot be ensured in the resulting final values from the runs. The process is as follows. The CHT with the best mean score is selected, then each of the other CHTs are tested individually against the best-performing CHT to determine if they are equivalent, forming the set of best-performing CHTs. If the predicted label is within this set, it is considered correct. This process is conducted for all problems in the training set and a final percentage of correct predictions is given.

4 Experimental Setup

In this section, the inputs to the methodologies are described, along with the packages used throughout.

There are several inputs to the difficult problem selection methodology. First, there is the set of problems, $\mathcal{P}$. The dimensions chosen were 2, 3, 5, 10 and 30, with only biobjective problems chosen. This resulted in 375 problem instances. The problems were translated from Matlab by hand or taken from `pymoo` [3].

For $\mathcal{A}$, i.e. the set of algorithms, the natural choice was to choose a base algorithm with different constraint handling techniques. The base algorithm chosen was NSGA-II [5]. This was chosen for its versatility with regards to adding various CHTs. Regarding CHTs, CDP, penalty, I-Epsilon and SR were chosen for their compatibility with NSGA-II. CDP was provided as default with NSGA-II by `pymoo`. The others were implemented by hand. The penalty value selected was a static 100, while the settings for all others were the proposed defaults. R was set at 30.

The number of difficult problems selected, $\mathcal{N}$, was set at 20. This number is adequate to test the methodology while still being small enough to manage. The budget selected was the one to be used throughout the study, i.e. $10{,}000 \cdot D$. The delta value for detecting practical differences was set at 0.001.

For the CHT selection methodology, the choice of training problems was the set of difficult problems derived from the setup above. The base algorithm and CHTs were the same as those selected above. The model selected was a decision tree (`scikit-learn` [17]). The tree depth parameter was the only parameter tuned. This tuning was done manually, decreasing from 10 to 3, until the performance began to reduce. Finally, the features used were the 80 features described in [1]. These were calculated with a sample size of $1{,}000 \cdot D$. The random walks were simulated using these same samples.

5 Results

In this section, the results from carrying out the methodologies are described. First, the construction of the set of difficult problems is discussed. Then, the experimental results are presented. Finally, the resulting decision tree is discussed.

The difficulty of each problem was calculated as described in Section 3. The results were heavily skewed towards the easy problem side. With the $\mathcal{N}$ parameter set to 20, that many problems were selected. The difficulties of these ranged from 0.02 to 0.79. The selected problems are listed in Table 1 in order of descending difficulty. They include 5, 10 and 30 dimensional problems, with 2 and 3 dimensional problems clearly being easier to solve. The problems come from the following suites: DC-DTLZ [13], NCTP [12], DOC [14], CTP [6] and C-DTLZ [10].

Table 1 additionally shows the results from the cross validation testing phase of the experiments. As described in Section 3, each problem was given its turn as the test problem, while the others acted as training problems. For 95% of these, the model predicted correctly from the set of actual best-performing CHTs.

Table 1: The results from cross validation testing using the leave-one-problem out methodology. The first column lists the test problems in order of difficulty (descending). D indicates the dimensionality. All problems are biobjective. The models were trained on all problems in the list, bar the test problem in question. 'Actual' lists the best-performing CHT labels, while the prediction column shows the predicted label. If the predicted label is in the actual labels list, the prediction is considered correct. The CHT labels 0, 1, 2 and 3 are CDP, penalty, I-Epsilon and SR, respectively.

Problem	D	Diffic.	Actual	Pred.	Correct
DC2-DTLZ3	30	0.024	[2]	2	Yes
DC2-DTLZ1	30	0.035	[2]	2	Yes
DC2-DTLZ1	10	0.459	[2]	2	Yes
DC2-DTLZ3	10	0.472	[2]	2	Yes
NCTP7	30	0.511	[0, 3]	0	Yes
NCTP8	10	0.645	[0, 1, 3]	3	Yes
NCTP15	10	0.661	[0, 1, 3]	3	Yes
DOC3	10	0.670	[0, 1, 3]	1	Yes
NCTP2	10	0.716	[0, 1]	3	**No**
NCTP1	10	0.721	[0, 1, 3]	3	Yes
NCTP7	10	0.731	[0, 3]	3	Yes
CTP6	30	0.743	[0, 1, 2]	1	Yes
CTP8	30	0.751	[0, 1, 2]	0	Yes
C1-DTLZ3	30	0.760	[0, 1, 2]	2	Yes
DC2-DTLZ1	5	0.770	[2]	2	Yes
CTP8	10	0.773	[0, 1, 2]	0	Yes
DC2-DTLZ3	5	0.781	[2]	2	Yes
DC3-DTLZ1	30	0.786	[2]	2	Yes
NCTP17	10	0.797	[0, 1, 2]	0	Yes
NCTP10	10	0.798	[0, 1, 2]	1	Yes

