Information and Software Technology 65 (2015) 1–13

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

An Integer Linear Programming approach to the single and bi-objective Next Release Problem

Nadarajen Veerapen [a,*], Gabriela Ochoa [a], Mark Harman [b], Edmund K. Burke [a]

[a] Computing Science and Mathematics, University of Stirling, FK9 4LA Scotland, UK
[b] Department of Computer Science, University College London, WC1E 6BT, UK

ARTICLE INFO

Article history:
Received 8 December 2014
Received in revised form 8 March 2015
Accepted 23 March 2015
Available online 31 March 2015

Keywords:
Integer Linear Programming
Multi-objective optimization
Next Release Problem
Requirements optimization
Search based software engineering

ABSTRACT

Context: The Next Release Problem involves determining the set of requirements to implement in the next release of a software project. When the problem was first formulated in 2001, Integer Linear Programming, an exact method, was found to be impractical because of large execution times. Since then, the problem has mainly been addressed by employing metaheuristic techniques.
Objective: In this paper, we investigate if the single-objective and bi-objective Next Release Problem can be solved exactly and how to better approximate the results when exact resolution is costly.
Methods: We revisit Integer Linear Programming for the single-objective version of the problem. In addition, we integrate it within the Epsilon-constraint method to address the bi-objective problem. We also investigate how the Pareto front of the bi-objective problem can be approximated through an anytime deterministic Integer Linear Programming-based algorithm when results are required within strict runtime constraints. Comparisons are carried out against NSGA-II. Experiments are performed on a combination of synthetic and real-world datasets.
Findings: We show that a modern Integer Linear Programming solver is now a viable method for this problem. Large single objective instances and small bi-objective instances can be solved exactly very quickly. On large bi-objective instances, execution times can be significant when calculating the complete Pareto front. However, good approximations can be found effectively.
Conclusion: This study suggests that (1) approximation algorithms can be discarded in favor of the exact method for the single-objective instances and small bi-objective instances, (2) the Integer Linear Programming-based approximate algorithm outperforms the NSGA-II genetic approach on large bi-objective instances, and (3) the run times for both methods are low enough to be used in real-world situations.

1. Introduction

A key aspect in any large project is determining an appropriate set of requirements as expressed by stakeholders. The Next Release Problem (NRP), originally proposed by Bagnall et al. [1], formalizes this issue and involves finding a subset of requirements or a subset of stakeholders that maximizes a desirable property, such as revenue, while being constrained by an upper bound on some other property, usually cost. It is important to note that the NRP is only concerned with the next release. Planning over several releases is formalized in the closely related, and more complex, Release Planning Problem [2].

In the bi-objective NRP [3], the upper bound is lifted and the constraint transformed into a second objective which is usually in conflict with the first. As a result, the decision-maker is presented with a set of potential solutions, the Pareto-optimal set, that represents the different possible trade-offs between the two objectives.

Research on the NRP has mainly been concerned with search based software engineering approaches [4] probably because initial efforts to use Integer Linear Programming (ILP), which is an exact approach, proved inconclusive beyond small problem instances because of large run times [1]. In this paper we revisit ILP in the light of much improved solvers to see if, and how, it can be used to address the NRP with one and two objectives.

The main contributions of this paper are as follows:

* Corresponding author. Tel.: +44 1786 46 7462.
E-mail address: nve@cs.stir.ac.uk (N. Veerapen).

http://dx.doi.org/10.1016/j.infsof.2015.03.008
0950-5849/

1. We show that ILP can now solve single-objective NRP instances under 4 s when several hours were required in 2001.
2. We show that iterated applications of ILP through the ε-constraint method can be used to generate the Pareto front of the bi-objective NRP, with the most complex instance clocking in at over 8 h.
3. We show how to deterministically generate a good approximation of the Pareto front when time is limited by applying an Anytime Dichotomic Scheme which performs iterated applications of ILP.

We also introduce two transformations on the NRP that make it more amenable to be solved by a genetic algorithm that cannot handle constraints.

The next section gives an overview of related work. Section 3 presents the NRP with one objective, the possible constraints between requirements and how to simplify the problem under certain conditions. The datasets and problem instances are presented in Section 4 and the results for the NRP with one objective are given in Section 5. Section 6 deals with the bi-objective NRP and presents the ε-constraint method, the Anytime Dichotomic Scheme and the NSGA-II algorithm. Results for two objectives are presented in Section 7. Section 8 examines the threats to the validity of this study. This is followed by the conclusion in Section 9.

The supplemental material contains model transformation examples, an exploratory analysis of the NSGA-II seeding used in the paper as well as the source code and results. It is available at the following address: www.cs.stir.ac.uk/nve/nrp/

2. Related work

Since the early 2000s a number of researchers have modeled software requirements selection and planning as specialized optimization problems. A variety of formulations considering single and multiple software releases, and single and multiple (different) optimization objectives, have been proposed. The term *Next Release Problem* (NRP) was coined by Bagnall et al. [1] who first formalized and successfully solved requirements optimization for a single release as a constrained optimization problem using metaheuristics.

Feather and Menzies [5] proposed an iterative approach to requirements optimization that involves human-expert decision making. A requirement interaction model is executed to randomly sample the space of options. The large data set produced is then condensed into a small list of critical decisions, which is presented to humans experts. At each iteration, experts select from a decreasing number of major alternatives to finally produce a near-optimal set of requirements.

In a series of publications Ruhe et al. [6–9] propose a comprehensive family of hybrid approaches, termed EVOLVE, to solve the so-called Release Planning Problem. *Release planning*, a generalization of the Next Release Problem, considers the selection and assignment of features to a sequence of consecutive product releases subject to resource and risk constraints.

Additional approximate algorithms employed to successfully solve the single objective NRP include ant colony optimization [10] and backbone-based multilevel search [11]. We compared our results to the latter in Section 5. Ant colony optimization is further explored with successful results against competing algorithms in the Release Planning Problem with dependencies [12].

A bi-objective formulation of the NRP, with requirement costs and stakeholder satisfaction values considered as two separate criteria, was presented by Zhang et al. [3] where evolutionary methods are used to find approximations of the Pareto optimal set (Section 6.5). In a follow up study [13], the authors incorporated requirement interactions.

Other authors have studied the bi-objective NRP with the aim of comparing the performance and highlighting the strengths of different algorithms including evolutionary approaches, simulated annealing and ant colony optimization [14,15].

Finkelstein et al. [16] considered the problem of fairness analysis in requirements optimization. A bi-objective formulation solved with evolutionary methods is used for balancing multiple stakeholders with different views on the priorities for requirements choice.

Apart from the present paper, renewed interest in exact methods for the NRP has materialized in the work by Harman et al. [17] which employs a dynamic programming algorithm to perform sensitivity analysis of unconstrained NRP instances.

3. The NRP with one objective

The single-objective NRP is reducible to a knapsack problem [1] and is, as such, an $\mathcal{NP}$-hard combinatorial optimization problem [18]. This means that there exists no polynomial time algorithm to solve this problem unless $\mathcal{P} = \mathcal{NP}$. Consequently several papers have investigated the use of heuristic methods to address the single-objective NRP [4]. Of course, the probable non-existence of a polynomial time algorithm does not imply that exact methods cannot perform well in practice. Incidentally, this is the case with the knapsack problem: as early as 1979, random knapsack instances with 10,000 variables were solvable within 3 s on a UNIVAC 1108 [19].

It was later proved [20] through average-case analysis that the Nemhauser–Ullman algorithm [21], a dynamic programming knapsack algorithm, has expected polynomial running time.

However, with the NRP, one needs to take into account the constraints on the requirements as well as the fact that NRP instances might exhibit some properties that make them more difficult to solve than random knapsack instances.

In the paper that introduced the NRP in 2001 [1], ILP was tested using CPLEX, a commercial solver: the more complex problem instances required several hours to solve exactly or were interrupted after running for 20 h. In a more recent paper [11] on approximately solving large scale constrained NRP instances, it was shown that solving real-world instances of around 4000 requirements could take about 27 min on a PC with an Intel Core 2.53 GHz CPU. A very recent paper [17] demonstrated that the above-mentioned Nemhauser–Ullman algorithm could solve unconstrained random NRP instances composed of 1500 requirements, with an Intel Core i7 2.76 GHz CPU, in 25 s when the data was uncorrelated and in twice the time with highly correlated data.

Since 2001, only a few papers have investigated ILP for the NRP [22,23] or for release planning [24,25], and then only for relatively small problem instances. During that time ILP solvers have improved drastically [26]. As an example, over the 1991–2009 period, CPLEX has seen an algorithmic speedup of over 55,000 [27] when discounting hardware speedup. This enables us to state the research question for the single-objective section of this paper.

RQ1 How efficient is CPLEX on large scale instances of the single objective NRP?

The NRP is constrained by a total maximum cost and potentially other constraints such as dependencies between requirements. Fig. 1 gives an illustrated example of the structure of the potential relationships between requirements and stakeholders.

