Analyzing Inter-objective Relationships: A Case Study of Software Upgradability

Zhilei Ren[1], He Jiang[1(✉)], Jifeng Xuan[2], Ke Tang[3], and Yan Hu[1]

[1] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian, China
{zren,jianghe,huyan}@dlut.edu.cn
[2] State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China
jxuan@whu.edu.cn
[3] School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
ketang@ustc.edu.cn

Abstract. In the process of solving real-world multi-objective problems, many existing studies only consider aggregate formulations of the problem, leaving the relationships between different objectives less visited. In this study, taking the software upgradability problem as a case study, we intend to gain insights into the inter-objective relationships of multi-objective problems. First, we obtain the Pareto schemes by uniformly sampling a set of solutions within the Pareto front. Second, we analyze the characteristics of the Pareto scheme, which reveal the relationships between different objectives. Third, to estimate the inter-objective relationships for new upgrade requests, we build a predictive model, with a set of problem-specific features. Finally, we propose a reference based indicator, to assess the risk of applying single-objective algorithms to solve the multi-objective software upgradability problem. Extensive experimental results demonstrate that, the predictive models built with problem-specific features are able to predict both algorithm independent inter-objective relationships, as well as the algorithm performance specific indicator properly.

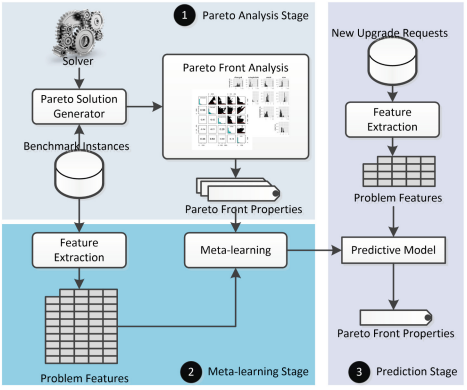
Keywords: Pareto front · Meta-learning · Empirical analysis

1 Introduction

Many real-world multi-objective problems are solved with single-objective approaches [8,10], leaving the inter-objective relationships less studied. For example, the software upgradability problem is among the great challenges in the field of software engineering [10,14]. The problem aims to find the most suitable upgrade scheme that satisfies the users' upgrade requests. An upgrade scheme consists of a sequence of operations, including installing, removing,

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 442–452, 2016.
DOI: 10.1007/978-3-319-45823-6_41

and/or upgrading packages. The software upgradability problem is inherently a multi-objective optimization problem, i.e., users may be interested in different upgrade objectives, such as software stability, package download size, etc. Even only considering single upgrade objective, the problem is reducible to the partial weighted MAXSAT problem [6], which is NP-hard. Moreover, the scalability of the software repositories poses great challenges for the upgrade process. Up to now, there are more than 43,000 packages in the Debian repository[1]. The intrinsic complexity and the scalability make the upgrade process a difficult problem. Meanwhile, in the literatures, most studies encode the upgrade requests into certain single-objective problem instances, such as partial weighted MAXSAT [6], Mixed Integer Linear Programming (MILP) [10], Pseudo Boolean Optimization [11], and Answer Set Programming [4]. Then, solvers are employed to resolve the encoded instances. However, despite the promising accomplishments these studies have achieved, there are still limitations to be improved. For example, in the existing approaches, multiple upgrade objectives are handled in aggregate ways, e.g., the weighted sum scalarization transformation or the lexicographic combination. Hence, a potential risk of such approaches is that, the relationships between different upgrade objectives may not be considered properly. If there exists drastic tradeoff between different objectives, the aggregation strategy has to be carefully chosen, e.g., the weight vector for the weighted sum approaches, or the objective order for the lexicographic approaches.



(a) Illustration of the research framework

```
Package: nano
Version: 2.5.3-2
...
Depends: libc6 (>= 2.14),
         libncursesw5 (>= 6),
         libtinfo5 (>= 6),
         zlib1g (>= 1:1.1.4)
Suggests: spell
Conflicts: pico
```

(b) Package example

Fig. 1. Background information

To face this challenge, we take the software upgradability problem as a case study, and intend to systematically investigate the relationships between different objectives. Motivated by the concept of the Pareto optimality, we are interested in the insights into the characteristics of the upgrade schemes that are not dominated by any other schemes (denoted as Pareto schemes). More specifically,

[1] http://www.debian.org.