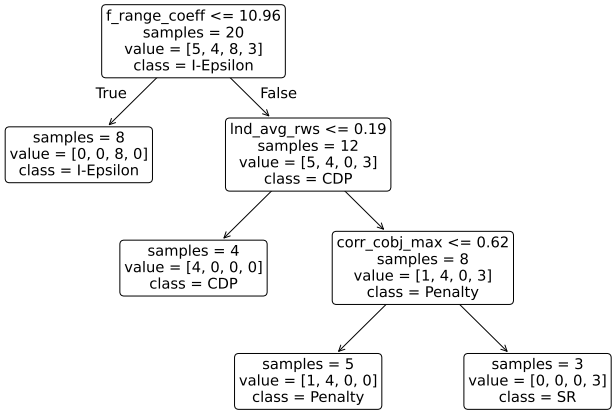

Figure 1: The decision tree built on all the training data. It is used to predict the four CHTs. The indices of the values in the value lists, indicating the number of instances, signify CDP, penalty, I-Epsilon and SR, respectively.

Figure 1 shows the decision tree that resulted from training on all of the available data. As can be seen, the decision tree leaf nodes are nearly pure, meaning it achieved near 100% accuracy on the training data. Due to its high accuracy on the test data and the low tree depth, this is not believed to be overfit.

Only 3 of the 80 supplied features were used by the model, indicating their importance in identifying appropriate CHTs. The

first of these, separating out I-Epsilon, was f_range_coeff (difference between the maximum and minimum of the absolute value of the linear model coefficients). This is a multiobjective landscape feature, focusing on variable scaling. The second feature, separating out CDP, was lnd_avg_rws (average proportion of locally non-dominated solutions in the neighbourhood). This is a multiobjective-violation landscape feature, focusing on evolvability, i.e. the degree to which the problem landscape facilitates evolutionary improvement. The final feature, distinguishing between penalty and SR, was corr_cobj_max (the maximum of the constraints and objectives correlation). This is also a multiobjective-violation landscape feature, focusing on evolvability. It should be noted that the features are not all related to the violation landscape, but also dealt with the objective functions.

6 Conclusion

In this study, the focus was on the needs of real-world CMOPs. These problems are often difficult for algorithms to solve and require expensive solution evaluations. Given the cost of these evaluations, it is helpful to know the best method for solving the problem prior to actually solving it. To address this, the study focused on selecting the most appropriate CHT, a crucial component of any algorithm operating in CMO. For this selection task, it was critical to test on problems with difficult constraint functions. These problems elicit the most variation among CHTs.

The proposition was made for a methodology that selects problems with difficult constraint functions from a larger set, with the end goal of conducting CHT selection. This methodology involved first collecting a large set of CMOPs, then running a set of algorithms on them to determine the difficulty of their constraints. Problems that were easy to solve or showed no variation in algorithm performance were discarded, as they provide no value in future CHT selection tasks. The methodology finally produced a set of $\mathcal{N}$ problems.

This set of difficult problems was used in the second methodology proposed, i.e. selecting CHTs using problem characterisation and machine learning. Four CHTs were chosen and added to the NSGA-II algorithm. These were CDP, penalty, I-Epsilon and SR. The goal of the selection task was to select the best-performing CHT on a given problem, noting that several CHTs can perform best. The methodology was evaluated using cross validation, with the leave-one-problem-out method. The findings from testing were positive and indicate that it is possible to select the most appropriate CHT for a given difficult problem.

Further, the final decision tree trained on all the considered difficult problems provides significant insights into the features used to describe CMOPs. Three features were used in the model and, surprisingly, not all were violation-related features, but also dealt with the objective functions.

In future work, the plans are to extend the CHT selection methodology to the broader domain of algorithm selection. Additionally, while the methodology used in this study conducts the selection prior to the algorithm run, another approach would be to dynamically switch between CHTs during the run. This may improve algorithm performance by making use of CHT strengths at different points during the run.