One key criterion for representing the NRP is whether a stakeholder's requirements need to be completely satisfied [1] or if partial satisfaction is acceptable [16]. The complete satisfaction

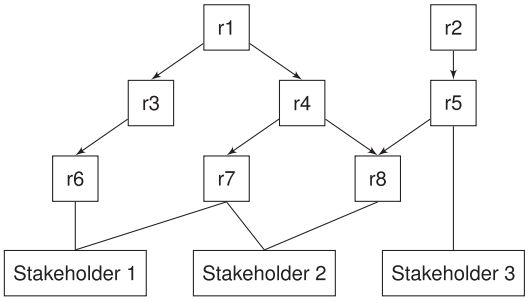


Fig. 1. Illustrated example of an NRP instance with prerequisites between requirements. Three stakeholders have some requirements. These depend on other requirements in order to be satisfied.

criterion requires the NRP formulation to contain specific variables for the stakeholders. This is not the case for partial satisfaction. This leads to three model representations that are relatively similar but which differ in the variables that are represented in the model:

1. The *general formulation* [1] requires decision variables for both the requirements and stakeholders.
2. The *basic-stakeholder formulation* is inspired by Bagnall et al. [1] and only represents stakeholder variables. It is a simplification of the general formulation and therefore applies to complete stakeholder satisfaction. As far as we are aware, this formulation has not been used before: Bagnall et al. [1] give an *independent basic* formulation which only uses stakeholder variables but which does not model the constraints of the general formulation.
3. The *basic-requirement formulation* [16] only represents the requirement decision variables. Here, a stakeholder's wishes can be partially satisfied. We selected this name for this formulation as we did not find a name for it in the literature.

3.1. General formulation

In this formulation, we assume that the decision maker wishes to maximize stakeholder profits and bound requirements costs. Let $\mathbf{X} = [x_1, x_2, \ldots, x_n]$ and $\mathbf{Y} = [y_1, y_2, \ldots, y_m]$ be the vectors of binary decision variables representing the inclusion or not of requirements 1 to n and stakeholders 1 to m respectively. Let $\mathbf{C} = [c_1, c_2, \ldots, c_n]$ be the cost vector associated to requirements, $\mathbf{W} = [w_1, w_2, \ldots, w_m]$ the profit vector associated to the stakeholders and b some bound on the total cost. The constraints are represented by binary relationships. Let P be the set of couples (i,j) where requirement i is a prerequisite for requirement j. Let Q be the set of couples (i,k) where requirement i is requested by stakeholder k. Then the problem can be modelled by the following ILP:

$$\max \quad f(y) = \sum_{i=1}^{m} w_i y_i \tag{1a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} c_i x_i \leqslant b \tag{1b}$$

$$x_i \geqslant x_j, \ \forall (i,j) \in P \tag{1c}$$

$$x_i \geqslant y_j, \ \forall (i,j) \in Q \tag{1d}$$

$$x \in \{0,1\}^n, \ y \in \{0,1\}^m \tag{1e}$$

In expression (1c), x_i needs to be equal to 1 if $x_j = 1$ but x_j can still be equal to 0 if $x_i = 1$, thus modelling the prerequisite accurately. In expression (1d), the same technique is used to force the inclusion of all the requirements requested by some stakeholder if this stakeholder's profit is used in objective function f.

3.2. Basic-stakeholder formulation

This simpler formulation is based on the transformations 2 and 3 described in Section 3.5, in order to convert the general formulation into one that only considers the stakeholders while still preserving the initial semantics:

$$\max \quad f(y) = \sum_{i=1}^{m} w_i y_i \tag{2a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} c_i \bigvee_{j \in S_i} y_j \leqslant b \tag{2b}$$

$$y \in \{0,1\}^m \tag{2c}$$

One drawback of this formulation is that it is no longer a linear formulation and using it with an ILP solver would require a translation step to convert the logical operators into linear inequalities. However, it is simpler to solve with a metaheuristic approach than the general formulation since fewer constraints need to be considered. Its other benefit is using m decision variables instead of $m + n$ in the general formulation.

3.3. Basic-requirement formulation

In this last formulation, the profits or levels of stakeholder satisfaction are not computed based on strict adherence to each stakeholder's complete set of requested requirements but instead they allow for partial satisfaction of each stakeholder's set of requirements. This effectively removes the need for stakeholder decision variables (y_i) in the model and the weight vector $\mathbf{W}$ now represents the profits associated to each requirement. If no other additional constraints are considered, we have:

$$\max \quad f(x) = \sum_{i=1}^{n} w_i x_i \tag{3a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} c_i x_i \leqslant b \tag{3b}$$

$$x \in \{0,1\}^n \tag{3c}$$

For Zhang et al. [3], stakeholder satisfaction is used instead of profit. A vector $\mathbf{W}$ represents weights assigned to each stakeholder. The weight is proportional to the stakeholder's importance and is associated to an $(n \times m)$ matrix $\mathbf{V}$ representing the value that each stakeholder assigns to each requirement. This value is potentially null. This gives us a slightly different version of Expression (3a):

$$\max \quad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_j v_{ij} x_i \tag{3a'}$$

In Section 4, we describe a number of instances that are tackled using the different formulations. We employ the general formulation with ILP and the basic-stakeholder formulation with genetic algorithms when addressing the classic and realistic instances. For the real-world instances, we use the basic-requirement formulation for all solving methods. This choice was made since this is how these instances have been approached in the literature.

3.4. Constraints

The NRP can be augmented by different constraints that describe the relationships between the various requirements. We briefly describe some of the interactions presented by Zhang et al. [13] that are used in this paper and that can be formulated as a binary relation on any pair of requirements (i,j). The corresponding linear constraint is given between parentheses.

- AND – If any one of the two requirements is selected then the other must be selected as well ($x_i = x_j$).
- OR – At most one of the two requirements can be selected ($x_i + x_j \leqslant 1$).
- PRECEDES – If j is selected then i must be selected as well ($x_i \geqslant x_j$). This is identical to the prerequisite constraint in the general formulation.

3.5. NRP transformations

An NRP formulation can be transformed in several ways whilst still maintaining the same properties. Here we present three transformations that can be used to simplify the models. To the best of our knowledge, only the second transformation has been used before in the context of the NRP. Examples of the transformations are provided in the supplemental material. In this paper, the main purpose of these transformations is to eliminate or reduce the number of explicit constraints for the benefit of NSGA-II since it cannot handle constraints natively.

3.5.1. Transformation 1

This removes the $x_i = x_j$ constraints. Since equality is, of course, an equivalence relation, each decision variable involved in an $x_i = x_j$ constraint is an element of the same equivalence class $[x_i] = \{x_j | x_i =_{\text{AND}} x_j, x_j \in X\}$. Those decision variables can therefore be substituted for a single variable representing the class, thus decreasing the total number of variables and eliminating the AND constraints.

Classic *union-find* algorithms over disjoint-set data structures [28, Chapter 21] can be employed to determine the equivalence classes of variables involved in multiple disjoint AND constraints.

3.5.2. Transformation 2

This removes the $x_i \geqslant x_j$ constraints [1]. Let R be the set of requirements and let S be the set of stakeholders. Let PRECEDES′ be the transitive closure of PRECEDES on R. Let R_i be the set of requirements requested by stakeholder $i \in S$. Then $\widehat{R}_i = R_i \cup \{r' | r' \text{ PRECEDES}' r, r \in R_i\}$ is the set of all the requirements required to fulfill the request of stakeholder i.

If PRECEDES′ is not readily available, it can be computed using Warshall's algorithm [29, Chapter 2] as suggested by Bagnall et al. [1]. Alternatively, it is fair to assume that no cycles are induced by the PRECEDES relation. In this context, the transitive closure can be constructed by parsing the requirements in reverse topological order [29, Chapter 4].

Given this information, we can generate $x_i \geqslant y_j$, $i \in R_j$, $\forall j \in S$ and remove any $x_i \geqslant x_j$ constraints. Let us note that the transformation will potentially generate a larger number of constraints than in the initial formulation.

3.5.3. Transformation 3

This removes the x_i variables and only leaves the y_j variables. Let S_i be the set of stakeholders who requested requirement i, then any variable $x_i = \vee_{j \in S_i} y_j$. The $x_i \geqslant y_j$ of the general formulation are no longer required.

4. Problem instances

To perform some computational experiments, a number of problem instances are required, ideally from real-world data. Unfortunately obtaining such data is problematic, especially for large instances, as private companies usually keep this confidential.

In order to test the single objective NRP, we will thus use the instances presented by Xuan et al. [11]. These are divided into two groups called the *classic instances* and the *realistic instances*. The number of requirements and stakeholders for these instances are given in Table 1.

We will also have two real-world datasets. However, because of their relatively small size, they will only be used for the bi-objective NRP.

4.1. Classic instances

The *classic instances* are composed of five synthetic datasets generated by Xuan et al. [11] in the way described by Bagnall et al. [1]. It is not possible to use the original datasets [1] since they are no longer available.

Each dataset, *nrp1* to *nrp5*, corresponds to a single problem instance for the bi-objective NRP. For the single-objective case, each dataset is used to generate two instances which differ only by the bound on the cost constraint. This bound is the sum of total costs multiplied by a coefficient (0.3, 0.5 or 0.7).

The data itself is generated according to a multi-level approach. Each level consists of a number of requirements specified beforehand. The cost of each requirement and the number of child requirements is chosen uniformly in a range that depends on the level.

4.2. Realistic instances

In order to obtain more realistic large instances, Xuan et al. [11] proposed to use the bug repositories for the Eclipse, Mozilla and Gnome open-source projects. Bugs correspond to the requirements and stakeholder requests correspond to users commenting on the bugs. The cost of a requirement is the severity of the bug and the profit per stakeholder is selected uniformly in a predefined range.