Fig. 1(a) illustrates the research framework in this study, which comprises three stages. First, we intend to analyze the characteristics of the Pareto schemes. By uniformly sampling a set of Pareto schemes, we are able to analyze the relationships between different upgrade objectives. Then, for the meta-learning stage, we intend to capture the characteristics of the Pareto front by training a predictive model with features extracted from instances. Finally, we are interested in the possibility of predicting the relationships between objectives with the trained model, and leveraging the predicted indicator to evaluate the suitability of applying certain algorithms. More specifically, we consider the following Research Questions (RQs), which are listed as follows: **RQ1**: How are the different objectives correlated? **RQ2**: Are the inter-objective relationship of the Pareto schemes predictable with problem specific features? **RQ3**: Given an upgrade request, how to assess the suitability of applying single-objective optimization approaches?

2 Problem and Motivation

Let a universe U be a set of software packages, in which each package p is determined by the package name and a version number. Associated with each package, there exists a tuple (D, C), where D denotes the dependency clause set of p, in which each clause indicates a list of software packages. In the clause, at least one of the packages have to be installed so that package p could be installed properly. Accordingly, C represents the conflict clause set for package p. To install package p, none of the packages in the conflict clause corresponding to package p should be installed. Given a universe U, a package installation profile is defined as a subset of the packages within U. In particular, a package installation profile is valid if all the constraints are satisfied. With the package installation profile described, the software upgradability problem could be formulated as follows. Given a universe U, a package installation profile P, as well as a software upgrade request (install, remove, or upgrade a package set), the software upgradability problem aims to determine whether there exists an installation profile P', so that P' is a valid installation profile that satisfies the upgrade request. Moreover, the operation sequence that transfers P to P' is denoted as the upgrade scheme. In Fig. 1(b), we give the package information snippet for `nano`, a text editor. To install `nano` version 2.5.3-2, the constraints have to be met, e.g., packages tagged in the Depends and Conflicts fields have to be installed and removed accordingly.

In this paper, we focus on the optimization version of the problem, i.e., how to determine the most compact valid upgrade scheme for the request. In the literatures, there are 5 minimization upgrade objectives, which aim to minimize the number of packages removed in the solution (f_1: "*removed*"), the packages changed by the solution (f_2: "*changed*"), the number of outdated packages in the solution (f_3: "*notuptodate*"), the number of unsatisfiable package recommendations (f_4: "*unsat*"), and the number of extra packages installed (f_5: "*new*"), respectively. In the existing studies, there are mainly two types of aggregate criteria, both of which consider the lexicographic combination of multiple objectives,

i.e., the objectives are handled in a hierarchical way, and the first objective has the highest priority. More specifically, the paranoid criterion first optimizes the "*removed*" objective, then the "*changed*" objective. Meanwhile, the trendy criterion considers the "*removed*", the "*notuptodate*", the "*unsat*", and the "*new*" objectives successively [11]. These lexicographic approaches do not have to enumerate all the combinations of the objectives. However, if there exists drastic tradeoff between these objectives, the search might be sensitive to the order of the objectives. Under such condition, analyzing the relationships between different upgrade objectives is necessary. Meanwhile, in the evolutionary computation literatures, a common resolution is to provide a set of Pareto optimal solutions. Such approaches do not only provide more choices for the decision maker, but also enable the analysis about the relationships between objectives, which might reveal insights into the problem [12]. Inspired by the concept of Pareto optimality, we are interested in deeper understanding of the software upgradability problem. In this process, the challenge lies in the fact that, obtaining the Pareto schemes for the software upgradability problem is very time consuming. Hence, we would adopt the meta-learning technique to tackle this challenge.

3 Experiments and Discussion

The experiments are conducted on an Intel Core i5 3.2 GHz CPU PC with 4 GB memory, running GNU/Linux with kernel 3.16. For the data set, we employ the benchmark from the Mancoosi International Solver Competition 2010–2012, in which the requests are generated from the Debian repository[2]. After filtering the infeasible upgrade requests[3], we obtain in total 350 upgrade requests. Then, we proceed to describe the Pareto scheme sampling procedure. The Pareto schemes could be defined as the upgrade schemes that are not dominated by any other upgrade schemes. In the literatures, there exist various mechanisms that could convert the problem of achieving the Pareto front into a number of scalar optimization problems, such as the weighted sum, the Tchebycheff aggregation, and the boundary intersection approaches [15]. Due to its simplicity and effectiveness, we adopt the weighted sum approach. More specifically, for each upgrade request, inspired by [15], the $\{5, 5\}$-simplex lattice design is employed to generate 126 weight vectors, which are then used to construct the weighted single-objective optimization problem. Then, for each weighted problem, we use a publicly available solver `mccs`[4] with Gurobi, which is a state-of-the-art MILP solver, to compute the optimal upgrade scheme in the corresponding direction.