Acknowledgements

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency (research core funding No. P2-0209 “Artificial Intelligence and Intelligent Systems”, and projects No. N2-0254 “Constrained Multiobjective Optimization Based on Problem Landscape Analysis” and GC-0001 “Artificial Intelligence for Science”).

References

[1] Hanan Alsouly, Michael Kirley, and Mario Andrés Muñoz. 2023. An instance space analysis of constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 27, 5, 1427–1439. DOI: 10.1109/TEVC.2022.3208595.

[2] Andrejaana Andova, Jordan N. Cork, Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2024. Predicting algorithm performance in constrained multiobjective optimization: A tough nut to crack. In *Applications of Evolutionary Computation*. Stephen Smith, João Correia, and Christian Cintrano, editors. Springer Nature Switzerland, Cham, 310–325. DOI: 10.1007/978-3-031-56855-8_19.

[3] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-objective optimization in Python. *IEEE Access*, 8, 89497–89509. DOI: 10.1109/ACCESS.2020.2990567.

[4] Jordan N. Cork and Bogdan Filipič. 2025. A Bayesian optimization approach to algorithm parameter tuning in constrained multiobjective optimization. In *Optimization and Learning*. Bernabé Dorronsoro, Martin Zagar, and El-Ghazali Talbi, editors. Springer Nature Switzerland, Cham, 109–122. DOI: 10.1007/978-3-031-77941-1_9.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 2, 182–197. DOI: 10.1109/4235.996017.

[6] Kalyanmoy Deb, Amrit Pratap, and T. Meyarivan. 2001. Constrained test problems for multi-objective evolutionary optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001*. Springer, 284–298. DOI: 10.1007/3-540-44719-9_20.

[7] Zhun Fan, Wenji Li, Xinye Cai, Han Huang, Yi Fang, Yugen You, Jiajie Mo, Caimin Wei, and Erik Goodman. 2019. An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions. *Soft Computing*, 23, 12491–12510. DOI: 10.1007/s00500-019-03794-x.

[8] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, and Tea Tušar. 2022. Anytime performance assessment in blackbox optimization benchmarking. *IEEE Transactions on Evolutionary Computation*, 26, 6, 1293–1305. DOI: 10.1109/TEVC.2022.3210897.

[9] Nikolaus Hansen, Nace Sever, Mila Nedić, and Tea Tušar. 2024. Moarchiving: Multiobjective nondominated archive classes with up to four objectives. https://github.com/CMA-ES/moarchiving. GitHub repository. (2024).

[10] Himanshu Jain and Kalyanmoy Deb. 2014. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18, 4, 602–622. DOI: 10.1109/TEVC.2013.2281534.

[11] William H. Kruskal and W. Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47, 260, 583–621. DOI: 10.1080/01621459.1952.10483441.

[12] Jia-Peng Li, Yong Wang, Shengxiang Yang, and Zixing Cai. 2016. A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 4175–4182. DOI: 10.1109/CEC.2016.7744320.

[13] Ke Li, Renzhi Chen, Guangtao Fu, and Xin Yao. 2019. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23, 2, 303–315. DOI: 10.1109/TEVC.2018.2855411.

[14] Zhi-Zhong Liu and Yong Wang. 2019. Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces. *IEEE Transactions on Evolutionary Computation*, 23, 5, 870–884. DOI: 10.1109/TEVC.2019.2894743.

[15] Henry B. Mann and Donald R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18, 1, 50–60. DOI: 10.1214/aoms/1177730491.

[16] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 829–836. DOI: 10.1145/2001576.2001690.

[17] Fabian Pedregosa et al. 2011. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12, 85, 2825–2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[18] Thomas P. Runarsson and Xin Yao. 2000. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4, 3, 284–294. DOI: 10.1109/4235.873238.

[19] Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2025. Characterization of constrained continuous multiobjective optimization problems: A performance space perspective. *IEEE Transactions on Evolutionary Computation*, 29, 1, 275–285. DOI: 10.1109/TEVC.2024.3366659.

[20] Yonas Gebre Woldesenbet, Gary G. Yen, and Biruk G. Tessema. 2009. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13, 3, 514–525. DOI: 10.1109/TEVC.2008.2009032.