Four subsets of bugs are extracted from the three repositories (*nrp-e1* to *nrp-e4*, *nrp-m1* to *nrp-m4*, *nrp-g1* to *nrp-g4*). These subsets correspond to bi-objective NRP instances. For the one-objective case, two bounds on the costs are defined. Again, these bounds correspond to the sum of total costs multiplied by a coefficient (0.3 or 0.5).

The requirements for realistic instances do not have prerequisites.

4.3. Real-world instances

We use two real-world datasets from Motorola and University College London (UCL). They are both relatively small datasets.

4.3.1. Motorola

The Motorola dataset [3] contains 35 requirements for handheld communication devices and 4 mobile telephone companies as stakeholders. The stakeholders have a different set of features they wish to be part of each device. The sets of features are specific to each company so no two companies want the same feature. Each company is equally important to Motorola and therefore no weights are used to assign priorities to stakeholders.

4.3.2. UCL – RALIC

Replacement Access, Library and ID Card – RALIC – was a 2.5 year project deployed in 2009 in University College London (UCL) to replace the previous access control systems and provide integration with the library access and borrowing [30]. The dataset [31] is publicly available online.[1] The instance as used in this paper is composed of 77 stakeholders and 142 requirements with AND and OR constraints. Simplifying the AND constraints with the transformation described in Section 3.5 reduces the number of requirements to 45.

[1] http://www.cs.ucl.ac.uk/staff/S.Lim/phd/dataset.html.

Table 1
Results for Classic and Realistic NRP instances. Column 1 indicates the name of the dataset set and the number of requirements, column 2 the number of stakeholders and column 3 the coefficient on the total cost bound. The optimal value found by ILP is given in column 4 followed by the mean run time. Column 6 provides the best results obtained by BMA as an indication of the performance of heuristic methods.

Instance			ILP		BMA
Name req.	Sta.	Coef.	Opt.	Time (s)	Best
nrp1		0.3	1204	0.15	1201
140	100	0.5	1840	0.05	1824
		0.7	2507	0.03	2507
nrp2		0.3	4970	3.63	4726
620	500	0.5	8065	2.95	7566
		0.7	11,316	0.56	10,987
nrp3		0.3	7488	0.47	7123
1500	500	0.5	11,159	0.32	10,897
		0.7	14,196	0.31	14,180
nrp4		0.3	10,690	1.81	9818
3250	750	0.5	15,985	2.50	15,025
		0.7	20,913	1.79	20,853
nrp5		0.3	18,510	0.32	17,200
1500	1000	0.5	24,701	0.31	24,240
		0.7	28,912	0.32	28,909
nrp-e1		0.3	7919	0.06	7572
3502	536	0.5	11,071	0.06	10,664
nrp-e2		0.3	7446	0.10	7169
4254	491	0.5	10,381	0.17	10,098
nrp-e3		0.3	6666	0.05	6461
2844	456	0.5	9362	0.06	9175
nrp-e4		0.3	5814	0.05	5692
3186	399	0.5	8174	0.10	8043
nrp-g1		0.3	6130	0.03	5938
2690	445	0.5	8897	0.03	8714
nrp-g2		0.3	4580	0.06	4526
2650	315	0.5	6553	0.03	6502
nrp-g3		0.3	5932	0.02	5802
2512	423	0.5	8501	0.04	8402
nrp-g4		0.3	4218	0.03	4190
2246	294	0.5	6063	0.03	6030
nrp-m1		0.3	10,770	0.14	10,008
4060	768	0.5	15,540	0.12	14,588
nrp-m2		0.3	8707	0.13	8272
4368	617	0.5	12,585	0.25	11,975
nrp-m3		0.3	10,391	0.14	9559
3566	765	0.5	15,096	0.10	14,138
nrp-m4		0.3	7777	0.06	7408
3643	568	0.5	11,369	0.07	10,893

5. Experiments on the 1-objective NRP

This section addresses the research question for the single-objective NRP.

RQ1 How efficient is CPLEX on large scale instances of the single objective NRP?

The large scale NRP has previously been solved by Xuan et al. [11] using a backbone-based multilevel algorithm which scales the problem down using multilevel reductions and uses multilevel refinements to construct a set of near-optimal stakeholders. The backbone identifies characteristics that are common in the optimal solutions. As those are unknown, an approximate backbone is generated from the intersection of a number of locally optimal solutions. Another type of backbone, the soft backbone, is also used.

While this method is relatively complex to implement, as far as we are aware, it is also the best attempt at solving large scale Next Release Problems. It was compared to a genetic algorithm and simulated annealing. For these reasons, we do not reimplement the backbone-based multilevel algorithm but quote the profits obtained. We do not provide direct run time comparisons as our test machine is different from theirs.

Using ILP is comparatively simpler as it only requires entering the model into a black-box solver. Here we use CPLEX an earlier version of which was also used by Bagnall et al. [1]. The solving process involves applying a number of techniques, all transparent to the user. In its most basic form, an ILP solver will first relax the integer linear problem to a continuous linear problem which can be solved efficiently by the Simplex algorithm. This provides a starting point for a Branch and Bound algorithm to explore the decision tree of integer solutions [32]. Modern solvers include many additional techniques such as model simplification, various tree pruning algorithms and tree exploration heuristics.

5.1. Experimental setup

We use CPLEX 12.5.1 to solve the *classic* and *realistic* instances using the general NRP formulation presented in Section 3.1. The CPLEX API is called from a C++ program whose only other task is to read in and output the data. CPLEX is only allowed to use one thread.

We set the `EpInt`, `EpGap` and `EpAGap` parameters to 0. Briefly, these specify the threshold beyond which a solution is considered to be close enough to optimality. Setting them to 0 ensures that we obtain the best possible value although in many cases this unnecessarily lengthens the search as the best value could be obtained with a small non-zero parameter value. Default values are used for the remaining CPLEX parameters.

The C++ source code is compiled with GCC 4.6.3 with the `-O3` compiler optimization flag. The experiments are carried out on a PC running Ubuntu 12.04 LTS 64-bit with an Intel Core i7-3770 CPU clocked at 3.4 GHz and with 16 GB of RAM.

Since the results are deterministic, we run each instance three times to account for potential run time differences due to external processes on the test machine having a detrimental effect on very small execution times. In practice, the run times for the same instance only differ by, at most, 0.01 s.

5.2. Analysis of results

The results for the *classic* and *realistic* instances are given in Table 1. As we can see, the optimal solution for most instances is usually found in less than one second. The hardest instances are the ones based on the classic synthetic data (especially *nrp2* and *nrp4*), which are still solved in less than 4 s. This answers RQ1. These low execution times would be suitable for most single objective requirements optimization scenarios.

The only *classic* instances which have approximately the same number of decision variables as the realistic instances are the ones based on *nrp4*. However, the realistic instances have smaller run times. The difference here is that the *classic* instances are the only ones that have requirement prerequisites. Resolving those constraints probably explains the higher run times.

While the *classic* instances are not the same ones that took several hours to solve using CPLEX in 2001 [1], they were nonetheless generated using the same algorithm [11] and are likely to exhibit the same level of difficulty. Solving them with an improved version of CPLEX and a modern desktop PC is now only a matter of seconds. Moreover, there is also a significant improvement over the backbone-based multilevel algorithm, which took between 52 s to 27 min to approximately solve the *classic* and *realistic* instances.

The marked improvement in execution time allows for highly flexible requirements management scenarios, for instance where fast recomputation of the solution is required following reevaluation of stakeholder wishes or requirement costs. Another possible

use-case occurs when the available data is imprecise: multiple scenarios can now be quickly evaluated across multiple different configurations in order to have a wider view of the possible outcomes.

Given that the run times are small, it is reasonable to think that it may be possible to use multiple applications of ILP to solve exactly the bi-objective NRP provided the number of solutions remains within some reasonable limit. If run time is constrained, an approximate method should be considered.

We will now consider the following research questions:

RQ2.1 How efficient is it to compute the exact Pareto front of the NRP with two objectives?

RQ2.2 How efficient is it to obtain an approximation of the Pareto front that can be considered good enough?

6. The bi-objective NRP

In the bi-objective NRP, the cost constraint is transformed into an objective of its own. There are now two objectives to solve. The first objective needs to be maximized whilst the second needs to be minimized.

It is often convenient to have all objectives require either maximization or minimization. In order to maximize the second objective, it is multiplied by -1. If we consider the general formulation of the bi-objective NRP, we have:

$$\max \quad f_1(y) = \sum_{i=1}^{m} w_i y_i \tag{4a}$$

$$f_2(x) = -\sum_{i=1}^{n} c_i x_i \tag{4b}$$

$$\text{subject to} \quad x_j \geqslant x_j, \ \forall (i,j) \in E \tag{4c}$$

$$x_i \geqslant y_j, \ \forall (i,j) \in C \tag{4d}$$

$$x \in \{0,1\}^n \tag{4e}$$

$$y \in \{0,1\}^m \tag{4f}$$

Naturally the different formulations can be supplemented by different constraints as described earlier. Let us introduce some terms and definitions in the next section.

6.1. Multi-objective combinatorial optimization

In multi-objective combinatorial optimization [33], two or more objectives are considered. Since the objectives are likely to be in conflict with each other, it is assumed that a solution optimizing all objectives does not exist. The aim is to find a set of solutions called the *efficient solutions*. This term as well as some other useful terms are defined below and illustrated in Fig. 2.

