Incorporating User Preferences in Multi-objective Feature Selection in Software Product Lines using Multi-Criteria Decision Analysis

Takfarinas Saber[1,2], Malika Bendechache[1,2], and Anthony Ventresque[1,3]

[1] Lero – the Irish Software Research Centre
[2] School of Computing, Dublin City University, Ireland
{takfarinas.saber, malika.bendechache}@dcu.ie,
[3] School of Computer Science, University College Dublin, Ireland
anthony.ventresque@ucd.ie

Abstract *Software Product Lines Engineering* has created various tools that assist with the standardisation in the design and implementation of clusters of equivalent software systems with an explicit representation of variability choices in the form of *Feature Models*, making the selection of the most ideal software product a *Feature Selection* problem. With the increase in the number of properties, the problem needs to be defined as a multi-objective optimisation where objectives are considered independently one from another with the goal of finding and providing decision-makers a large and diverse set of non-dominated solutions/products. Following the optimisation, decision-makers define their own (often complex) preferences on how does the ideal software product look like. Then, they select the unique solution that matches their preferences the most and discard the rest of the solutions—sometimes with the help of some Multi-Criteria Decision Analysis technique. In this work, we study the usability and the performance of incorporating preferences of decision-makers by carrying-out Multi-Criteria Decision Analysis directly within the multi-objective optimisation to increase the chances of finding more solutions that match preferences of the decision-makers the most and avoid wasting execution time searching for non-dominated solutions that are poor with respect to decision-makers' preferences.

Keywords: Feature Selection, Software Product Line, Multi-Objective Evolution Algorithm, Multi-Criteria Decision Analysis.

1 Introduction

Software Engineering is divided into multiple domains [1]. One of these domains is Software Product Lines (SPL) which considers groups of related software systems as a whole, rather than dealing with every single one of them separately [2]. Feature Models (FMs) is the most recurrent representation of SPLs. Furthermore,

the FM holds a listing of all the possible feature configurations/combinations which could be viewed as constraints. Therefore, making the FM a representation of all valid software products that could be made out the features in the SPL. Building a software product out of a particular SPL requires the selection of features that respect the desired software configuration. With the multiple characteristics/objectives that are interesting to decision-makers in practice (e.g., cost, technical feasibility, or reliability), the problem of finding the 'best' feature configuration is seen as an instance of a *multi-objective optimisation problem* [3,4].

Evolutionary algorithms have long been used to efficiently optimise problems in various domains from Computer Networks (e.g., [5–7]) to Intelligent Transport Systems (e.g., [8]), to Software Engineering, based on analytical/mathematical (e.g., [5, 6]) or simulated (e.g., [8, 9]) models. Evolutionary algorithms are particularly effective when dealing with multi-objective optimisation problems in software engineering (e.g., [10–13]). This is also the case for multi-objective feature selection in SPL for which the state-of-the-art SATIBEA [3] is an Indicator-Based Evolutionary Algorithm (IBEA) that uses a SAT solver as a mutation operator to correct infeasible solutions.

Multi-objective optimisation techniques result in a set of non-dominated products/solutions from which decision-makers select the product that fits their preferences the most. Given that the number of solutions in the set of non-dominated solutions is often large and that preferences of decision-makers are often complex, decision-makers are usually assisted by Multi-Criteria Decision Analysis (MCDA) tools to accomplish this task [14]. There exist multiple MCDA techniques that take decision-makers' preferences (each of them with its degree of preference expressibility) and return the product that match them the most. We show in this paper that: (i) some MCDA techniques are simplistic and can only handle a limited number of preference types (e.g., only take weights into accounts such as ELECTRE-IV), but they are fast, whereas (ii) other more elaborate MCDA techniques handle larger preference variations (e.g., they enable the use of different utility functions such as PROMETHEE-II), but they are slower and more time-consuming.

In this paper, we aim to include preferences of the decision-makers directly in the multi-objective search process to avoid spending a precious execution time searching for solutions that are (despite being non-dominated) far from decision-makers' preferences. In this paper, we study the effects of using MCDA techniques in the selection process of SATIBEA instead of the Indicator-Based technique (i.e., based on the contribution in Hypervolume of each solution). Particularly, we would like to evaluate the impact in terms of both: (i) the execution time overhead that it would induce, and (ii) quantity of non-dominated solutions matching preferences of decision-makers missed by SATIBEA.