3.1 RQ1: Conflict Analysis

First, we are interested in the comparisons between the two existing lexicographical criteria. With the trendy and the paranoid criterion, we apply `mccs`

[2] http://mancoosi.org/misc/.
[3] Note that these requests could be detected by the feature extraction phase, see RQ2.
[4] http://www.i3s.unice.fr/~cpjm/misc/mccs.html.

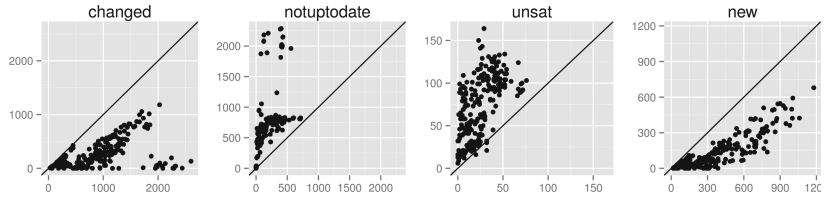
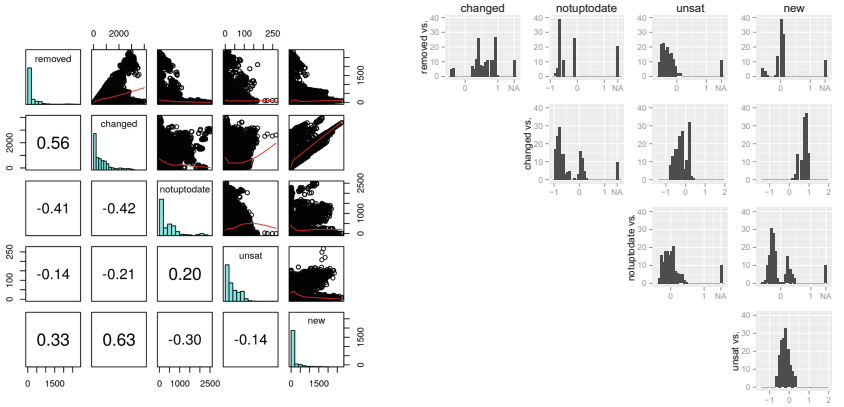


Fig. 2. Comparison between the trendy (x-axis) and the paranoid (y-axis) criteria



(a) Scatter plot for the Pareto schemes (b) Correlation distributions

Fig. 3. Properties of the Pareto schemes

over each upgrade request respectively, and plot the obtained single-objective optimal upgrade schemes in Fig. 2. Since the two criteria share the first objective "*removed*", we only plot the comparisons between the two upgrade criteria under the rest 4 objectives. In each subfigure, the x-axis and the y-axis represent the trendy and the paranoid criteria, respectively. From Fig. 2, the following observations could be drawn. (1) For the "*changed*" objective, `mccs` with the paranoid criterion outperforms `mccs` with the trendy criterion. This observation is as expected, in that the trendy criterion does not consider optimizing the "*changed*" objective. (2) Similarly, when we consider the "*notuptodate*" and the "*unsat*" objectives, which the paranoid criterion does not care, `mccs` with the trendy criterion could achieve better performance. (3) Surprisingly, for the "*new*" objective, which only the trendy criterion considers, `mccs` with the trendy criterion is outperformed by the same solver with the paranoid criterion. A possible reason is that, there exist certain correlation between objectives.

To examine the assumption, we plot the Pareto schemes in Fig. 3(a). Due to the dimensional issue of the problem, we adopt the pairwise scatter plot to illustrate the relationship between the 5 objectives. The figure comprises 3 components. First, the upper panel represents the scatter plot of the Pareto schemes projected on each specific plane. Second, the diagonal panel illustrates