For simplicity and relevance to this work, let us consider only 2 maximization objectives. Let X be the set of all feasible solutions of the problem in the decision space, $X = \{x \in \{0,1\}^n | Ax \leqslant b\}$, where $Ax \leqslant b$ are the constraints of the problem. Let $Z = \{f(x) | x \in X\}$ be the feasible set in the objective space.

Definition 1. Given two vectors u and v, u *dominates* v if $(u_1 \geqslant v_1 \wedge u_2 > v_2) \vee (u_1 > v_1 \wedge u_2 \geqslant v_2)$. This is often referred to as *Pareto dominance* and is denoted by $u \succ v$.

Definition 2. A solution $\hat{x} \in X$ is said to be *efficient* if there is no $x \in X$ such that $f(x) \succ f(\hat{x})$.

Definition 3. The image $f(\hat{x})$ of an efficient solution $\hat{x}$ is said to be a *non-dominated point*. The set of all non-dominated points is called the *Pareto front*.

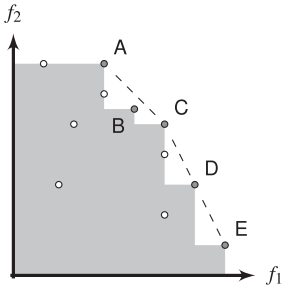


Fig. 2. Different types of points in bi-objective optimization. Points A–E are non-dominated points. They form the Pareto front. The remaining points, in the shaded region, are dominated. A, C and E are points of extreme supported efficient solutions. D is a point of a non-extreme supported efficient solution. B is a point of a non-supported efficient solution. The area of the shaded region is the hypervolume.

Efficient solutions can be divided into two categories: supported efficient solutions and non-supported efficient solutions.

- A *supported efficient solution* is the optimal solution of a weighted sum single-objective problem with weight vector $[\lambda, 1-\lambda]$, $0 < \lambda < 1$. It is said to be *extreme* if its image is a vertex (in general position, i.e, no three collinear vertices) of the convex hull of the Pareto front and *non-extreme* if it is found in its relative interior.
- A *non-supported efficient solution* is an efficient solution for which there does not exist an optimal solution to a weighted sum single-objective problem for any weight vector $\lambda > 0$.

Definition 4. The *lexicographic order*, $u >_{lex} v$, is defined by $u_1 > v_1 \vee (u_1 = v_1 \wedge u_2 > v_2)$. In this paper *lexmax* will be the operator that returns the lexicographically maximal solution.

6.2. Which methods to generate Pareto fronts?

To address RQ2.1, we require an exact bi-objective algorithm that can use the single objective solver results to generate the exact Pareto front. Here we choose to employ the ε-constraint method which has the advantage of being very simple both to understand and implement.

To address RQ2.2, we will consider NSGA-II since it was previously used to solve the bi-objective NRP [3,15,13]. With NSGA-II there is no guarantee that any of the solutions obtained is part of the real Pareto front. We also wish to test if the exact single objective approach can be used within an approximate algorithm and thus generate an approximation that is a subset of the real Pareto front. The simplest way to leverage the efficiency of a single objective solver is to solve multiple weighted sums of the two objectives [33]. Even if the single objective solver is an exact one, this only provides an approximation of the Pareto front since non-supported efficient solutions cannot be reached this way. The Anytime Dichotomic Search [34] is an example of an algorithm based on weighted sums and we will be using it in our experiments. Its main advantage over other weighted sum algorithms is that it provides a good distribution of points on the Pareto front irrespective of the execution time it is allotted.

The next three subsections provide more details on these multi-objective algorithms.

6.3. The Epsilon-constraint method

The concept of ε-constraint involves solving one of the objectives of a multi-objective combinatorial optimization problem

and transforming the remaining objectives into constraints bounding the search space [35]:

$$\max_{x \in X} \{f_i(x) | f_j(x) \geqslant \varepsilon_j, i \neq j\}$$

In the bi-objective case we thus have:

$$\max_{x \in X} \{f_1(x) | f_2(x) \geqslant \varepsilon_2\} \quad (5)$$

A systematic variation of ε can be used to generate a Pareto front [36,37]. This is illustrated in Fig. 3, which gives the pseudo-code of the algorithm. Fig. 4 provides a visual example.

The general idea is to start by solving one of the two objectives, say f_1. This gives us solution x^1, and $f_1(x^1)$ is the best possible value for f_1. The search space is then bounded by the constraint $f_2 \geqslant f_2(x^1) + \delta$ and we solve for f_1 within that space. The process is repeated as long as a new solution for f_1 is found. During this procedure, f_2 is gradually improving whilst f_1 is worsening.

Notice that δ is a key parameter. If we want an approximation of the Pareto front, then δ can be used to generate solutions that will have at least δ difference between the values of f_2. Here we want the exact Pareto front, δ should thus be set to the smallest representable value, i.e, 1 for integer functions and the smallest machine precision for floating point functions. One might wonder why a strict inequality is not used instead of the δ parameter. Strict inequalities are not allowed in linear programming because they might lead to solutions that are not well-defined, e.g., in $\max_{x \in \mathbb{R}}\{x | x < 10\}$.

6.4. Anytime Dichotomic Scheme

The Anytime Dichotomic Search (ADS) presented here is based on Aneja and Nair's dichotomic scheme [38] where an approximation of the Pareto front is recursively computed using weighted sums of the objectives. One problem of this approach, which is also shared by the ε-constraint method, is that if there is a time limit then the generated points are grouped in a subset, or a number of unevenly distributed subsets, of the Pareto front. Dubois-Lacoste et al. [34] describe an anytime behavior of the dichotomic scheme to obtain a more even distribution of points, which we will use here. Note however that Dubois-Lacoste et al. [34] use a local search algorithm as solver while we use ILP. Fig. 5 presents the pseudo-code of the Anytime Dichotomic Scheme and Fig. 6 gives a visual example.

The main idea of this approach involves taking two solutions, at first x^1 and x^2 such that they are lexicographic optima of (f_1, f_2) and (f_2, f_1) respectively, and finding a solution in between. This is done by calculating the weight λ that defines a line perpendicular to the segment joining the two solutions. Solving the weighted sum $\lambda f_1 + (1 - \lambda) f_2$ gives a new solution x'. If the latter is any of the two previous solutions then no new solution can be found between those solutions with the dichotomic scheme. Otherwise additional solutions can potentially be found between the two newly created segments. This process is repeated until no new solutions are found or the time limit is reached.

The defining characteristic of the anytime approach is that the longest segment is always chosen, thereby targeting the exploration of the front to sections with fewer solutions.

Recall from Section 6.1 that weighted sums produce supported efficient solutions. Therefore, this approach will not produce any non-supported efficient solutions and may miss non-extreme supported efficient solutions.

6.5. NSGA-II

NSGA-II, Non-dominated Sorting Genetic Algorithm, is an elitist version of the original NSGA [39]. It is one of the most popular multi-objective metaheuristic algorithms with over 7500 citations to the original paper in the Scopus database at the time of writing.

Determine x^1 an optimal solution for f_1
$A \leftarrow \{x^1\}$
$\epsilon_2 \leftarrow f_2(x^1) + \delta$
while $\max_{x \in X}\{f_1(x) | f_2(x) \geq \varepsilon_2\}$ is feasible **do**
  $\hat{x} \leftarrow \max_{x \in X}\{f_1(x) | f_2(x) \geq \varepsilon_2\}$
  $A \leftarrow A \cup \hat{x}$
  $\epsilon_2 \leftarrow f_2(\hat{x}) + \delta$
end while
Filter dominated solutions in A

Fig. 3. The Epsilon-constraint algorithm for two objectives.

Fig. 7 presents a top-level view of NSGA-II. The starting population of size N is initialized with random solutions. After the solutions have been evaluated, a fitness corresponding to the dominance depth is assigned to each of them. The dominance depth of a given solution is the number of solutions within the population which dominate this solution. Solutions with the same depth form a front and the lower the depth the better. A measure of spread, the crowding distance, is computed for each solution. A new population is generated through recombination, mutation and tournament selection operators. For each new population, the dominance depth and crowding distance are computed. Then the solutions are initially sorted according to dominance depth and then according to crowding distance within the same front. The first N solutions are kept and the process is repeated until a stopping criterion is met, usually a fixed number of generations.

NSGA-II was successfully used to solve the bi-objective NRP [3,15,13]. This approach cannot readily deal with constraints. Where necessary, we employ the strategy already used by Zhang et al. [13]: invalid solutions in the population are repaired at the end of the `Create-Random-Population` and `Make-New-Population` steps. Since we use the transformations described in Section 3.5, in practice only the OR constraints need repair for the instances in this paper. When x_i OR x_j is violated, x_j is set to 0.

Through preliminary tests we experienced the same phenomenon described by Zhang et al. [3], namely that NSGA-II had some difficulty reaching both ends of the Pareto front, with solutions being generally clustered in the middle.

It has been shown that the spread of solutions obtained with genetic algorithms can be improved when starting with a population that includes good non-random solutions in both the single objective [40] and the multi-objective case [41]. This has been dubbed *inoculation* or *seeding*. More recently, seeding has also been used in other software engineering problems [42,43]. The exploratory analysis we performed, included in the supplemental material, showed that seeding was also beneficial for the NRP and we are thus using it in this paper.