This paper makes the following contributions:

- We propose SAT_MCDA_EA, a hybrid algorithm that includes decision-makers preferences in an MCDA form directly in the evolutionary search process.

– We show that using MCDA techniques as a selection operator has an insignificant impact in terms of execution time overhead in comparison to the execution time taken by one generation of SATIBEA.
– We also show that using MCDA techniques (particularly PROMETHEE-II) enables finding a large number of solutions which better match preferences of decision-makers and that are missed by SATIBEA (despite not outperforming SATIBEA on most of the multi-objective performance metrics).

Combining MCDA techniques with multi-objective evolutionary algorithms has already been attempted in a few recent works (e.g., [15–17]). However, to the best of our knowledge, this is the first time it is attempted in the Software Engineering domain in general and on the multi-objective feature selection in FM in particular.

The remainder of this paper is organised as follows: Section 2 presents the background of our study. Section 3 describes some common MCDA techniques and details our SAT_MCDA_EA approach. Section 4 provides our overall set-up and benchmark for multi-objective feature selection in SPL. Section 5 reports the results of our evaluation in terms of execution time overhead and performance of SAT_MCDA_EA against SATIBEA. Finally, Section 6 concludes the paper.

2 Background

In this section, we detail two aspects that make up the background of our work.

2.1 Software Product Line Engineering

Software Product Line Engineering is the paradigm that attempts to manage software variations more systematically and provide tools that cover the domain engineering and the application engineering processes with their multiple phases/activities [18]. In SPL, all software artefacts (i.e., variations of the same feature) could be picked and put together to form a particular product as long as they are compatible.

Feature Models is a way to represent an SPL. FMs represent the set of all available features with their variations and incompatibilities (i.e., constraints). Figure 1 shows a toy FM example with ten inter-connected features. It shows, for example, that the final product requires a 'Screen'. It also shows that there exist three 'Screen' types (i.e., 'Basic', 'Colour' or 'High Resolution') and only one of them could be selected for the final product. To build a software product from the SPL, we need to select a subset of features $\mathcal{S} \subseteq \mathcal{F}$ such that constraints of the FM $\mathcal{F}$ are satisfied. Constraints of the FM can be modelled as a satisfiability (SAT) problem for instantiating Boolean variables to true or false (in our case, every variable represents a feature) in a way that satisfies all the constraints. A variable $f_i \in \{\text{true, false}\}$ is set to true if the feature $F_i \in \mathcal{F}$ is picked to be part of $\mathcal{S}$, and false otherwise.

An FM can be represented in a conjunctive normal form (CNF). Therefore, searching for a valid software product in the SPL is equivalent to searching

for a feasible solution to the SAT problem. For instance, the FM in Figure 1 describes the screen alternatives in its SAT model with these clauses: $(Basic \vee Colour \vee High\ resolution) \wedge (\neg Basic \vee \neg Colour) \wedge (\neg Basic \vee \neg High\ resolution) \wedge (\neg Colour \vee \neg High\ resolution)$.

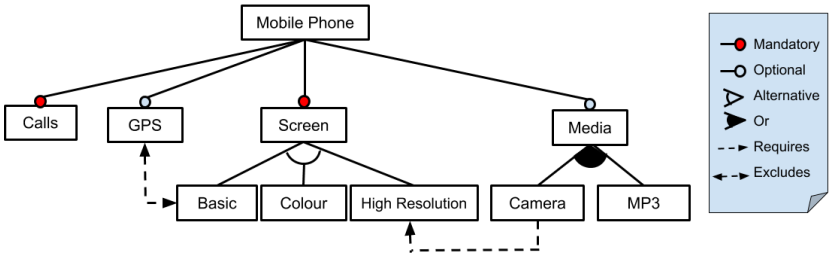

Figure 1: Example of a Feature Model

2.2 Multi-Objective Optimisation

Multi-Objective Optimisation (MOO) considers the optimisation of more than two objective functions at the same time. Software products can be seen from various perspectives (e.g., development cost, reliability, performance). Therefore, by considering each of the perspectives as independent objectives, feature selection in SPL is a suitable candidate for MOO [14].