Table 1. Feature category and examples

Domain	Feature category	Feature example
SAT (54 features)	Problem size features	Variable and clause numbers
	Variable–clause graph features	Variable and clause node degree statistics
	Variable graph features	Node degree statistics
	Clause graph features	Clause graph node degree statistics
	Balance features	Horn clauses fraction
MILP (141 features)	Problem size features	Number of variables and constraints
	Variable-constraint graph features	Variable and constraint node degree statistics
	Linear constraint matrix features	Cariable and constraint coefficient statistics
	Objective function features	Normalized objective coefficient statistics
	LP-based features	Mean value of integer slack vector
	Right-hand side features	Mean value of the right-hand side
	Probing based features	Mixed integer programming gap

the histograms that capture the distributions of the objective values for each objective. Finally, in the lower panel, we present the Spearman correlation coefficients between objectives. From Fig. 3(a), several interesting phenomena could be observed. First, under different objectives, the distributions of the objective values vary significantly. For example, for the "*unsat*" objective, the objective values of the Pareto schemes range within $[100, 250]$. Meanwhile for the "*new*" objective, the corresponding interval is $[500, 2500]$. Second, from the upper panel of Fig. 3(a), we observe that the relationships between objectives vary greatly as well. For example, when we consider the relationship between the "*changed*" and the "*new*" objectives, the points in the corresponding subfigure (row 2, column 5) exhibits a near linear pattern. This phenomenon conforms with what we observe in Fig. 2. Furthermore, the hypothesis that the two objectives are correlated is supported by the Spearman test, with a coefficient 0.63. When we consider the coefficients between other objectives, conflicts could be detected. For example, there exists a clear negative correlation between the "*removed*" and the "*notuptodate*" objectives.

More importantly, the correlation coefficients between objectives may also vary significantly over different upgrade requests. In Fig. 3(b), we plot the pairwise histograms for the 5 objectives. In Fig. 3(b), each subfigure corresponds to

the distribution of correlations between objectives over the 350 upgrade requests, e.g., the upper left subfigure describes the distribution of the correlation between the "*removed*" and the "*changed*" objectives. From the figure, we can observe several phenomena that support our hypothesis. As expected, the correlation coefficients between objectives vary greatly. For example, the majority of correlation coefficients ranges within $[-1, 0]$, when we consider the relationship between the "*removed*" and the "*unsat*" objectives. This implies that the two objectives are negatively correlated. Meanwhile, the "*changed*" and the "*new*" objectives are positively correlated. Besides, in the figures, the "NA" indicates the upgrade request and the corresponding objectives, for which the objective value is constant.

Summary of RQ1: We detect conflict between different upgrade objectives. Moreover, the correlation coefficients between objectives vary greatly over different upgrade requests. Hence, more analysis is required, to reveal more insights.

3.2 RQ2: Correlation Prediction

In the previous experiment, we detect variation of the correlation coefficients when considering the relationships between objectives. In practice, the correlation coefficients between objectives could be helpful, such as the objective reduction in the field of many-objective optimization [13]. However, calculation of these coefficients requires sampling Pareto schemes, which further relies on exactly solving NP-hard problems. Consequently, this procedure is very time consuming. In our experiment setup, sampling Pareto schemes for the upgrade requests costs 225,070.4 s, which may not be tolerable in practice. As a solution, we adopt the meta-learning approach, to investigate the underlying linkages between problem specific features and the properties of Pareto scheme. In the literatures, meta-learning has been widely used for algorithm selection [7], and performance prediction [5]. These approaches share a commonality, i.e., a set of features are extracted from the instances, to characterize their properties.

To extract problem-specific features from benchmark instances, we first encode the upgrade requests with different formulations, and construct features with off-the-shelf feature extractors[5]. First, due to the close relationship between the software upgradability problem and SAT, we could transfer the upgradability requests into their decision version SAT instances, and obtain the problem-specific features accordingly. Second, the software upgradability problem could also be described as MILPs. Consequently, given an upgrade request, we generate a MILP instances considering the equally weighted sum of the 5 objectives. Then, feature extraction is conducted over the instance. In particular, we merge the features from the two problem domains together, which results in 195 features[6]. The feature categories and the examples for each category are listed in Table 1. For both problem formulations, the detail of the problem features could be found in [5]. Besides, a byproduct of the feature extraction is that, we could

[5] The code is obtained from http://www.cs.ubc.ca/labs/beta/Projects/EPMs/.
[6] We make the data publicly available at http://oscar-lab.org/upgradability/.