6.6. Performance metrics

With the single-objective problem, we could simply rely on the objective function's value to measure a solution's quality. With two objectives, we need to select appropriate metrics to gauge the quality of approximate fronts.

There are a number of multi-objective performance metrics [44] and in this paper we will use two of them: the number of exact solutions found on the approximate front and the hypervolume.[2]

[2] As implemented in the R `mco` package, http://cran.r-project.org/web/packages/mco/.

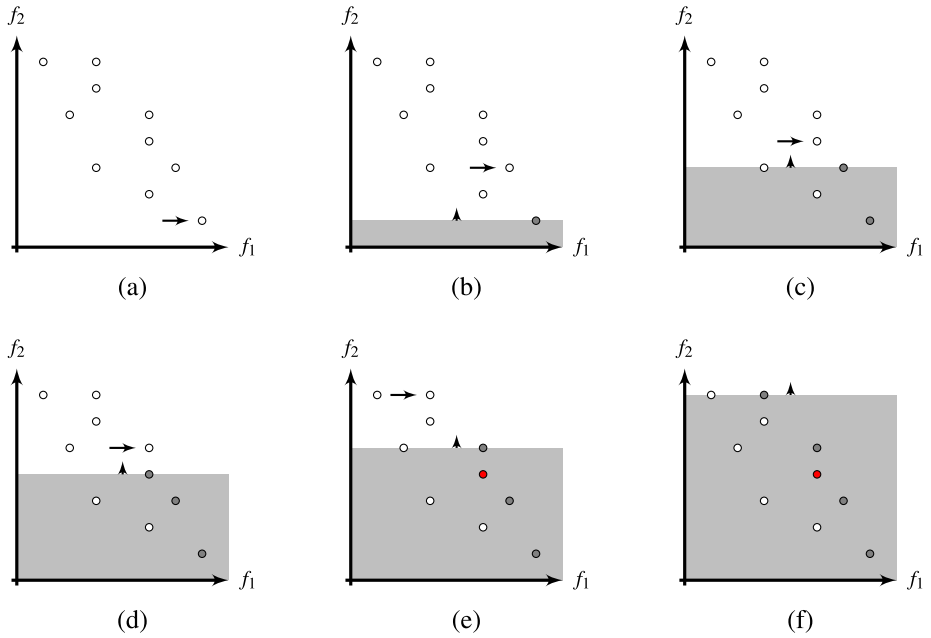


Fig. 4. Illustrated example of running the ε-constraint algorithm in order to maximize two objectives, f_1 and f_2: (a) find an optimum solution of f_1; (b) restrict the solution space according to the f_2 coordinate of the solution found and look for an optimum solution of f_1 in that space; (c) repeat the previous step, notice that in this step there are two optimal solutions for f_1 and the single objective solver returns a solution that is weakly efficient; (d) the next solution is found, it dominates the previous one; (e) again there are two optimal solutions for f_1 but here the single objective solver immediately finds the non-dominated one; and (f) there are no more solutions to be found, the problem is infeasible and the algorithm ends. The weakly efficient (dominated) solutions need to be filtered out of the set of solutions.

$x^1 \leftarrow \text{lexmax}_{x \in X}(f_1(x), f_2(x))$
$x^2 \leftarrow \text{lexmax}_{x \in X}(f_2(x), f_1(x))$
$A \leftarrow \{x^1, x^2\}$
$S \leftarrow \{(x^1, x^2)\}$
while stopping criterion is not met **do**
  $(x^r, x^s) \leftarrow \arg\max_{(x,x') \in S} \|(x, x')\|$
  $\lambda \leftarrow \frac{f_2(x^r) - f_2(x^s)}{f_2(x^s) - f_2(x^s) + f_1(x^s) - f_1(x^r)}$
  $x \leftarrow \max_{x \in X}\{\lambda f_1 + (1 - \lambda) f_2\}$
  $A \leftarrow A \cup x$
  $S \leftarrow \{(x^r, x), (x, x^s)\} \cup S \setminus \{(x^r, x^s)\}$
end while

Fig. 5. Anytime Dichotomic Scheme.

6.6.1. Number of exact solutions found

The number of solutions of the exact front that are found on the approximate front is probably the simplest unary measure.

Other than what it measures, it cannot be used by itself to compare two approximate fronts. Say front A has 5 well-spaced solutions on the exact front and front B has 10 solutions on the exact front but clustered together. Using this measure B would be better than A.

6.6.2. Hypervolume

The hypervolume indicator computes the space that is dominated by the solutions in the front [45]. With two objectives, this actually corresponds to the union of the area of all the rectangles that are dominated by the solutions, e.g. the shaded area in Fig. 2. The area of the rectangle is defined w.r.t. the solution's objective values and a reference point p. This point corresponds to the worst values for each objective. The hypervolume indicator is denoted by $I_{\mathcal{H}}$.

$$I_{\mathcal{H}}(A, p) = \bigcup_{a \in A} \text{area}(a, p)$$

An advantage of the hypervolume is that it is strictly Pareto-compliant, i.e., if front A dominates front B then $I_{\mathcal{H}}(A) > I_{\mathcal{H}}(B)$. One drawback is that it is biased towards convex parts of the front.

7. Experiments on the 2-objective NRP

7.1. Statistical methodology

We are performing comparisons with a stochastic algorithm, NSGA-II, which generates a different result for each execution. However both the ε-constraint method and ADS are deterministic. To evaluate the "better" option between NSGA-II and ADS, we mainly use boxplots of NSGA-II results plotted against the corresponding ADS single result, the difference between them being very obvious. We also use *Cliff's test* [46, pp. 180–185], at the 0.05 significance level, to corroborate this.

Quantifying the extent to which the null hypothesis is false, i.e., the effect size, is important. However in this paper we dispense from explicitly measuring it since the differences that matter to us are clearly visible on boxplots.

7.2. Experimental setup

For this set of experiments, we use the same base setup as described in Section 5.1. All the programs are coded in C++ and NSGA-II is implemented using the ParadisEO framework [47]. It is key to note that the use of the `-O3` compiler optimization flag plays a very important part in making the NSGA-II algorithm fast since it roughly produces a fivefold speedup on our machine. The deterministic algorithms are run only once.

For the ε-constraint method, since cost is an integer function for all instances, $\sigma = 1$ and the search space is progressively restricted w.r.t. cost.

The NSGA-II parameter values are mostly those used by Zhang et al. [13]: a population of 500, a tournament size of 5, a cross-over probability $P_c = 0.8$ and a mutation probability $P_m = 1/n$ where n

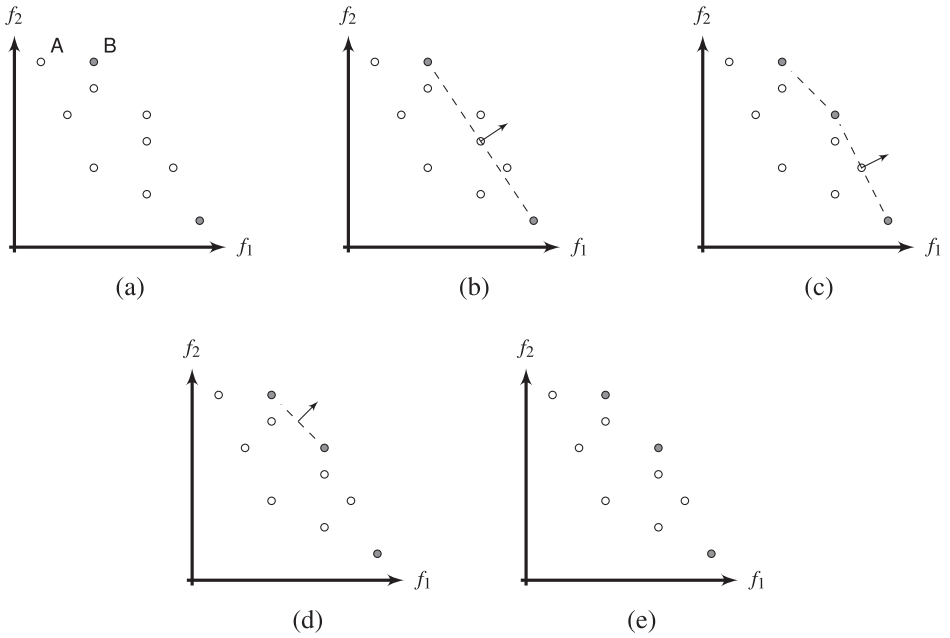


Fig. 6. Illustrated example of the Anytime Dichotomic Scheme to maximize two objectives, f_1 and f_2: (a) find a lexicographically optimal solution for the orders (f_1,f_2) and (f_2,f_1); A and B are two possible solutions for f_2 alone but if we then consider f_1, B is the lexicographically optimal solution; (b) find the weight λ defining a perpendicular line through the segment joining the two points, find an optimum solution for $\lambda f_1 + (1-\lambda)f_2$; (c) choose the longest segment and repeat the previous step, this time there are no solutions beyond the segment; there is a point on the segment which could potentially be found, assume it is not; (d) no new solution is found for this segment; and (e) there are no longer any segment to examine and the algorithm ends.

```
Create-Random-Population(P, N)
Evaluate(P)
Dominance-Depth-Rank-Assignment(P)
Front-by-Front-Crowding-Distance-Assignment(P)
while stopping criterion is not met do
  Make-New-Population(P)
  Evaluate(P)
  Dominance-Depth-Rank-Assignment(P)
  Front-by-Front-Crowding-Distance-Assignment(P)
  Sort-by-Rank-then-Distance(P)
  Resize(P,N)
end while
```

Fig. 7. High-level view of the NSGA-II algorithm.

is the number of boolean decision variables. However, we determined that we needed to increase the number of generations from 100 to 200 in the case of the *classic* and *realistic* instances, mainly to deal with larger instance size and complexity. Seeding is set to 2%. These two choices were determined by performing an exploratory analysis that is described in the supplemental material. For each instance, NSGA-II is executed 30 times.