As a meaningful sample case, we use a set of commonly used optimisation objectives in the literature [19–21]:

- *Correctness* – reduce the number of violated constraints.
- *Richness of features* – increase the number of picked features (have products with more functionality, minimisation of its negative value is considered).
- *Features used before* – reduce the number of picked features that were not used before.
- *Known defects* – reduce the number of known defects in picked features.
- *Cost* – reduce the cost of the picked features.

3 State-of-the-Art and Proposed Approach

In this section, we describe the state-of-the-art algorithm SATIBEA and our proposed approach.

3.1 SATIBEA

SATIBEA [3] is an extension to the Indicator-Based Evolutionary Algorithm (IBEA) which guides the optimisation through a quality indicator selection

process (in this case, the Hypervolume); a SAT solver has been introduced as a mutation operator to assist IBEA.

Note that there are multiple algorithms designed to address the multi-objective feature selection in SPL problem. Most of these algorithms perform in a similar fashion as SATIBEA (evolutionary algorithm + exact algorithm such as SMT [20] or MILP [21, 22]). In this work, we do not compare to them as we do not aim to design an algorithm that is better in terms of multi-objective metrics (even if we report the performance with respect to those metrics below). Instead, our goal is to showcase the fact that including preferences of the decision-makers in the evolutionary search process is worth considering when decision-makers have complex preferences as: (i) it only adds a marginal execution time overhead, and (ii) it finds solutions that are interesting with respect to decision-makers' preferences, but missed by particular IBEA algorithms (in our case SATIBEA).

3.2 Multi-Criteria Decision Analysis

Providing a set of non-dominated solutions, decision-makers explore them to find their preferred one. Given the large size of the non-dominated sets that are obtained after performing the multi-objective optimisation, decision-makers take advantage of MCDA techniques to select the ideal solution with respect to their preferences.

MCDA deals with decision-making constrained by multiple and often conflicting criteria (or objectives or goals). MCDA has been broadly divided into two categories [14]: (i) Outranking Methods: builds a preference relation, and (ii) Multiple Attribute Utility and Value Theory: the 'utility' of every action is scored based on its utility.

In this work, we select three commonly used MCDA techniques: two outranking methods (ELECTRE-IV [23] and PROMETHEE-II [24]) and one Multiple Attribute Utility and Value Theory method (MAUT [25]).

We propose in this paper to substitute the Indicator-Based selection operator in the original SATIBEA algorithm by one of the aforementioned MCDA techniques (i.e., ELECTRE-IV, PROMETHEE-II or MAUT) to create what we call SAT_MCDA_EA. Therefore, we are creating three distinct algorithms under the same umbrella of SAT_MCDA_EA: (i) SAT_ELECTRE-IV_EA, where we use ELECTRE-IV as the selection operator, (ii) SAT_PROMETHEE-II_EA, where we use PROMETHEE-II as the selection operator, and (iii) SAT_MAUT_EA, where we use MAUT as the selection operator.

4 System Set-up

This section presents the different elements that we have used in our experiments: the dataset, the multi-objective performance metrics, the parameters of the genetic algorithms (i.e., SATIBEA and SAT_MCDA_EA), the parameters we use for the MCDA techniques, and the hardware configuration.

4.1 System and Algorithms Set-up

We use the implementation of SATIBEA that is made available to us by its creators (implemented in Java) and implement our approach on top of it. We conduct our experiments on a machine with a 4 core CPU (our algorithms use a core at a time though) and 16 GB of RAM. We ran all our algorithms and determined the average results over 30 runs for each instance.

We use the same parameters for SATIBEA as those defined by its authors (e.g., population size: 300, crossover rate: 0.8, mutation rate of each feature selection: 0.001, and solver mutation rate: 0.02). We also use the same parameters as SATIBEA for our SAT_MCDA_EA approach. Furthermore, we define addition parameters for the MCDA techniques to simulate preferences of decision-makers. Note that the chosen preferences are only selected to showcase different capabilities of each MCDA method. Therefore, it will be worth performing a more robust analysis with different kinds of preferences and a full parameters sweeping for each of these MCDA methods in a future work.