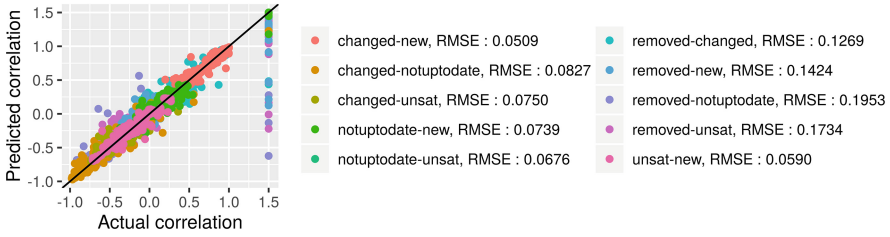


Fig. 4. Correlation prediction results (Color figure online)

detect infeasible upgrade request at an early stage. For example, if the transfered SAT instance is claimed to be unsatisfiable, satisfying the corresponding upgrade request will not be possible. For the sake of simplicity, with respect to the correlation associated with each pair of objectives, we build a regression model, namely random forest [1], due to its effectiveness. Given all the 350 upgrade request, we adopt the 10-fold cross-validation to evaluate the performance of the predictive model. In particular, those "NA" values are assigned with an exception value (1.5 in this study).

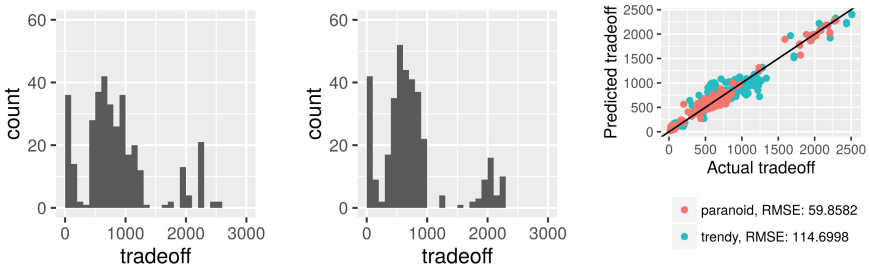
In Fig. 4, we present the prediction results of the regression. In the figure, the x-axis and the y-axis indicates the actual and the predicted correlation over the test set, within each fold of validation, respectively. Different objective combinations are denoted with different colors. Moreover, in the figure we present the Root Mean Square Error (RMSE) in the figure, to measure the accuracy of the prediction. From the figure, we observe that the trained model is able to estimate the correlation properly. The majority of the points lie closely around the reference line $y = x$. For the accuracy measure, all the RMSE values achieved by random forest lie below 0.2.

Summary of RQ2: With the problem specific features extracted from the upgrade requests, we could detect potential correlations between objectives.

3.3 RQ3: Tradeoff Assessment

As in RQ1, we observe that the performances of `mccs` with the trendy and the paranoid criteria may vary greatly over different upgrade requests. More importantly, using single-objective approaches such as lexicographic programming may pose risks within the problem solving process. If the search is overly concerned with certain objective, chances are that there may be other objectives over which the single-objective approaches may perform poorly [2]. Consequently, applying these methods may cause risks during the problem solving process.

In this experiment, we propose a measurements inspired by the reference based solution evaluation routine in the evolutionary computation literatures [9]. More specifically, the idea originates from the concepts of the ideal reference points, which are constructed by the best objective values with respect to each objective, considering all the Pareto schemes. With the ideal points described, we



(a) trendy (b) paranoid (c) prediction results

Fig. 5. Distribution of *tradeoff* considering different criteria, and prediction results

define the risk of applying single-objective approaches for software upgradability problem as the maximum loss considering all the optimizing criteria:

$$tradeoff(s) = \max_{1 \leq i \leq 5} \{f_i(s) - f_i(ideal)\}, \tag{1}$$

where f_i indicates the objective function of each optimizing criteria, and s indicates the upgrade scheme achieved by certain algorithm. Given an upgrade scheme s achieved by certain lexicographic programming algorithm, if s is close to the ideal point, it would be suitable to accept the upgrade scheme to realize the upgrade process. Contrarily, a large *tradeoff* value implies that the corresponding scheme's quality is poor with respect to at least one objective. Accordingly, applying the corresponding upgrade scheme may be risky.