7.3. Answers to research questions

RQ2.1 How efficient is it to compute the exact Pareto front of the NRP with two objectives?

Table 2 presents the results for the ε-constraint method with CPLEX, NSGA-II with seeding and the Anytime Dichotomic Scheme with CPLEX on the *classic* and *realistic* instances w.r.t. number of solutions, run time and hypervolume. Plots for the hypervolume are provided in Fig. 8.

We can answer RQ2.1 immediately. The longest run time is over 8 h and several instances take less than an hour. This is again testimony to the improved performance of ILP solvers since in 2001 the longest single-objective run was stopped after 20 h [1]. Obviously, execution times of the order of the hour allow for less flexible use cases but can be useful nonetheless.

For the smallest *classic* instance, *nrp1*, all solutions are found in 24 s. The real-world *Motorola* and *RALIC* instances are solved even more quickly since they are very small and feature no or few constraints. The instance producing the longest run, *nrp4*, is also the one with the second largest number of solutions. The next most computationally expensive instance is *nrp2*. Both these instances share the characteristic of having been generated with five levels of requirement dependency, compared with three for the other *classic* instances and no dependencies for the *realistic* instances.

All but one of the *classic* and *realistic* instances produce fronts with several thousands of solutions. Although choice is usually a good thing from a decision maker's point of view, such a large number of solutions is probably unnecessary in most circumstances. The greater the number of solutions, the longer the run time, therefore this leads us to the next research question.

RQ2.2 How efficient is it to obtain an approximation of the Pareto front that can be considered good enough?

Fig. 9 presents the empirical attainment functions [48] of the various algorithms for the *nrp1* and *nrp-e1* instances. These functions describe the probabilistic distributions of the solutions obtained by stochastic algorithms in the objective space. They allow us to identify the worst, median and best fronts, as well as the empirical probability of obtaining the solutions. For deterministic algorithms, an empirical attainment function is simply the front produced.

Let us examine the NSGA-II results in Table 2. The seeded NSGA-II solutions are well distributed but an overwhelming majority of them do not reach the real Pareto front, especially for larger instances. For smaller instances where NSGA-II performs well, the exact method finds all solutions within a shorter time-frame.

Table 2
Bi-objective results. The table presents the number of solutions, time and hypervolume for the ε-constraint method (Exact), NSGA-II and the Anytime Dichotomic Scheme (ADS). The values for NSGA-II are means over 30 runs and the rest use single-run values. *All* and *Conv* correspond to the number of solutions of the exact Pareto front and of its convex hull respectively. The numbers between parentheses are the number of solutions found by NSGA-II that are on the exact Pareto front. The time for NSGA-II corresponds to the mean time for 200 generations except for the *Motorola* and *RALIC* instances where 100 generations are used. The NSGA-II mean time for each instance is used as cutoff for ADS. The raw hypervolume value is given for the exact front and as a percentage of it for NSGA-II and ADS. Plots for the hypervolume are presented in Fig. 8.

Instance	Number of solutions				Time (s)		Hypervolume		
	Exact		NSGA-II	ADS	Exact	NSGA-II	Exact	NSGA-II (%)	ADS (%)
	All	Conv							
nrp1	465	27	206 (29.6)	28	24	53.06	$1{,}32{\cdot}10^{6}$	95.4	86.9
nrp2	4540	70	305 (1.67)	72	12,217	63.78	$3{,}7{\cdot}10^{7}$	74.1	51.1
nrp3	6296	172	327 (1.80)	181	1748	69.29	$5{,}85{\cdot}10^{7}$	81.0	96.7
nrp4	13,489	195	345 (1.77)	101	29,264	70.78	$2{,}17{\cdot}10^{8}$	73.5	90.1
nrp5	2898	264	324 (1.60)	301	435	65.90	$5{,}6{\cdot}10^{7}$	70.9	99.6
nrp-e1	10,331	309	348 (1.63)	201	3426	79.27	$1{,}34{\cdot}10^{8}$	89.3	99.6
nrp-e2	10,572	300	345 (1.43)	131	5798	76.45	$1{,}52{\cdot}10^{8}$	89.3	99.4
nrp-e3	8344	268	346 (1.67)	248	1726	75.93	$8{,}95{\cdot}10^{7}$	90.2	99.6
nrp-e4	8303	257	339 (1.40)	224	2384	76.86	$8{,}83{\cdot}10^{7}$	90.4	99.6
nrp-g1	9280	233	358 (1.83)	224	1534	78.80	$1{,}09{\cdot}10^{8}$	90.6	99.4
nrp-g2	6393	209	335 (1.47)	201	1153	75.66	$7{,}56{\cdot}10^{7}$	91.6	99.5
nrp-g3	8457	228	353 (1.60)	223	1222	79.24	$9{,}64{\cdot}10^{7}$	91.1	99.4
nrp-g4	6171	201	341 (1.57)	202	786	76.53	$5{,}97{\cdot}10^{7}$	92.3	99.5
nrp-m1	13,773	351	341 (1.50)	132	7562	81.09	$2{,}25{\cdot}10^{8}$	86.9	99.3
nrp-m2	12,933	329	341 (1.40)	111	8625	78.08	$1{,}96{\cdot}10^{8}$	88.8	99.3
nrp-m3	12,624	324	364 (1.60)	173	5091	82.00	$1{,}93{\cdot}10^{8}$	86.3	99.5
nrp-m4	11,547	295	350 (1.40)	161	4949	78.93	$1{,}49{\cdot}10^{8}$	88.4	99.4
Motorola	36	21	36.0 (34.1)	21	0.05	20.17	$4{,}74{\cdot}10^{4}$	99.9	98.8
RALIC	1165	45	294 (234)	45	4.26	65.42	$1{,}02{\cdot}10^{4}$	97.0	98.9

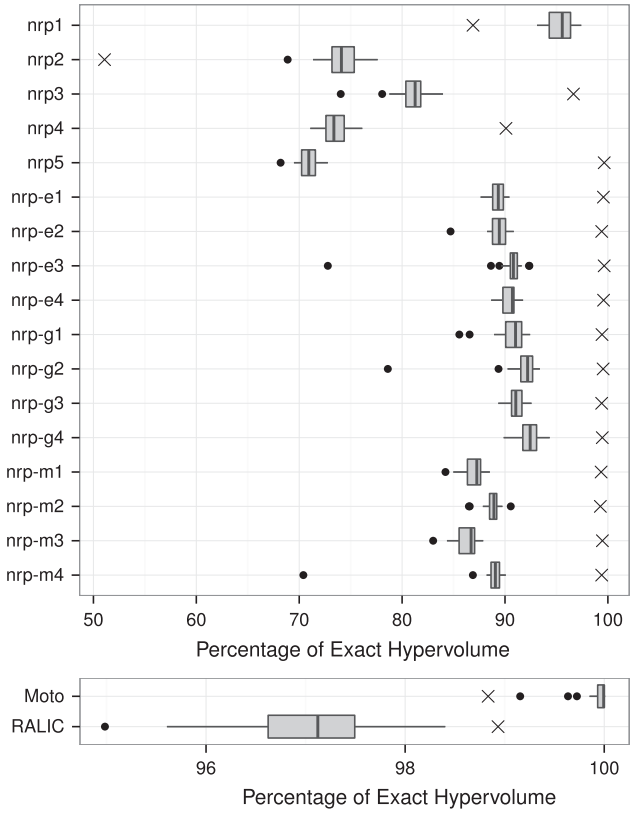


Fig. 8. Hypervolume for NSGA-II (boxplots) and the Adaptive Dichotomic Scheme (crosses). Relevant numbers are presented in Table 2. Naturally, had the exact method's results been plotted, they would be at the 100% mark. Notice that the *x*-axis scale for *Motorola* and *RALIC* instances is different from the other instances.