- ELECTRE-IV: requires a parameter triplet (optimisation threshold, preference threshold, and indifference threshold) for every objective. We set these triplets to (5,6,5), (3,4,3), (0.1,0.3,0.1), (1,2,1) and (3,4,3) for Correctness, Richness of features, Feature used before, Known defects, and Cost.
- PROMETHEE-II: requires a parameter pair (weight and preference function) for each objective. We set equal weights for all objectives and set their preference functions to Level, Linear, Linear, Level, and Gaussian for Correctness, Richness of features, Feature used before, Known defects, and Cost.
- MAUT: only requires one parameter per objective (weight) that we set equally for all the objectives.

Based on the parameters that each of the MCDA techniques requires, we see that PROMETHEE-II is the most expressive between them as it enables decision-makers to design their own custom utility function for each objective and feed it to the MCDA.

4.2 Dataset

For our experiments, we use the five of the largest open source FMs we could find [20]. Table 1 shows the version and the size of each of the FMs that we consider in our experiments. The table also reports the number of features and the size of the SAT problem necessary to represent the FM in a conjunctive normal form (in terms of number of variables and number of clauses). Similarly to the SATIBEA paper [3], we set the execution time on the Linux Kernel to 1,200s. For the other datasets, we use smaller execution times based on the convergence time of SATIBEA [19, 26].

4.3 Multi-Objective Performance Metrics

To assess the performance of our algorithms we use 5 multi-objective performance metrics: 4 quality metrics (Hypervolume, Epsilon, Generation Distance, and Inverted Generation Distance) and 1 diversity metric (Spread).

Table 1: Versions and characteristics of the feature models used in our experiments.

Dataset	Version	#Features	#Variables	#Clauses	Time (s)
Linux kernel	2.6.28.6	5,701	6,888	343,944	1,200
eCos	20100825	1,244	1,244	3,146	50
Fiasco	2011081207	300	1,638	5,228	200
FreeBSD	8.0.0	1,396	1,396	62,183	200
μClinux	3.0	616	1,850	2,468	100

- Hypervolume (HV): computes the volume (measured in k dimensions of the problem's search space) that is dominated by the Pareto front (to maximise).
- Epsilon (ϵ): evaluates the smallest distance that is needed for every solution in Pareto front to dominate the Reference front (to minimise).
- Generation Distance (GD): evaluates the smallest distance needed for every solution in Pareto front to dominate the Reference front (to minimise).
- Inverted Generation Distance (IGD): evaluates average distance between every solution in Reference front and its closest solution in Pareto front (to minimise).
- Spread (S): computes the solutions' distribution to evaluate their extent spread in Pareto front (to maximise).

5 Evaluation

5.1 Execution Time Overhead

One of the major issues that kept designers of evolutionary algorithms away from using MCDA techniques within the search process is the excessive execution time that these techniques require. More researchers and practitioners favour less time-consuming indicator-based methods. This is even more true with problems that are only given a few seconds as a total optimisation time budget. In this section, we evaluate the overhead execution time that is introduced by the use of MCDA techniques. We compare the execution time of MCDA techniques to the execution time needed to evolve a full generation and also to the execution time of the default indicator-based method (in our case, the Hypervolume).

Table 2 shows the average execution time in millisecond over 30 iterations of the second generation of SATIBEA (the generation following the evolution of the randomly generated initial population) using the default indicator-based (Hypervolume). The table also shows the average execution time of each particular selection technique from Indicator-Based, to the three considered MCDA techniques (i.e., ELECTRE-IV, MAUT, and PROMETHEE-II).

We clearly see that the execution time of a full SATIBEA generation is very large in comparison to the execution time of the different selection operators (148 times larger on average than the largest selection time per instance). A single generation takes on average 531, 11, 84, 100, and 12 times larger execution times than the most time-consuming selection process (in this case, using

Table 2: Average execution time (ms) of the second generation of SATIBEA, indicator-based selection, and MCDA selection methods.

Dataset	Generation	Indicator-Based	ELECTRE-IV	MAUT	PROMETHEE-II
Linux Kernel	53,788	30.50	1.71	62.75	101.23
eCos	1,235	30.22	1.93	60.33	114.82
Fiasco	12,477	44.49	1.42	59.68	149.04
FreeBSD	12,742	29.57	1.56	71.28	127.30
uClinux	1,197	31.6	1.55	58.09	96.44

PROMETHEE-II) on the instances Linux Kernel, eCos, Fiasco, FreeBSD and uClinux respectively. This is a clear indication that using any of the studied MCDA techniques is less likely to add a significant execution time overhead. The execution time of the section process is particularly insignificant when dealing with the large instances (Linux Kernel, Fiasco and FreeBSD).