In Fig. 5, we present the distribution of the *tradeoff* measurements considering the two different criteria. From the figure, the following observations could be drawn. First, for both the criteria, the *tradeoff* value varies diversely over different upgrade requests. For example, for the paranoid criterion, the *tradeoff* value ranges within $[0, 2286]$, which means that there exists both easy upgrade requests ($tradeoff = 0$), and requests which leads to large *tradeoff*. Second, Similar as in RQ2, calculating the *tradeoff* indicator requires exactly solving the software upgradability problem. To make the measurement practical for guiding the problem solving process, we resort to the meta-learning mechanism again, to estimate the tradeoff with the problem specific features. The experimental setup is the same as in RQ2, except that we change the response variable from the correlation to *tradeoff* defined in Eq. 1. The prediction results are illustrated in Fig. 5(c), which is organized similarly with Fig. 4. From the figure, we can observe that the random forest model is able to predict the risk measurement *tradeoff* accurately. For both the trendy and the paranoid criteria, the RMSEs achieved by the random forest model are 59.8582 and 114.6998, respectively.

Summary of RQ3: In this experiment, we propose a reference based indicator *tradeoff*, to assess the suitability of applying aggregation based single-objective algorithms to solve the problem. Furthermore, we demonstrate that the measure is predictable, using the problem-specific features, which to some extent prevent the time consuming Pareto scheme sampling.

4 Conclusions and Future Work

In this study, we systematically investigate the relationships between multiple objectives of the software upgradability problem. The contributions of the paper could be summarized as follows. First, we design a series of experiments to analyze the inter-objective relationships for the software upgradability problem. Second, we apply the meta-learning technique to investigate the characteristics of the upgrade requests. Furthermore, the trained model enables the prediction of the properties of new upgrade requests. Finally, we propose a risk indicator, to measure the suitability of applying single-objective algorithms to solve the multi-objective software upgradability problem. However, there are still limitations that deserve future work. For example, the Pareto schemes are achieved by an exact solver. Due to the intrinsic complexity, sampling Pareto schemes for large scale upgrade request can be very time consuming. In the future, we intend to resort to multi-objective evolutionary algorithms [3,15] to approximate the Pareto schemes. Besides, in this study, we treat the off-the-shelf feature extractor as a black box, to capture the characteristics of the upgrade requests. Hence, deeper insights could be gained, if we further study the properties of the features mined from the upgrade requests.

Acknowledgement. This work is supported in part by the National Natural Science Foundation of China under Grants 61370144 and 61403057, in part by National Program on Key Basic Research Project under Grant 2013CB035906, and in part by the Fundamental Research Funds for the Central Universities under Grants DUT15TD37 and DUT16RC(4)62.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Corne, D.W., Knowles, J.D.: Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 773–780. ACM (2007)
3. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evol. Comp. **18**(4), 577–601 (2014)
4. Gebser, M., Kaminski, R., Schaub, T.: Aspcud: a linux package configuration tool based on answer set programming. In: Proceedings of Workshop on Logics for Component Configuration 2010 (2010)
5. Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artif. Intell. **206**, 79–111 (2014)
6. Ignatiev, A., Janota, M., Marques-Silva, J.: Towards efficient optimization in package management systems. In: Proceedings of the 36th International Conference on Software Engineering, pp. 745–755 (2014)
7. Lindauer, M., Hoos, H.H., Hutter, F., Schaub, T.: Autofolio: an automatically configured algorithm selector. J. Artif. Intell. Res. **53**, 745–778 (2015)
8. Liu, Z., Yan, Y., Qu, X., Zhang, Y.: Bus stop-skipping scheme with random travel time. Transp. Res. Part C Emerg. Technol. **35**, 46–56 (2013)

9. Lokman, B., Köksalan, M.: Finding highly preferred points for multi-objective integer programs. IIE Trans. **46**(11), 1181–1195 (2014)
10. Michel, C., Rueher, M.: Handling software upgradeability problems with MILP solvers. In: Proceedings of Workshop on Logics for Component Configuration 2010, vol. 29, pp. 1–10 (2010)
11. Trezentos, P., Lynce, I., Oliveira, A.L.: Apt-pbo: solving the software dependency problem using pseudo-boolean optimization. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 427–436 (2010)
12. Verel, S., Liefooghe, A., Jourdan, L., Dhaenens, C.: Analyzing the effect of objective correlation on the efficient set of MNK-landscapes. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 116–130. Springer, Heidelberg (2011)
13. Wang, H., Yao, X.: Objective reduction based on nonlinear correlation information entropy. Soft Comput. **20**(6), 2393–2407 (2016)
14. Xuan, J., Martinez, M., DeMarco, F., Clement, M., Marcote, S.L., Durieux, T., Berre, D.L., Monperrus, M.: Nopol: automatic repair of conditional statement bugs in java programs. IEEE Trans. Software Eng. (2016, online)
15. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comp. **11**(6), 712–731 (2007)