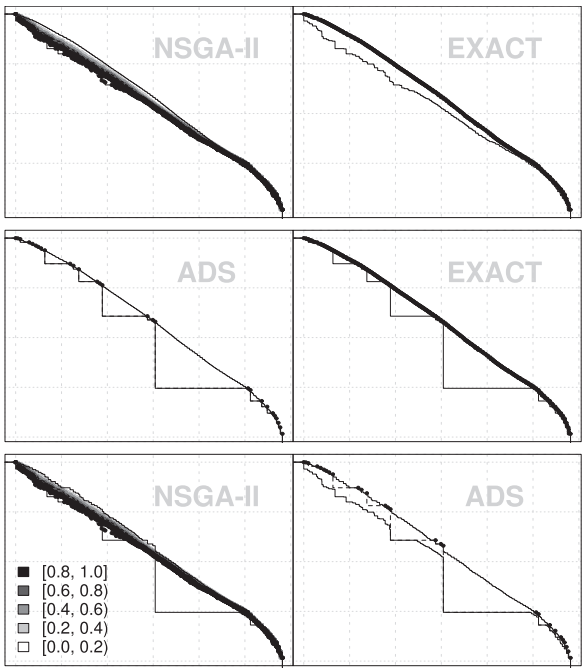
The Anytime Dichotomic Scheme produces larger hypervolume values than NSGA-II for all but the first two *classic* instances and the *Motorola* instance. On Fig. 9, we can see that the 28 solutions found by ADS for *nrp1* are too few and too unevenly spread to accurately describe the Pareto front. One shortcoming of ADS is that it is only able to find extreme supported solutions and, possibly, some non-extreme supported ones. In other words, if the Pareto front has some concave regions, these will not be discovered. The *Conv* column in Table 2 gives the number of solutions that define the convex hull of the exact front, i.e, the extreme supported solutions. This represents a tiny fraction of the total number of solutions, thereby being ADS' limiting factor. However, for the instances considered, it appears that, once roughly 100 or more extreme solutions have been found, these can provide a fairly good approximation of the Pareto front and result in large hypervolume values.

The Anytime Dichotomic Scheme sometimes finds more solutions than the number of solutions in the convex hull. Recall from Section 6.1 that the convex hull is composed of the extreme supported solutions. When the Anytime Dichotomic Scheme ends within its time limit, it will find all of these solutions and may also find some non-extreme supported solutions (Section 6.4), which explains the difference.

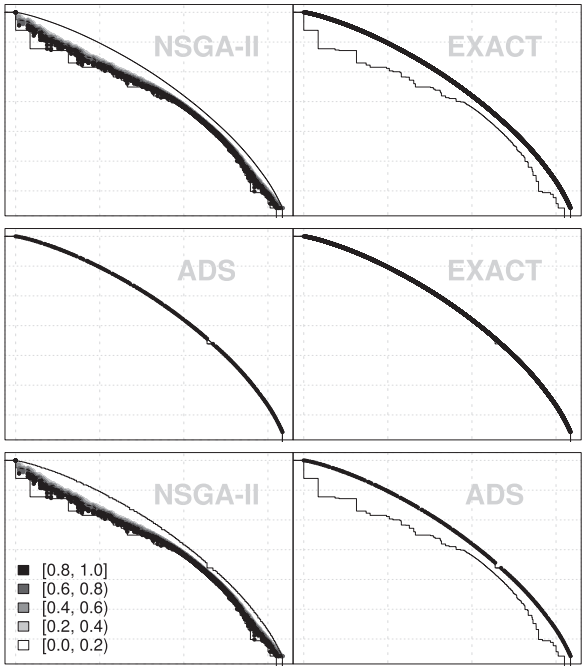
The *Motorola* instance is very small and has no constraints. We can observe that these conditions are beneficial to the NSGA-II performance leading to most of the actual Pareto being discovered and thus to an almost perfect hypervolume. We should nevertheless reiterate that small and simple instances are extremely easy for the exact method leading it find the Pareto front extremely quickly.

To answer RQ2.2, we can first note that both approximate methods presented here seem to give results of reasonable quality within 90 s. However, there is no single method that could be recommended in all cases. Considering the algorithms we tested, the best choice is likely to follow the scenarios below:

- If the instance is small enough, the ε-constraint method with an ILP solver such as CPLEX can generate the exact front quickly.
- If the Pareto front is "convex enough" and the extreme supported points are relatively evenly distributed, both of which depend largely on the user's appreciation, the Adaptive Dichotomic Scheme with an ILP solver can produce a good approximation of the front. The solutions obtained are a subset of the real Pareto front.



(a) nrp1



(b) nrp-e1

Fig. 9. Pairwise comparison of the Empirical Attainment Functions for the *nrp1* and *nrp-e1* instances. The algorithms involved are indicated in the upper right corner. Two solid lines bound the area containing solutions for the two compared algorithms. The lower line connects the worst solutions and the upper line connects the best solutions for both algorithms. When it is visible, the dashed line corresponds to the median attainment surface of the algorithm. For the points, the degree of shading corresponds to their probability as given by the bottom left legend. We dispense from labeling the axes, representing the two objectives, since we are interested in the shape of the fronts.

- If the Pareto front is non-convex, the seeded NSGA-II algorithm provides a good approximation but the solutions obtained are suboptimal for most of them.
- Considering that the combined runtime for the Adaptive Dichotomic Scheme and seeded NSGA-II is below three minutes, it seems that running both and combining their results would be entirely reasonable.

8. Threats to validity

In this section we discuss the validity of the results [49] presented in this paper.

Construct validity refers to the degree a test measures what it claims it is measuring. For the bi-objective case, metrics are required to assess the quality of the fronts. These produce scalar values that obviously cannot describe the entirety of a set of solutions. Furthermore, the quality of a front is potentially subjective to the decision maker's appraisal. We have used plots to determine if the fronts "looked" good enough.

Internal validity is concerned with the causal relationships that are examined. This is typically the case when considering randomized algorithms. We investigate whether some algorithm provides better results but then the randomness of the process plays a part as well. This is mitigated by performing a number of independent runs with the same (algorithm, instance) couple. In the present study we chose to have 30 runs and used strict statistical procedures to evaluate the results. In addition, we have not relied entirely on *p*-values and also used boxplots to gauge the differences between the algorithms.

External validity deals with the extent to which it is possible to generalize the results. We tested the different algorithms on instances with specific characteristics which will obviously not be necessarily the same for new instances. Nevertheless, the instances used were relatively varied in terms of number of variables, number and type of constraints. We would expect to obtain similar results on instances that do not entirely depart from these. Another issue is that most of our instances are not completely real-world instances. The two real-world instances are quite small and can easily be solved exactly. Yet there are probably larger real-world problem instances. The performance of the different algorithms on the *classic* synthetic instances and the *realistic* part real-world part synthetic instances provide a rough idea of what one could expect on larger real-world instances.

9. Summary and conclusion

This paper examines the use of Integer Linear Programming to solve the single and the bi-objective Next Release Problem. This allows us to exactly solve large single objective instances very quickly.

For the bi-objective case, we show that, in conjunction with the ε-constraint method, ILP can be used to generate the exact Pareto fronts. On smaller instances this is very fast but can take several hours for larger, more complex instances. Comparing the exact method with NSGA-II and the Anytime Dichotomic Scheme, using ILP, shows that these approximate methods can provide relatively good approximations of the fronts. ADS is particularly useful on the larger instances where it clearly outperforms NSGA-II. For instances with non-convex fronts, which can be problematic for ADS but less so for NSGA-II, we suggest pairing the results of both approaches to obtain the best of both worlds.

Based on those observations, we conclude that ILP is a useful approach for accurately solving the single and bi-objective NRP and can be used within an approximate algorithm when time is an issue.

Determining how well the different approaches presented will scale to problems that are quite different to the ones studied is beyond the scope of this paper. Nevertheless, using ILP in single

objective mode with several thousand more requirements is quite conceivable. Beyond two objectives, the ε-constraint method remains usable since the same principle of progressively restricting the search space may be carried out across several dimensions. Obviously performance may be an issue. Going beyond three objectives with ADS involves various geometrical challenges: in two dimensions, line segments are considered; in three dimensions, faces will need to be considered; but in n dimensions $(n-1)$-polytopes need to be manipulated.

In the future, the seeding mechanism could be enhanced by incorporating non-trivial initial solutions obtained through ILP. An alternative hybrid approach could incorporate ILP within a genetic algorithm in the context of a memetic algorithm. This paper deals with data that we assume exact in order to produce exact results. A logical progression of this work will be to investigate the results' sensitivity to potential changes in user-provided data. A related concern is how to perform robust optimization when dealing with instances that contain uncertain data, as is often the case in the real world.

Acknowledgments

This work has been funded under the DAASE project, EPSRC programme grant EP/J017515/1. The authors would like to thank Dr. Y. Zhang from University College London (UCL) for kindly providing the real-world instances used in this paper and for discussions on the NRP. IBM graciously provided an academic license for the IBM ILOG CPLEX Optimizer.