We see that with the exception of ELECTRE-IV, MCDA techniques (i.e., MAUT and PROMETHEE-II) necessitate a larger execution time than the default Indicator-Based selection. This is one of the main reasons why the simplistic weighted-sum is the de-facto go to in absence of a pure multi-objective objective optimisation (keeping objectives separate with no aggregation). However, we notice in our usecase that the order by which the execution time of these MCDA techniques exceed the Indicator-Based selection is rather small ($\sim$0.9 and $\sim$2.5 more execution time on average for MAUT and PROMETHEE-II respectively).

Therefore, we could claim that from an execution time perspective and in the context of multi-objective feature selection in large software product lines such as the ones studied in our paper, decision-makers should no longer be reluctant to provide their preferences in advance to be embedded in the multi-objective optimisation process.

5.2 Multi-Objective Performance Metrics

Knowing that using MCDA techniques in the multi-objective optimisation process does not add a significant execution time overhead is good, but obtaining improved results is better –despite not being the most important in our case as our goal is to find more solutions that match decision-makers' preferences. Therefore, we would like to evaluate the impact of our approach in terms of performance and quantify it using the different multi-objective metrics seen in Section 4.

Table 3 shows the average performances achieved by SATIBEA and SAT_-MCDA_EA techniques (i.e., SAT_ELECTRE-IV_EA, SAT_MAUT_EA, SAT_-PROMETHEE-II_EA) with respect to the quality metrics HV, IGD, GD, Epsilon and Spread. We put in bold the best achieved performances per instance and per metric. We also put (*) when results are not statistically significant between SATIBEA and the best performing SAT_MCDA_EA technique (p-value ¡ 0.05 when evaluated using the non-parametric two-tailed Mann-Whitney U test).

Table 3 clearly shows that SATIBEA achieves the best performances on the metrics HV and IGD on all instances. SATIBEA also achieves the best performances on Epsilon in 4 out of 5 instances on average. This is a clear indication that SATIBEA maintains its supremacy with regards to very important multi-objective performance metrics. This is quite understandable as SATIBEA's aim by design is to cover most of the search space, which yields better multi-objective quality metrics performances. However, SAT_MCDA_EA algorithms target solutions that better match the predefined preferences of the decision-makers and leave large parts of the search space unprobed, which yields low multi-objective quality metrics performances.

Table 3 also shows that SATIBEA does not always achieve the best results with respect to the Spread metric. SAT_ELECTRE-IV_EA achieves the best performance on Spread on 3 out of 5 instances on average. Although, Spread is a secondary metrics and should not be interpreted alone without the other quality metrics. Looking at SAT_ELECTRE-IV_EA's performance in terms of HV, we see that it is poor, which reduces the importance of its Spread performance.

Table 3 also shows that SATIBEA is not achieving the best GD on any instance (achieved by SAT_PROMETHEE-II_EA). This is an indication that most of the solutions that are found by SAT_PROMETHEE-II_EA are non-dominated by the solutions found by the other algorithms. However, given that the performance of SAT_PROMETHEE-II_EA in terms of HV is poor, we can deduce that its solutions are not diverse enough. While this might seem negative, we believe that this is a good characteristic. Decision-makers would rather be provided with several non-dominated solutions that are similar and better match their preferences, rather than a set of non-dominated solutions covering a larger space, but match their preferences less. Furthermore, SAT_MAUT_EA also achieves a better performance than SATIBEA in terms of GD on 3 out of 5 instances on average.

5.3 SAT_MCDA_EA's Strictly Non-Dominated Solutions

With SAT_PROMETHEE-II_EA and SAT_MAUT_EA achieving good GD performances, we would like to measure the ratio of non-dominated solutions found by SAT_MCDA_EA algorithms, but missed by SATIBEA. We gather all non-dominated solutions found over all iterations by each algorithm and perform a pairwise non-dominance comparison. Table 4 shows the ratio (in percentage) of solutions found by each SAT_MCDA_EA that are strictly non-dominated (neither equal nor dominated) by any solution found by SATIBEA.