References

[1] A. Bagnall, V. Rayward-Smith, I. Whittley, The next release problem, Inf. Softw. Technol. 43 (14) (2001) 883–890, http://dx.doi.org/10.1016/S0950-5849(01)00194-X.
[2] D. Greer, G. Ruhe, Software release planning: an evolutionary and iterative approach, Inf. Softw. Technol. 46 (4) (2004) 243–253, http://dx.doi.org/10.1016/j.infsof.2003.07.002.
[3] Y. Zhang, M. Harman, S.A. Mansouri, The multi-objective next release problem, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, USA, 2007, pp. 1129–1137, http://dx.doi.org/10.1145/1276958.1277179.
[4] A.M. Pitangueira, R.S.P. Maciel, M.d.O. Barros, A.S. Andrade, A systematic review of software requirements selection and prioritization using SBSE approaches, in: G. Ruhe, Y. Zhang (Eds.), Search Based Software Engineering, Lecture Notes in Computer Science, vol. 8084, Springer, Berlin Heidelberg, 2013, pp. 188–208.
[5] M. Feather, T. Menzies, Converging on the optimal attainment of requirements, in: IEEE Joint International Conference on Requirements Engineering, 2002. Proceedings, 2002, pp. 263–270. (http://dx.doi.org/10.1109/ICRE.2002.1048537).
[6] G. Ruhe, D. Greer, Quantitative studies in software release planning under risk and resource constraints, in: 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings, 2003, pp. 262–270. (http://dx.doi.org/10.1109/ISESE.2003.1237987).
[7] G. Ruhe, A.N. The, Hybrid intelligence in software release planning, Int. J. Hybrid Intell. Syst. 1 (1) (2004) 99–110.
[8] G. Ruhe, M. Saliu, The art and science of software release planning, IEEE Softw. 22 (6) (2005) 47–53, http://dx.doi.org/10.1109/MS.2005.164.
[9] G. Ruhe, Product Release Planning: Methods, Tools and Applications, CRC Press, 2011.
[10] H. Jiang, J. Zhang, J. Xuan, Z. Ren, Y. Hu, A hybrid ACO algorithm for the next release problem, in: 2010 2nd International Conference on Software Engineering and Data Mining (SEDM), 2010, pp. 166–171.
[11] J. Xuan, H. Jiang, Z. Ren, Z. Luo, Solving the large scale next release problem with a backbone-based multilevel algorithm, IEEE Trans. Softw. Eng. 38 (5) (2012) 1195–1212, http://dx.doi.org/10.1109/TSE.2011.92.
[12] J.T. de Souza, C.L.B. Maia, T. do Nascimento Ferreira, R.A.F. do Carmo, M.M.A. Brasil, An ant colony optimization approach to the software release planning with dependent requirements, in: M.B. Cohen, M.O. Cinnéide (Eds.), Search Based Software Engineering, Lecture Notes in Computer Science, vol. 6959, Springer, Berlin Heidelberg, 2011, pp. 142–157.
[13] Y. Zhang, M. Harman, S.L. Lim, Empirical evaluation of search based requirements interaction management, Inf. Softw. Technol. 55 (1) (2013) 126–152, http://dx.doi.org/10.1016/j.infsof.2012.03.007.
[14] J. del Sagrado, I. del Aguila, F. Orellana, Ant colony optimization for the next release problem: a comparative study, in: 2010 Second International Symposium on Search Based Software Engineering (SSBSE), 2010, pp. 67–76. (http://dx.doi.org/10.1109/SSBSE.2010.18).
[15] J.J. Durillo, Y. Zhang, E. Alba, M. Harman, A.J. Nebro, A study of the bi-objective next release problem, Empirical Softw. Eng. 16 (1) (2011) 29–60, http://dx.doi.org/10.1007/s10664-010-9147-3.
[16] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, Y. Zhang, A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making, Requir. Eng. 14 (4) (2009) 231–245, http://dx.doi.org/10.1007/s00766-009-0075-y.
[17] M. Harman, J. Krinke, I. Medina-Bulo, F. Palomo-Lozano, J. Ren, S. Yoo, Exact scalable sensitivity analysis for the next release problem, ACM Trans. Softw. Eng. Methodol. 23 (2) (2014) 19:1–19:31, http://dx.doi.org/10.1145/2537853.
[18] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[19] E. Balas, E. Zemel, An algorithm for large zero-one knapsack problems, Oper. Res. 28 (5) (1980) 1130–1154.
[20] R. Beier, B. Vöcking, Random knapsack in expected polynomial time, in: Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, STOC '03, ACM, New York, NY, USA, 2003, pp. 232–241, http://dx.doi.org/10.1145/780542.780578.
[21] G.L. Nemhauser, Z. Ullmann, Discrete dynamic programming and capital allocation, Manage. Sci. 15 (9) (1969) 494–505.
[22] J.M. van den Akker, S. Brinkkemper, G. Diepen, J. Versendaal, Determination of the next release of a software product: an approach using integer linear programming, in: Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'05, 2005.
[23] F.G. Freitas, D.P. Coutinho, J.T. de Souza, Software next release planning approach through exact optimization, Int. J. Comput. Appl. 22 (8) (2011) 1–8, http://dx.doi.org/10.5120/2607-3636.
[24] M. van den Akker, S. Brinkkemper, G. Diepen, J. Versendaal, Software product release planning through optimization and what-if analysis, Inf. Softw. Technol. 50 (1–2) (2008) 101–111, http://dx.doi.org/10.1016/j.infsof.2007.10.017.
[25] C. Li, J.M. van den Akker, S. Brinkkemper, G. Diepen, Integrated requirement selection and scheduling for the release planning of a software product, in: P. Sawyer, B. Paech, P. Heymans (Eds.), Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol. 4542, Springer, Berlin Heidelberg, 2007, pp. 93–108.
[26] A. Lodi, Mixed integer programming computation, in: M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, L.A. Wolsey (Eds.), 50 Years of Integer Programming 1958–2008, Springer, Berlin Heidelberg, 2010, pp. 619–645.
[27] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D.E. Steffy, K. Wolter, MIPLIB 2010, Math. Program. Comput. 3 (2) (2011) 103–163, http://dx.doi.org/10.1007/s12532-011-0025-9.
[28] T.H. Cormen, C.E. Leiverson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press, Cambridge, Mass, 2009.
[29] J. Bang-Jensen, G.Z. Gutin, Digraphs: Theory, Algorithms and Applications, Springer, 2008.
[30] S.L. Lim, A. Finkelstein, StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation, IEEE Trans. Softw. Eng. 38 (3) (2012) 707–735, http://dx.doi.org/10.1109/TSE.2011.36.
[31] S.L. Lim, Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation, Ph.D. Thesis, The University of New South Wales, Sydney, Australia, August 2010.
[32] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, 1998.
[33] M. Ehrgott, X. Gandibleux, Multiobjective combinatorial optimization – theory, methodology, and applications, in: M. Ehrgott, X. Gandibleux (Eds.), Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, International Series in Operations Research & Management Science, vol. 52, Springer, US, 2003, pp. 369–444.
[34] J. Dubois-Lacoste, M. López-Ibáñez, T. Stützle, Improving the anytime behavior of two-phase local search, Ann. Math. Artif. Intell. 61 (2) (2011) 125–154, http://dx.doi.org/10.1007/s10472-011-9235-0.
[35] Y.Y. Haimes, L.S. Lasdon, D.A. Wismer, On a bicriterion formulation of the problems of integrated system identification and system optimization, IEEE Trans. Syst. Man Cybern. 1 (3) (1971) 296.
[36] C.-L. Hwang, A.S.M. Masud, Multiple objective decision making — methods and applications – a state-of-the-art survey, Lecture Notes in Economics and Mathematical Systems, vol. 164, Springer-Verlag, Berlin, Heidelberg, 1979.
[37] L.N. Van Wassenhove, L.F. Gelders, Solving a bicriterion scheduling problem, Eur. J. Oper. Res. 4 (1) (1980) 42–48.
[38] Y.P. Aneja, K.P.K. Nair, Bicriteria transportation problem, Manage. Sci. 25 (1) (1979) 73–78, http://dx.doi.org/10.1287/mnsc.25.1.73.
[39] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast elitist multi-objective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–-197, http://dx.doi.org/10.1109/4235.996017.
[40] P.D. Surry, N.J. Radcliffe, Inoculation to initialise evolutionary search, in: T.C. Fogarty (Ed.), Evolutionary Computing, Lecture Notes in Computer Science, vol. 1143, Springer, Berlin Heidelberg, 1996, pp. 269–285.
[41] A. Hernandez-Diaz, C.A. Coello Coello, F. Perez, R. Caballero, J. Molina, L. Santana-Quintero, Seeding the initial population of a multi-objective evolutionary algorithm using gradient-based information, in: IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on

Computational Intelligence), 2008, pp. 1617–1624. (http://dx.doi.org/10.1109/CEC.2008.4631008).
[42] G. Fraser, A. Arcuri, The seed is strong: seeding strategies in search-based software testing, in: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), 2012, pp. 121–130. (http://dx.doi.org/10.1109/ICST.2012.92).
[43] A. Sayyad, J. Ingram, T. Menzies, H. Ammar, Scalable product line configuration: a straw to break the camel's back, in: 2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE), 2013, pp. 465–474. (http://dx.doi.org/10.1109/ASE.2013.6693104).
[44] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132, http://dx.doi.org/10.1109/TEVC.2003.810758.
[45] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms – a comparative case study, in: A.E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Computer Science, vol. 1498, Springer, Berlin Heidelberg, 1998, pp. 292–301.
[46] R.R. Wilcox, Introduction to Robust Estimation and Hypothesis Testing, third ed., Academic Press, 2012.
[47] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, E.-G. Talbi, ParadisEO-MOEO: a software framework for evolutionary multi-objective optimization, in: C.A.C. Coello, C. Dhaenens, L. Jourdan (Eds.), Advances in Multi-Objective Nature Inspired Computing, Studies in Computational Intelligence, vol. 272, Springer, Berlin Heidelberg, 2010, pp. 87–117.
[48] M. López-Ibáñez, L. Paquete, T. Stützle, Exploratory analysis of stochastic local search algorithms in biobjective optimization, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin Heidelberg, 2010, pp. 209–222.
[49] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Softw. Eng. 14 (2) (2009) 131–164, http://dx.doi.org/10.1007/s10664-008-9102-8.