Table 4 confirms our assumption that many solutions found by SAT_MAUT_-EA and SAT_PROMETHEE-II_EA are strictly non-dominated by those found by SATIBEA. We see that SAT_PROMETHEE-II_EA finds the largest number of solutions non-dominated by those found by SATIBEA (∼83% non-dominated solutions on average, and 94% on Fiasco). Therefore, if decision-makers have a prior knowledge of what makes a good software, they are better off using PROMETHEE-II as a selection operator. While this will not yield optimal multi-objective metrics, it will yield more solutions matching their preferences.

Table 3: Comparison of the average performances achieved by SATIBEA and the various SAT_MCDA_EA algorithms.

Dataset	Metric	SATIBEA	SAT_ELECTRE-IV_EA	SAT_MAUT_EA	SAT_PROMETHEE-II_EA
Linux Kernel	HV	**0.136**	0.124	0.123	0.134
	IGD	**0.010**	0.016	0.016	0.012
	GD	0.030	0.130	0.012	**0.007**
	ϵ	**1982**	2047	2051	1991
	S	1.16	**1.24**	1.21	1.19
eCos	HV	**0.252**	0.206	0.188	0.085
	IGD	**0.0071**	0.0072	0.008	0.016
	GD	0.0722	3.8714	0.0935	**0.0031**
	ϵ	**147**	260	217	149
	S	**1.51***	1.30	1.33	1.55
Fiasco	HV	**0.195**	0.133	0.132	0.124
	IGD	**0.009**	0.022	0.024	0.018
	GD	0.065	0.237	0.076	**0.008**
	ϵ	277	917	950	**171**
	S	**1.58**	1.14	1.16	1.27
FreeBSD	HV	**0.24**	0.18	0.18	0.08
	IGD	**0.006**	0.011	0.012	0.018
	GD	0.091	0.156	0.066	**0.004**
	ϵ	**133**	303	308	498
	S	1.21	**1.23***	1.20	1.21
uClinux	HV	**0.893**	0.89	0.891	0.805
	IGD	**0.054**	0.055	0.056	0.060
	GD	0.043	0.016	0.015	**0.012**
	ϵ	**598***	611	604	1199
	S	1.067	**1.229**	1.198	1.003

Table 4: Ratio (in per cent) of strictly non-dominated solutions found over the 30 iterations by SATIBEA using one of the MCDA methods in comparison with the solutions found by SATIBEA when using the default Indicator-Based method.

Dataset	SAT_ELECTRE-IV_EA vs SATIBEA	SAT_MAUT_EA vs SATIBEA	SAT_PROMETHEE-II_EA vs SATIBEA
Linux Kernel	40	41	66
eCos	33	42	90
Fiasco	27	59	94
FreeBSD	26	48	92
uClinux	5	34	73

6 Conclusion and future work

In this paper, we proposed using MCDA techniques directly within the multi-objective search process by employing them as the selection operator. We have evaluated their impact both in terms of induced execution time overhead and in terms of quality of the obtained solutions. We have seen that using the MCDA techniques introduces a non-significant overhead execution time with respect to the execution time of the other operators that make up the evolution. However, we have also seen that using the MCDA techniques within the search process impacts negatively the performance of the algorithm with respect to various multi-objective performance metrics with the exception of GD. We have confirmed that the SAT_MCDA_EA algorithms perform particularly well with respect to GD as they find a large number of solutions that match their preferences but that are not dominated by the solutions found by SATIBEA. The insight obtained from this study encourages us to deepen the investigation of combining MCDA techniques with the multi-objective feature selection in SPL.

Acknowledgement: This work was supported, in part, by Science Foundation Ireland grants No. 13/RC/2094_P2 (Lero) and 13/RC/2106_P2 (ADAPT).

References

1. Ramirez, A., Romero, J.R., Ventura, S.: A survey of many-objective optimisation in search-based software engineering. Journal of Systems and Software **149** (2019) 382–395
2. Metzger, A., Pohl, K.: Software product line engineering and variability management: achievements and challenges. In: FSE, ACM (2014) 70–84
3. Henard, C., Papadakis, M., Harman, M., Le Traon, Y.: Combining multi-objective search and constraint solving for configuring large software product lines. In: ICSE. (2015) 517–528
4. Yadav, H., Chhikara, R., Kumari, A.C.: A novel hybrid approach for feature selection in software product lines. Multimedia Tools and Applications **80**(4) (2021) 4919–4942
5. Lynch, D., Saber, T., Kucera, S., Claussen, H., O'Neill, M.: Evolutionary learning of link allocation algorithms for 5g heterogeneous wireless communications networks. In: GECCO. (2019) 1258–1265
6. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A hierarchical approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks. In: TPNC. (2018) 225–237
7. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O'Neill, M.: A multi-level grammar approach to grammar-guided genetic programming: the case of scheduling in heterogeneous networks. Genetic Programming and Evolvable Machines **20**(2) (2019) 245–283
8. Saber, T., Wang, S.: Evolving better rerouting surrogate travel costs with grammar-guided genetic programming. In: IEEE CEC. (2020) 1–8
9. Bendechache, M., Svorobej, S., Endo, P.T., Mario, M.N., Ares, M.E., Byrne, J., Lynn, T.: Modelling and simulation of elasticsearch using cloudsim. In: DS-RT. (2019) 1–8

10. Saber, T., Gandibleux, X., O'Neill, M., Murphy, L., Ventresque, A.: A comparative study of multi-objective machine reassignment algorithms for data centres. Journal of Heuristics **26**(1) (2020) 119–150
11. Saber, T., Delavernhe, F., Papadakis, M., O'Neill, M., Ventresque, A.: A hybrid algorithm for multi-objective test case selection. In: CEC. (2018) 1–8
12. Saber, T., Ventresque, A., Brandic, I., Thorburn, J., Murphy, L.: Towards a multi-objective vm reassignment for large decentralised data centres. In: UCC, IEEE (2015) 65–74
13. Saber, T., Ventresque, A., Gandibleux, X., Murphy, L.: Genepi: A multi-objective machine reassignment algorithm for data centres. In: International workshop on hybrid metaheuristics, Springer (2014) 115–129
14. Mjeda, A., Wasala, A., Botterweck, G.: Decision spaces in product lines, decision analysis, and design exploration: an interdisciplinary exploratory study. In: VaMoS, ACM (2017) 68–75
15. Mohammed, A., Harris, I., Soroka, A., Nujoom, R.: A hybrid mcdm-fuzzy multi-objective programming approach for a g-resilient supply chain network design. Computers & Industrial Engineering **127** (2019) 297–312
16. Kapsoulis, D., Tsiakas, K., Trompoukis, X., Asouti, V., Giannakoglou, K.: Evolutionary multi-objective optimization assisted by metamodels, kernel pca and multi-criteria decision making techniques with applications in aerodynamics. Applied Soft Computing **64** (2018) 1–13
17. Jafarian-Namin, S., Kaviani, M.A., Ghasemi, E.: An integrated moea and mcdm for multi-objective optimization (case study: control chart design). In: IEOM. (2016)
18. Horcas, J.M., Pinto, M., Fuentes, L.: Software product line engineering: a practical experience. In: SPLC. (2019) 164–176
19. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Is seeding a good strategy in multi-objective feature selection when feature models evolve? Information and Software Technology **95** (2018) 266–280
20. Guo, J., Liang, J.H., Shi, K., Yang, D., Zhang, J., Czarnecki, K., Ganesh, V., Yu, H.: Smtibea: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines. Software & Systems Modeling **18**(2) (2019) 1447–1466
21. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Reparation in evolutionary algorithms for multi-objective feature selection in large software product lines. SN Computer Science **2**(3) (2021) 1–14
22. Saber, T., Brevet, D., Botterweck, G., Ventresque, A.: Milpibea: Algorithm for multi-objective features selection in (evolving) software product lines. In: EvoCOP. (2020) 164–179
23. Govindan, K., Jepsen, M.B.: Electre: A comprehensive literature review on methodologies and applications. European Journal of Operational Research **250**(1) (2016) 1–29
24. Brans, J.P., De Smet, Y.: Promethee methods. In: Multiple criteria decision analysis. Springer (2016) 187–219
25. Allah Bukhsh, Z., Stipanovic, I., Klanker, G., O'Connor, A., Doree, A.G.: Network level bridges maintenance planning using multi-attribute utility theory. Structure and infrastructure engineering **15**(7) (2019) 872–885
26. Brevet, D., Takfarinas, S., Goetz, B., Anthony, V.: Preliminary study of multi-objective features selection for evolving software product lines. SSBSE (2016)